Image and Vision Computing 89 (2019) 276–288

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Interpretable Relative Squeezing bottleneck design for compact convolutional neural networks model☆

Qi Zhao*, Jiahui Liu, Boxue Zhang, Shuchang Lyu, Nauman Raoof, Wenquan Feng

School of Electronics and Information Engineering, Beihang University, Beijing 100191, China

ARTICLE INFO

Article history:
Received 16 June 2019
Accepted 24 June 2019
Available online 7 July 2019

Keywords:
Image recognition
Compact CNN
Relative-Squeezing bottleneck
Learned group convolutions

ABSTRACT

Convolutional neural networks (CNN) are mainly used for image recognition tasks. However, some huge models are infeasible for mobile devices because of limited computing and memory resources. In this paper, feature maps of DenseNet and CondenseNet are visualized. It could be observed that there are some feature channels in locked state and some have similar distribution property, which could be compressed further. Thus, in this work, a novel architecture — RSNet is introduced to improve the computing efficiency of CNNs. This paper proposes Relative-Squeezing (RS) bottleneck design, where the output is the weighted percentage of input channels. Besides, RSNet also contains multiple compression layers and learned group convolutions (LGCs). By eliminating superfluous feature maps, relative squeezing and compression layers only transmit the most significant features to the next layer. Less parameters are employed and much computation is saved. The proposed model is evaluated on three benchmark datasets: CIFAR-10, CIFAR-100 and ImageNet. Experiment results show that RSNet performs better with less parameters and FLOPs, compared to the state-of-the-art baseline, including CondenseNet, MobileNet and ShuffleNet.

1. Introduction

Image classification is one of the most basic tasks of the image recognition. After AlexNet [1] won the ImageNet Challenge: ILSVRC 2012 [2] in 2012, the convolutional neural networks (CNN) started dominating computer vision field. Since then, researchers focused on exploring complex CNN model to obtain a higher performance [3-7]. In recent years, improvement on computing hardware makes the deep and complex model design possible. Accuracy was substantially improved, but it came at the cost of extensive parameters, training time, computing and memory resources [3, 8, 9]. Besides, deep models also bring the problem of gradient vanishing. As a result, some researches aim to design more computing efficient and lighter CNN models without deteriorating existing accuracy standards. This leads to the development of efficient CNNs with different techniques: removing redundant connections [10-14], using quantized weights [15-17] and improving network architectures [3, 8, 10, 11, 18-27].

Similarity Networks (SimNets) [28] use the similarity operator and global MEX pooling method to improve the capability of small-size networks. SqueezeNet [26] reduces the channels of 3 × 3 filters and partially substitutes them with 1 × 1 filters in order to simplify the networks without impeding network capability too much. MobileNet [24] uses depth-wise separable convolutions to build light weight networks, which work well on embedded devices. In the pursuit of compressed CNN models, engineering methods like weight quantization and encoding were also utilized like Binary-Weight-Networks [16] and XNOR-Net [29] represents weight and even feature maps with binary values, which are quite efficient in terms of memory and computation. Network Pruning is another technique used for model compression [13, 30, 31]. Moreover, global average pooling [32] was also proposed to cut down the number of fully-connected layers to design simple models.

Besides, bottleneck module design is focused by lots of researchers, and there are plenty of achievements reached.

NIN (Network in Network) [32] uses 1 × 1 convolutional layer for the first time to combine different feature maps. Inspired from NIN, GoogleNet [9] proposed inception module that contains bottleneck structure in the form of 1 × 1 convolutions. Inception module uses 1 × 1 convolutional layer to reduce feature outputs followed by 3 × 3 or 5 × 5 convolutional layers, which maintain the local perception capacity of model with lower computational cost.

☆ This paper has been recommended for acceptance by Sinisa Todorovic.

* Corresponding author.
E-mail address: zhaoqi@buaa.edu.cn (Q. Zhao).

https://doi.org/10.1016/j.imavis.2019.06.006

ResNet [3] training details gradient vanishing and reduce parameters greatly by using shortcut connections. Residual block [conv1 × 1-conv3 × 3-conv1 × 1] also implements bottleneck by using 1 × 1 convolution layer. However, because of the fusion operation at the end of the residual block, ResNet also involves additional parameters to expand the feature map dimension.

DenseNet [8] introduces direct connections from all layers before the subsequent layer. Within one dense block, by concatenating of previous feature maps, DenseNet could reduce computation by reusing them. It relatively improves computational efficiency and substantially reduces parameters. DenseNet employs 1 × 1 convolution as bottleneck layer in DenseBlock-B structure [BN+ReLU+conv1 × 1+BN+ReLU+conv3 × 3]. Though, it achieves obvious improvements but still requires plenty of FLOPs to meet desired accuracy levels.

CondenseNet [25] explicitly removes redundant features by using the idea of learned group convolutions (LGCs), which significantly improves computational efficiency. Group convolutions is proposed in AlexNet [1] to solve the problem of limited GPU memory. Now, group convolutions are widely used in small models to reduce computational cost. Multiple group convolutions have one side effect when stacked together: outputs of a certain channel group are derived from a small fraction of input channels. This behavior limits blocks information flow between channel groups and weakens representations. To mitigate this effect, ShuffleNet [27] investigates the effectiveness of shuffle layer that was used for a two-stage convolution [33]. CondenseNet takes a step forward and proposes LGCs to elude adaption to certain group of features, besides using permutation layer. CondenseNet basic block replaces 1 × 1 standard convolutions (G=1) with LGCs. Then the following 3 × 3 standard convolution is replaced by 3 × 3 group convolutions.

However, similar to DenseNet, the bottleneck layer in CondenseNet also has the fixed output channels both in shallow layer and deep layer. In this paper, our work shows that it generates superfluous feature maps both in initial layers and in deep layers. Introduction of increasing growth rate (IGR) somehow curbed these growth trends, but problems remained within the single dense block. Our basic block follows the structure of DenseNet and CondenseNet. The bottleneck later structures of ResNet, DenseNet and RSNet are shown in Fig. 1. Because DenseNet and CondenseNet share the same bottleneck layer design, so the bottleneck layer of CondenseNet is not presented here.

Based on the above observation, it could be found that reusing feature maps could reduce model parameters, and LGCs could reduce parameters and FLOPs also. However, the fixed bottleneck layer could be designed more flexible, by changing the output channels of each bottleneck layer. Thus, we propose RSNet to solve the problem of how to use feature maps more efficiently.

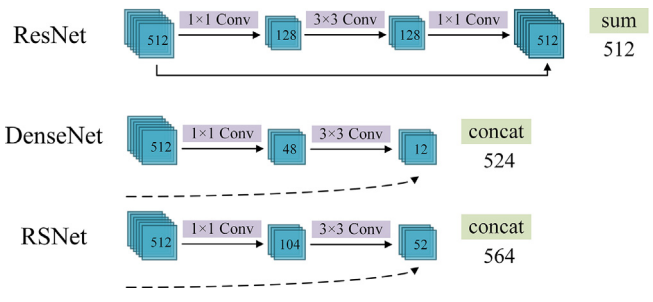

Fig. 1. Bottleneck design comparison of ResNet and DenseNet with relative squeezing bottleneck design of RSNet.

The main contributions of our work is summarized as follows:

1. A Relative Squeezing bottleneck module is proposed to improve the computing efficiency. By adopting an in-out correlation rule, the bottleneck layer will be set different output channels for each dense layer, even in the same dense block.
2. Multiple compression layers as well as LGCs layers are improved to reduce the redundant parameters.
3. Feature maps of DenseNet and CondenseNet are visualized to observe the feature maps state. Results show that there are some feature maps in lock state both in shallow and deep layers. Also, some feature maps share the similar patter after dimension reduction by T-sne. Besides, channel-wise similarity measurement of feature maps experiments show that the feature maps of RSNet is more expressive. These are motivations of the proposed RSNet, which will be presented in Section 4.

The organization of the rest of the paper is as follows; in Section 2, the proposed Relative Squeezing bottleneck layer is presented and the compression layer as well as transition layer are described. The performances of RSNet in CIFAR and ImageNet datasets are presented in Section 3. Further experimental results about feature maps of the model are analyzed in Section 4. Besides, this section also gives the insights of our motivation of proposing RSNet. Section 5 concludes the work.

2. Methods

RSNet is based upon four major principles to achieve a light, accurate and computing efficient CNN model:

1. Efficient bottleneck design combined with simple basic block that should transfer maximum information while generating moderate feature outputs.
2. Incorporating multiple compression layers, which limit the number of feature maps fed to next blocks and restrict transfer of features from initial layers to deeper layers. This results in reduction of computations, transfer of most useful features and increase in depth of model (which eventually helps to increase the classification accuracy) while generating relatively less parameters.
3. Shortcut connections from previous layers to exploit feature reuse so as to generate minimum feature outputs at each layer. This helps to achieve best bottleneck design besides reducing processing cost and number of parameters.
4. Implementing LGCs technique from CondenseNet, which intelligently learns most useful feature maps, eradicates superfluous features and groups relevant feature inputs.

2.1. Model architecture

RSNet is based upon RS blocks, multiple compression and transition layers. Fig. 2 shows the model architecture.

Initially, image is processed by 3 × 3 standard convolutional layer (G=1) to produce 48 output feature maps, which are forwarded to RS block. Architecture for CIFAR-10/100 datasets is composed of six RS blocks, three compression layers, two transition layers and linear classifier at the end. RS block contains multiple basic RS units, where output feature maps from all previous RS units are stacked together by using shortcut connections. Basic RS unit contains one 1 × 1 LGC layer and one 3 × 3 group convolutional layer. Each convolutional layer (except 1*st* layer) is organized as the form of BN-ReLU-Conv. RS units generate feature maps in bottleneck fashion, which is intelligibly designed by using a novel approach of Relative Squeezing *C* it is percentage squeezing of input channels. First

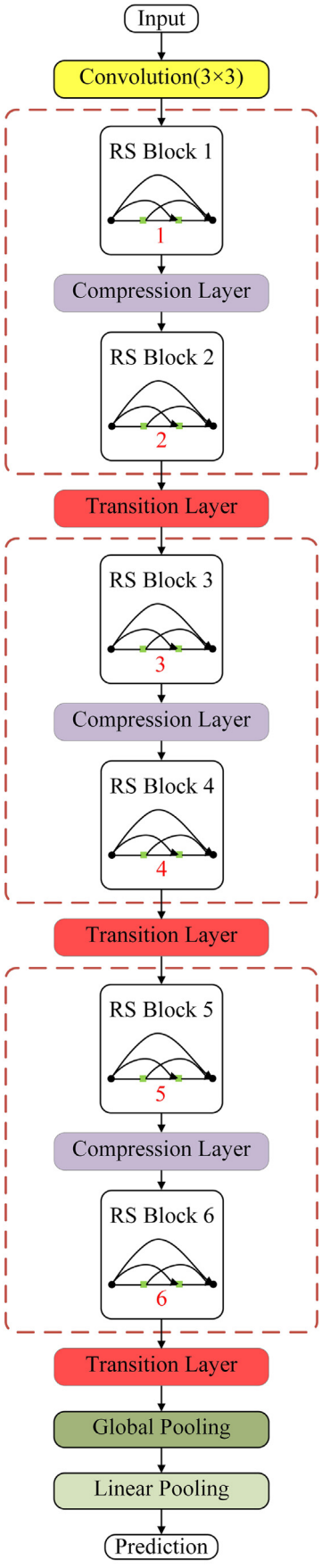

Fig. 2. RSNet architecture design for CIFAR datasets containing 32 × 32 sized images.

1 × 1 convolution uses LGCs technique [25] to intelligently select more expressive and relevant feature maps. Features generated by point-wise LGCs are forwarding to 3 × 3 group convolutional layer. Compression layers (comprising 1 × 1 group convolutional layer) and transition layers (comprising 1 × 1 group convolutional layer and average pooling layer) are alternately used between two RS blocks. Compression layer produces more powerful features and reduces the number of features transmitted to next RS block; meanwhile, it drops less informative features. Transition layer, in addition to compression layer function, downsamples the input feature maps; this alleviates computational efficiency by downsizing feature maps for next blocks.

2.1.1. Notations of the model

Here, the hyperparameters we used in our model are introduced as the following:

- S: squeezing ratios. S_1 and S_3 are squeezing ratios of 1 × 1 convolutions and 3 × 3 convolutions respectively.
- C: compression ratio, which should be between 0 and 1.
- G: convolution groups.
- B: number of basic blocks in each RS block.
- y: block output channels before concatenation operation.
- Y: block output channels after concatenation operation.
- D_g: divisible function.

Trade-off between parameters and FLOPs can be achieved by gradually increasing or decreasing the number of basic blocks B for RS blocks while keeping other hyper-parameters constant. Increasing Basic block pattern (2 – 4 – 6 – 8 – 10 – 12, means layers number B in each basic block) for RS blocks, results in significant increase in parameters as compared to FLOPs and vice-versa.

2.2. Relative squeezing bottleneck design

We observe that the bottleneck design of DenseNet is based upon fixed number of outputs in a dense block despite different input channels of each dense layer. This fixed bottleneck design outputs superfluous channels at initial layers. Besides, in deep layers, it restricts feature propagation. We improve bottleneck design by proposing idea of Relative Squeezing: output channels of bottleneck layer are weighted squeezing of input channels. It generates outputs as percentage squeezing of input channels. This eliminates superfluous feature maps from initial layers, transfers the most useful features and propagates features to next layer as each percentage of input channels. Keeping the same accuracy, it effectively reduces the number of FLOPS and model parameters. Let y and Y denote basic block output channels before and after concatenation respectively. We introduce bottleneck design parameter S that is termed as squeezing ratio. Where S_1 and S_3 are squeezing ratios of 1 × 1 convolutions and 3 × 3 convolutions respectively:

$$y_i = D_g(D_g(Y_{i-1} * S_1) * S_3) \tag{1}$$

$$Y_i = Y_{i-1} + y_i \tag{2}$$

Number of output channels at layer l can be calculated as under:

$$Y_l = \sum_{i=1}^{l}(Y_{i-1} + D_g(D_g(Y_{i-1} * S_1) * S_3)) \tag{3}$$

Here, D_g represents divisible function to ensure that output features are divisible by number of groups G and value of S ranges between 0 and 1.

Bottleneck design comparison of ResNet, DenseNet and RSNet is given in Fig. 1. Input channels are kept 512 for comparison. ResNet bottleneck (Left) employs squeeze-expand design; last 1×1 convolutional layer generates too many features to satisfy the summation function requirement that drastically increases parameters and computations. DenseNet bottleneck (Middle) has very narrow design, which inhibits feature propagation and reduces the feature extraction efficacy. However, RSNet bottleneck (Right) has moderate number of output features and transfers powerful features to next layer.

Structural comparison of CondenseNet and our Relative Squeezing technique is given in Fig. 3. CondenseNet (Left) increases the growth rate for the next Block but still remains constant within each block. Moreover, its transition layer only downsamples the input features that produce a lot of parameters at later layers. In RSNet (Right), output features moderately increase with input features increasing. Compression layers in RSNet transfer less features to the next layer (only 252 input features in 47*th* basic unit as compared to 848 in case of CondenseNet), which makes its parameters less and compute efficient.

2.3. Compression layer

Compression layer contains only one 1×1 convolution layer and lies between two RS blocks. It does not contain any shortcut connections. In DenseNet, as block becomes deeper, features of initial layers are transferred to later layers through shortcut connections. Then the feature maps will be concatenated. Initial features lack the representational power and contain less useful information for deeper layers. Hence, compression layers are used to restrict the flow of less useful features and create more powerful feature representations for deeper layers. It helps to reduce the number of feature maps, transferred to next RS block, which results in similar classification accuracy with less computational requirements and parameters. Let

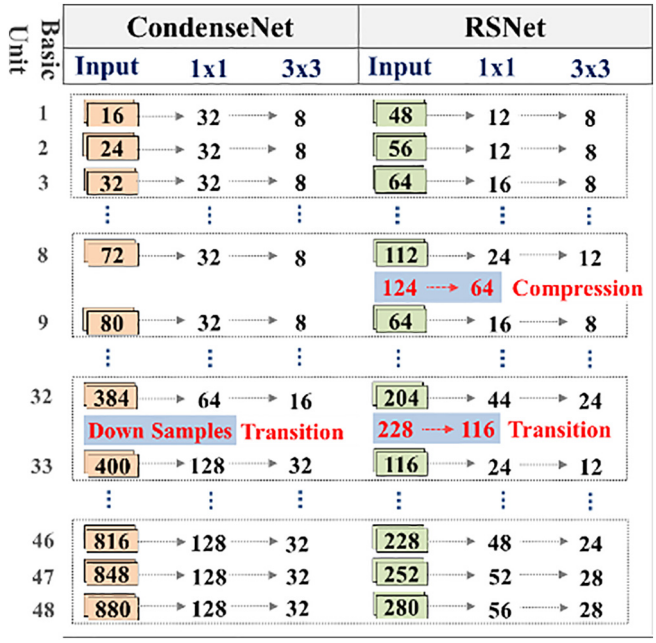

Fig. 3. Structural comparison of CondenseNet (Left) and RSNet (Right), comprising 48 basic blocks each. CondenseNet has increasing growth rate of $8 - 16 - 32$ and RSNet has $S_1 = 0.2$ and $S_3 = 0.5$.

C denote compression ratio ranging between 0 and 1; Y_c and Y_l are number of output channels of compression layer and RS block (with l number of layers) respectively, then:

$$Y_c = D_g(C * Y_l) \tag{4}$$

Structure design of compression layer is given in Fig. 3 (Middle).

2.4. Transition layer

Transition layer contains a compression layer followed by an average pooling layer. Design of transition layer is given in Fig. 3 (Right). Fig. 3 explains that in the first step, it compresses the number of channels and in the second step it reduces the size of feature maps. Combination of these layers not only reduces the number of feature maps and computational cost by down sampling, but also produces more expressive features. Like compression layer, it does not contain any shortcut connections and even not support concatenation operation as downsampling operation changes size of feature maps. To facilitate compression and downsampling, we divide the network architecture into multiple RS blocks and alternately insert compression and transition layers between these blocks as shown in Fig. 2.

2.5. Convolutional layers

2.5.1. Group convolutions

AlexNet introduces the idea of group convolutions to distribute the model over two GPUs and is effectively exploited in ResNet. Group convolutions work well in a lot of deep neural network architectures [27,33,34] and are intelligibly combined with depth-wise separable convolutions in ShuffleNet. All layers in our structure use group convolutions except first convolution layer. Group convolutions reduce computational cost by a factor G (comparing with cost when $G = 1$) to $F_{in} \times F_{out}/G$, by partitioning the input features into G mutually exclusive groups. Where, F_{in} and F_{out} are input and output features (Fig. 4).

2.5.2. Learned Group Convolutions (LGCs)

Huang et al. [8] explained the importance of early features for later layers but not all prior features are needed to every subsequent layers. It is difficult to predict which features are feasible at what point. So, we employ the idea of LGCs [25] to learn the features needed at subsequent layers besides selecting the most relevant features for group convolutions.

2.6. Model configure and training details

2.6.1. Configuration details

We follow the same settings for all experiments, unless otherwise specified. The 1*st* convolutional layer (3×3) implements standard convolutions ($G = 1$) and the rest of all convolutions are group convolutions. 1×1 Convolutions in RS Blocks use LGCs from CondenseNet and condensation factor used in [25] is reduced to number of groups. Output channels of convolutional layers in RS units and compression/transition layers are calculated by multiplying the squeezing ratio and compression ratio with input channels respectively; the resulting output is rounded off to upper ceiling to make it divisible by the number of groups. We have used both constant and descending squeezing ratios S_3 for training, while S_1 and C are kept for 0.2 and 0.5 respectively. Size of RS Blocks (B) is mentioned for major experiments. While these have been adjusted for comparative analysis.

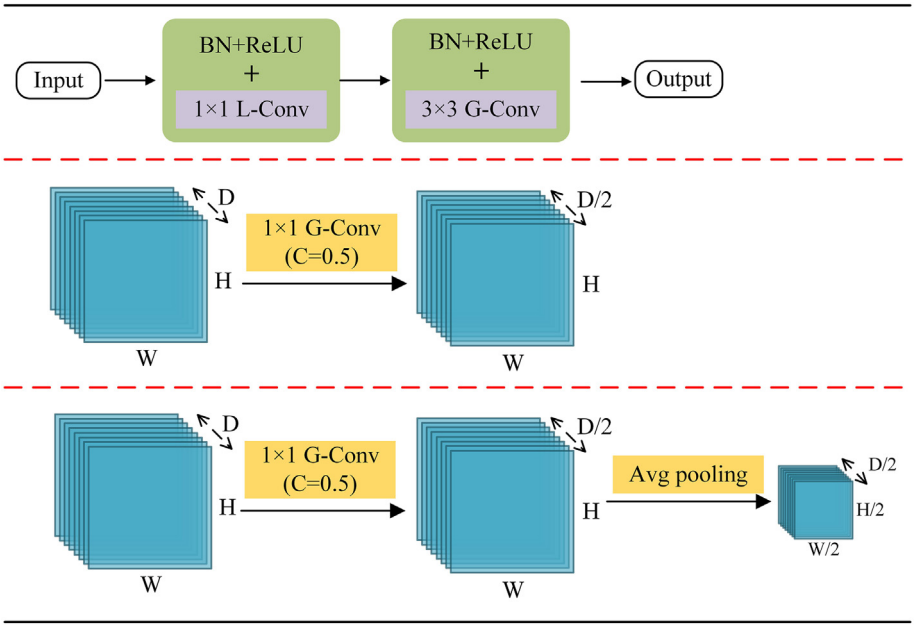

Fig. 4. Structure design of RS Unit (Up), compression layer (Middle) and transition layer (Below).

2.6.2. Training details

We follow most of the training settings used in [4]. Stochastic gradient descent (SGD) is used with momentum weight of 0.9 (Nesterov Momentum = False) and weight decay of 0.0001. We use a cosine shape learning rate starting from 0.1 and gradually reducing to 0. All models are trained for 200 epochs with mini-batch size 64 for CIFAR and 256 for ImageNet, unless otherwise specified. We do not use group lasso regularization [35], which is used in [25] for ImageNet experiments to minimize the negative effect of

Table 1
Comparison of classification error rate (%) with other convolutional networks on CIFAR-10 (C-10) and CIFAR-100 (C-100) datasets. * indicates models that are trained with cosine shape learning rate for 600 epochs. RSNet-135 uses constant squeezing ratios ($S_1 = 0.2$ and $S_3 = 0.5$). In RSNet-195, S_1 is set under the descending strategy and using [0.25, 0.25, 0.2, 0.2, 0.15, 0.15] in each block. Here, $S_3 = 0.35$.

Model	Para(M)	FLOP(M)	C-10	C-100
ResNet-1001 [41]	16.1	2357	4.62	22.71
Stochastic-Depth-1202 [37]	19.4	2840	4.91	-
Wide-ResNet-28 [42]	36.5	5248	4.17	20.50
ResNeXt-29 [34]	68.1	10,704	3.58	17.31
DenseNet-190 [8]	25.6	9388	3.46	17.18
NASNet-A* [43]	3.3	-	3.41	-
CondenseNet-182* [25]	4.2	513	3.76	18.47
RSNet-135(G=4)	1.4	203	3.75	19.56
RSNet-195(G=8)	3.0	455	3.65	18.61

Table 2
Comparison of classification error rate (%) on CIFAR-10 (C-10) and CIFAR-100 (C-100) with state of the art filter-level weight pruning methods and smaller models. RSNet-135 uses $S_1 = 0.2$. And S_3 is set under the descending strategy and using [0.6, 0.6, 0.5, 0.5, 0.4, 0.4] in each block. Model is trained for 300 epochs.

Model	Para(M)	FLOP(M)	C-10	C-100
VGG-16-pruned [13]	5.40	206	6.60	25.28
VGG-19-pruned [44]	2.30	195	6.20	-
VGG-19-pruned [44]	5.00	250	-	26.52
ResNet-56-pruned [45]	-	62	8.20	-
ResNet-56-pruned [13]	0.73	90	6.94	-
ResNet-110-pruned [13]	1.68	213	6.45	-
ResNet-164-B-pruned [44]	1.21	124	5.27	23.91
DenseNet-40-pruned [44]	0.66	190	5.19	25.28
CondenseNetlight-94 [25]	0.33	122	5.00	24.08
CondenseNet-86 [25]	0.52	65	5.00	23.64
RSNet-115(G=4)	0.32	63	4.83	23.23

Table 3
RSNet architecture for ImageNet; in is input features; stride= 2 for all average pooling layers.

Layer	Components	Output size	Output channels
Convolution	3 × 3Conv(stride=2)	112 × 112	48
RS	1 × 1L-Conv	112 × 112	in*S_1*S_3
Block	3 × 3G-Conv		
Compression	1 × 1L-Conv	112 × 112	in*C
RS	1 × 1L-Conv	112 × 112	in*S_1*S_3
Block	3 × 3G-Conv		
Transition	1 × 1L-Conv	56 × 56	in*C
	Avg pool		
RS	1 × 1L-Conv	56 × 56	in*S_1*S_3
Block	3 × 3G-Conv		
Compression	1 × 1L-Conv	56 × 56	in*C
RS	1 × 1L-Conv	56 × 56	in*S_1*S_3
Block	3 × 3G-Conv		
Transition	1 × 1L-Conv	28 × 28	in*C
	Avg pool		
RS	1 × 1L-Conv	28 × 28	in*S_1*S_3
Block	3 × 3G-Conv		
Compression	1 × 1L-Conv	28 × 28	in*C
RS	1 × 1L-Conv	28 × 28	in*S_1*S_3
Block	3 × 3G-Conv		
Transition	1 × 1L-Conv	14 × 14	in*C
	Avg pool		
RS	1 × 1L-Conv	14 × 14	in*S_1*S_3
Block	3 × 3G-Conv		
Compression	1 × 1L-Conv	14 × 14	in*C
RS	1 × 1L-Conv	14 × 14	in*S_1*S_3
Block	3 × 3G-Conv		
Transition	1 × 1L-Conv	7 × 7	in*C
	Avg pool		
RS	1 × 1L-Conv	7 × 7	in*S_1*S_3
Block	3 × 3G-Conv		
Compression	1 × 1L-Conv	7 × 7	in*C
RS	1 × 1L-Conv	7 × 7	in*S_1*S_3
Block	3 × 3G-Conv		
Pooling	Global Avg(7 × 7)	1 × 1	in
Linear	SoftMax	-	1000

Table 4
Comparison of Top-1 and Top-5 classification error rate (%) with other state-of-the-art compact models on ImageNet. RSNet models are trained for 120 epochs. RSNet-215 uses descending S_1 value but constant value of $S_3 = 0.35$. RSNet-197 uses $S_1 = 0.2$ and descending value of S_3 between 0.6 and 0.4.

Model	Para(M)	FLOP(M)	C-10	C-100
Inception V1 [9]	6.6	1448	30.2	10.1
1.0 MobileNet-224 [24]	4.2	569	29.4	10.5
ShuffleNet 2x [27]	5.3	524	29.1	10.2
NASNet-A(N=4) [43]	5.3	564	26.0	8.4
NASNet-B(N=4) [43]	5.3	488	27.2	8.7
NASNet-C(N=3) [43]	4.9	558	27.5	9.0
CondenseNet(G=C=8) [25]	2.9	274	29.0	10.0
CondenseNet(G=C=4) [25]	4.8	529	26.2	8.3
RSNet-215(G=4)	2.2	396	28.5	9.5
RSNet-197(G=4)	3.1	499	26.3	8.3

weight pruning and over-fitting. RSNet uses only group convolution in compression/transition layers to overcome this effect.

2.6.3. Decreasing squeezing ratio training strategy

We use decreasing squeezing ratio strategy to implement larger and deeper models. Higher S in initial RS blocks generates more output channels to accrue maximum advantage from diverse nature of feature maps at initial layers. Deeper layers have abstract feature maps and more number of input features. Thus, lower S in later blocks produces less number of feature outputs. It helps to achieve improved accuracy while producing less number of parameters. Moreover, RS block contains less number of RS units in case of small models and architectures with more compression layers; thus need higher S values to produce significant number of features before compression layer. Preliminary experiments demonstrate that decreasing S_3 gives better results than S_1.

3. Results

The proposed RSNet is assessed on three datasets: CIFAR-10, CIFAR-100 [36] and ImageNet (ILSVRC 2012) [20].

3.1. Datasets

CIFAR-10 and CIFAR-100 datasets contain RGB images of size 3232 pixels each corresponding to 10 and 100 classes, respectively. Both datasets contain 60,000 images, which are divided into 50,000 training and 10,000 test images. Standard data-augmentation scheme in [17,24,25, 32,37-40] has been implemented to avoid over-fitting. ImageNet contains 1.2 million training images and 50,000 validation images with a total 1000 classes. Data-augmentation scheme in [25] has been used at training time and rescaled to 256 before 224 center crop at test time.

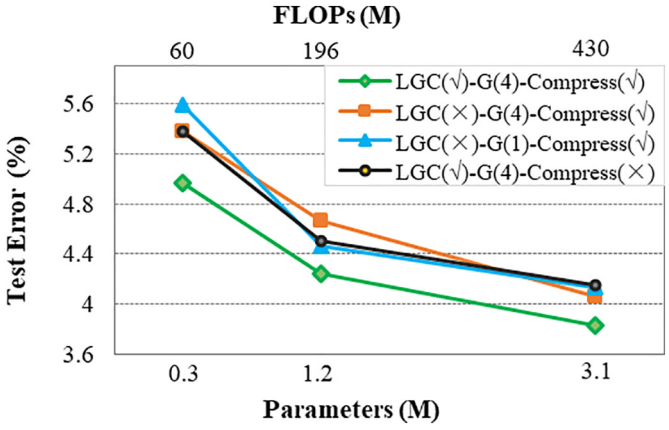

Fig. 5. Effect of different components on classification error rate for different size RSNet models. M means million.

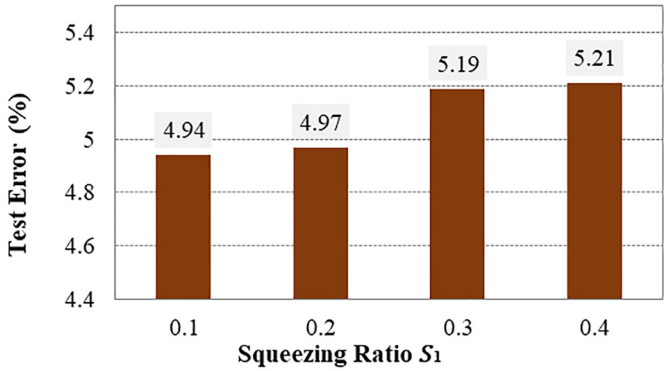

Fig. 6. Classification error rate comparison for different squeezing ratios (S_1). Parameters and FLOPs are the same in all the cases.

3.2. Results

3.2.1. CIFAR-10/100

We carry out experiments on CIFAR-10 to validate the efficacy of RSNet, with the new Relative Squeezing bottleneck layer. Then it will be further evaluated on CIFAR-100.

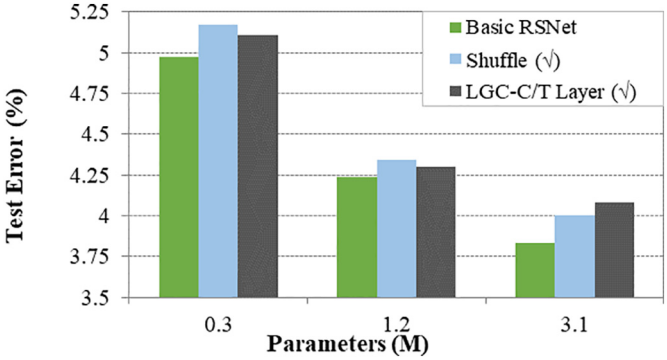

Fig. 7. Effect of using shuffle layer [between 11 and 33 convolutional layers] and 11 learned group convolutions (LGCs) [in compression/transition (C/T) layers] on classification error rate.

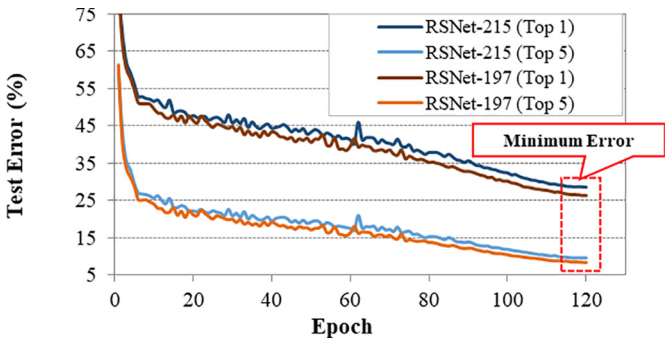

Fig. 8. Classification error rate curve of RSNet for ImageNet. Red dotted rectangle indicates that maximum accuracy lies in 120 epochs.

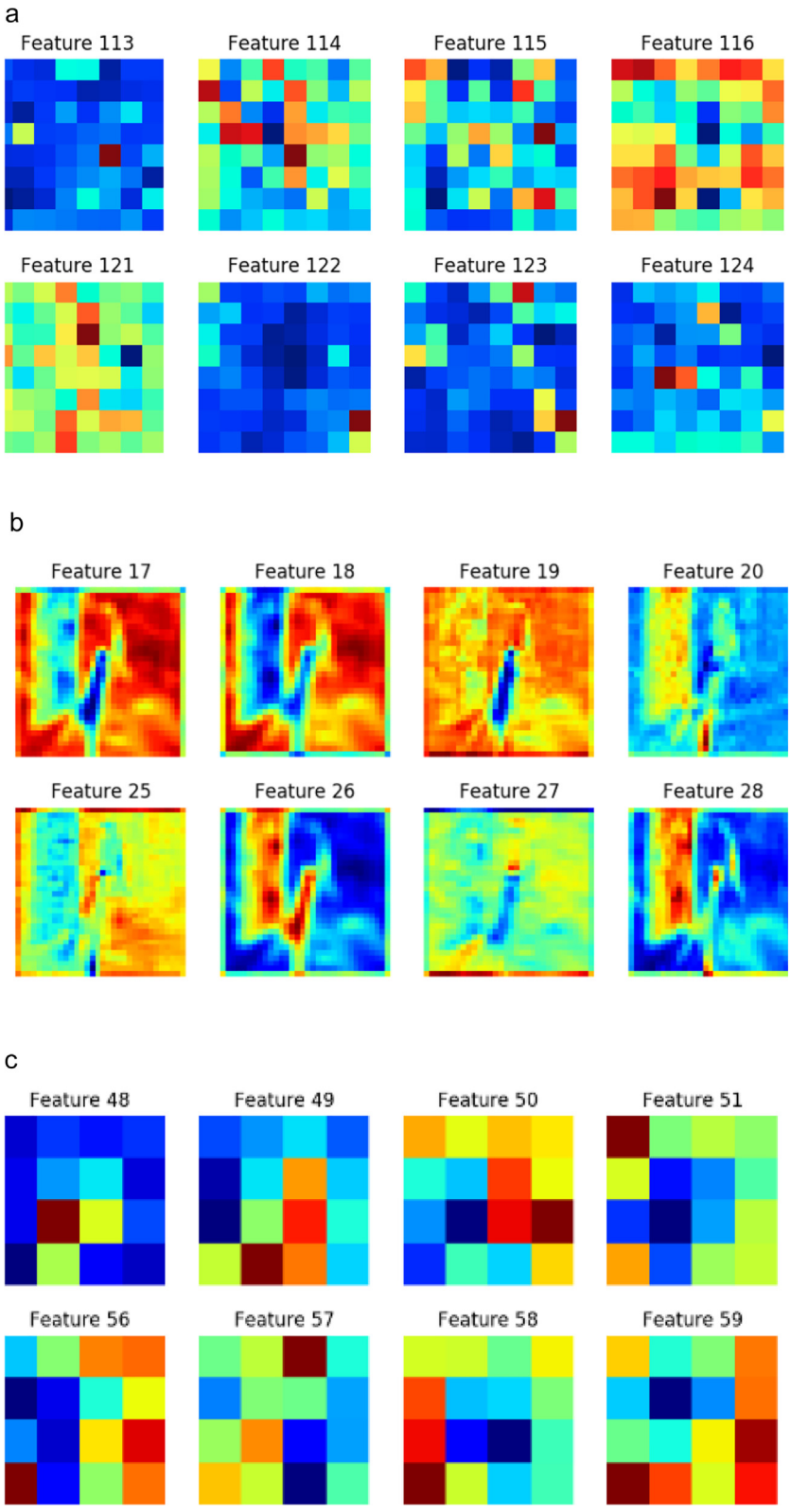

Fig. 9. Feature maps of CondenseNet and RSNet. (a) Feature maps of the final CondenseNet dense block; (b) feature maps of 1*st* RS block; (c) feature maps of 8*th* RS block.

RSNet architecture for CIFAR-10/100 is given in Fig. 2. InTable 1, we compare experimental results of RSNet-135 and RSNet-195 with other state-of-the-art CNN models. RSNet-135, with only 1/3 parameters and less than half Flops, achieves the same result on CIFAR-10. RSNet-195, with only 2/3 parameters and less Flops, surpasses CondenseNet-182 on CIFAR-10 and gives quite competitive result on CIFAR-100 as well.

In Table 2, experimental results show that RSNet-115 outperforms filter-level weight pruning techniques [13,44,45] in all aspects. It achieves higher accuracy than CondenseNet-86 with quite less

parameters and CondenseNetlight-94 with half the number of FLOPs, on CIFAR-10/100 datasets. Results are more pronounced on smaller models like RSNet-115 and RSNet-135, and perform extremely well.

3.2.2. ImageNet

RSNet architecture for ImageNet is given in Table 3. Most of the configuration and training details are already covered in Section 2.6. Compression layers along with moderate bottleneck design of RSNet help in designing a much deeper model. Table 4 compares RSNet results with state-of-the-art efficient CNNs. RSNet-215 outperforms CondenseNet (G=8) in both Top-1 and Top-5 accuracies with less parameters but more FLOPs. RSNet-197, with less than 2/3 times parameters of CondenseNet (G=4) and less FLOPs, achieves close results. However, more investigation is required to hit better results that could not be carried out due to paucity of time. In Fig. 8, classification error rate curves are continuously converging for 120 epochs, which shows that models still have capacity to converge and produce much better results with more number of epochs.

3.3. Ablation analysis

Ablation study has been carried out on CIFAR-10 to investigate the effect of various factors.

Experimental results in Table 1 show that the proposed bottleneck design helps to achieve better accuracy besides using less computations or parameters. We also analyze the effect of other components in Fig. 5. Our model (using LGCs, group convolutions and compression layers) performs better than models without LGCs, group convolutions and compression layers in all size models. This validates efficacy of bottleneck design and use of multiple compression layers. RSNet structures without group convolutions and compression layers have similar results except the small model, where RSNet without compression layers performs better. For large size model, RSNet structure without LGC (still using group convolutions) tends to improve performance towards baseline structure and achieves higher accuracy than structures without group convolutions and compression layers.

3.3.1. Varying squeezing ratio

Fig. 6 displays the result of changing squeezing ratios S_1. It is found that $S_1=0.1$ and $S_1=0.2$ have similar classification results, but further increasing the squeezing ratio deteriorates the accuracy. Results show that large squeezing ratio leads to wider bottleneck design, which needs more parameters to hit certain accuracy threshold. Low squeezing ratios implement the moderate bottleneck design, produce less feature output channels and also help to design deeper models. Better bottleneck design eventually helps to achieve higher accuracy as compared to other models with the same number

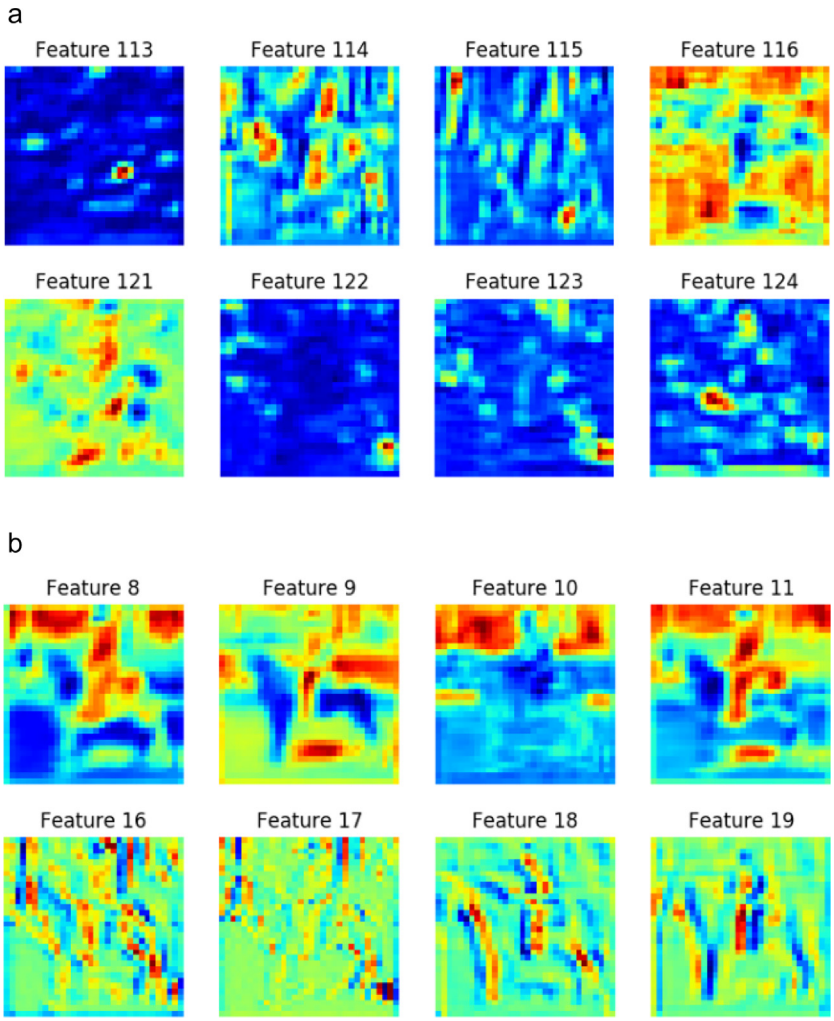

Fig. 10. Feature maps of the 1*st* CondenseNet dense block. (a) Some feature maps are in lock state; (b) some feature maps are similar intuitively.

of parameters and Flops. Squeezing ratio values less than 0.2 lead to very deep models without any major improvement in classification results. Thus, we select $S_1 = 0.2$ for our experiments.

3.3.2. *Shuffle layer and LGCs in compression /transition layer*

We also investigate the effect of shuffle layer and use of LGCs in compression and transition layer. The shuffle layer would be of no use when LGCs are used, because these themselves select the most suitable features from previous layers and learn related feature groups. Results in Fig. 7 approve our hypothesis that shuffle layer, instead of improving the results, slightly increases classification error. When output channel groups of 1 × 1 convolutional layers are not shuffled, 3 × 3 convolutional layers produce diverse feature outputs based on different group inputs; this slightly improves accuracy. Use of LGCs in all 1 × 1 convolutional layers leads to model over-fitting. CondenseNet used Group Lasso Regularization to mitigate this effect in ImageNet experiments. For large models, this effect becomes vibrant and decreases classification accuracy. Results in Fig. 7 show that large model with 3.1 M parameters has pronounced effect of over-fitting. We use 1 × 1 group convolutions (without LGCs) in compression/transition layers to curb this trend and to produce diverse features; thus enabling all 1 × 1 LGCs layers in the next RS block to select most feasible feature maps.

4. Discussion

Convolutional filters could extract features from images which will be more abstract in the deeper layers of the model. This part we are going to clarify our motivation of purposing the RSNet model. DenseNet improves the classification accuracy by reusing feature maps from previous layers, while CondenseNet adopts LGCs module to reduce the redundant parameters. At the same time, it also focuses more on the later layers. These two models have made significant improvement in the image classification tasks. Moreover, our team considers how to reduce parameters and improve computing

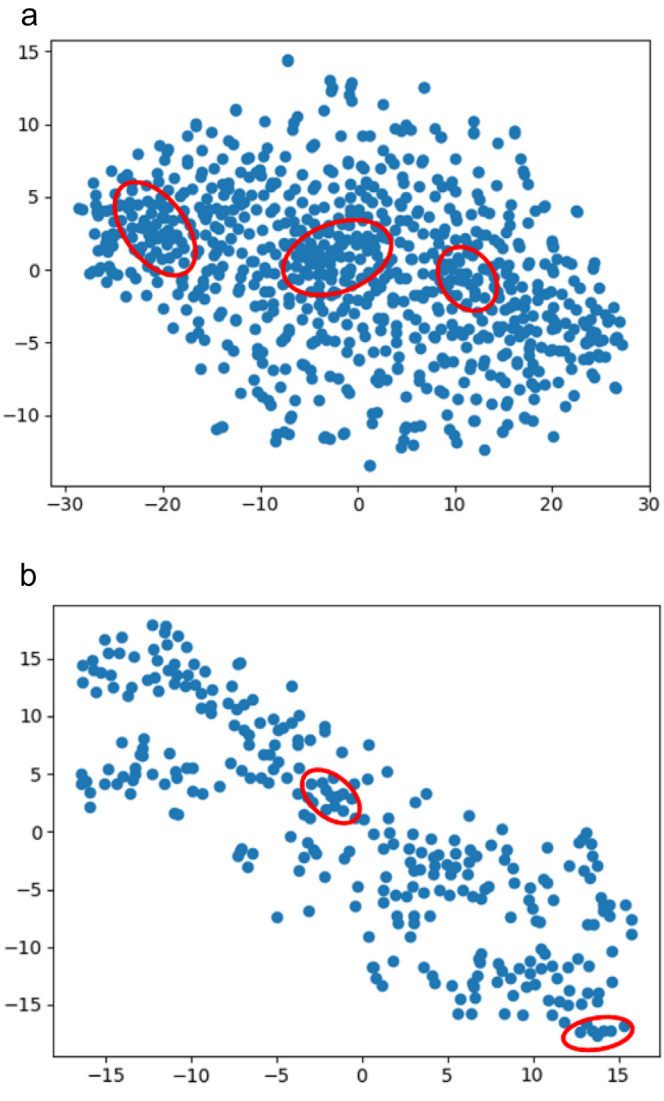

Fig. 11. Result of reducing the feature maps of CondenseNet as well as RSNet final dense block to 2 dimension by T-sne. And the points in red cycles show the similar feature maps, i.e. the redundant feature maps. (a) Reducing the feature maps of CondenseNet to 2 dimension; (b) reducing the feature maps of RSNet to 2 dimension.

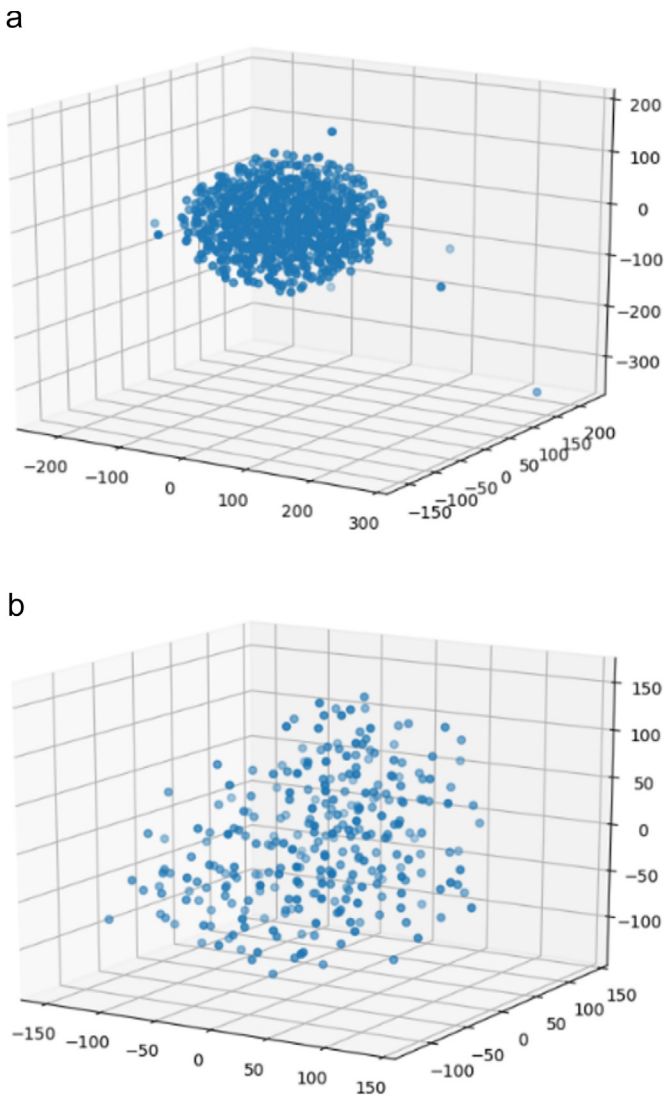

Fig. 12. Result of reducing the feature maps of CondenseNet as well as RSNet final dense block to 3 dimension, by T-sne. (a) Reducing the feature maps of CondenseNet to 3 dimension; (b) reducing the feature maps of RSNet to 3 dimension.

efficiency further. So we visualize the feature maps of DenseNet as well as CondenseNet separately, from 1st dense block to 3rd dense block. Besides, the feature maps by T-sne method have been visualized too.

4.1. Lock state feature maps

We find that some feature maps are not to be activated, i.e. they are in lock state, both in the shallow layers and even in the deep layers of CondenseNet. Fig. 9 (a) shows feature maps of the final dense block in CondenseNet. Our experiments have shown that the Relative-Squeezing bottleneck design not only slightly improves the model accuracy, but reduces model parameters greatly. Besides, when we visualize feature maps of 1*st* and 8*th* RS block, we find that there is almost no feature maps in lock state both in shallow and deep layer.

4.2. Redundant feature maps

Fig. 10 shows the feature maps from the first dense block of CondenseNet. Intuitively, we could clearly find that some feature maps have similar distribution pattern and we could also find feature maps in lock state here. Although these feature maps could not be considered useless, our work has shown that the fewer feature channels is enough. And it could be realized by RSNet.

Furthermore, beyond the intuitive feeling, we are going to find the similar distribution pattern of feature maps by T-sne.

For visualization, we reduce the feature maps of both CondenseNet and RSNet final block to 2 dimension (Fig. 11) and 3 dimension (Fig. 12), by T-sne. Although some information will be lost in the process of reduction, we could surmise that there are some redundant feature-maps among CondenseNet. Results of our experiments also support that the redundant could be compressed. However, Figs. 11 and 12 show that there are also some tiny redundant feature maps.

4.3. Channel-wise similarity measure

To study redundant feature maps in the model, the experiments on channel-wise feature maps similarity measurement are carried. To measure the similarity among feature maps, the Kullback-Leibler divergence (KL divergence, relative entropy) of different channel

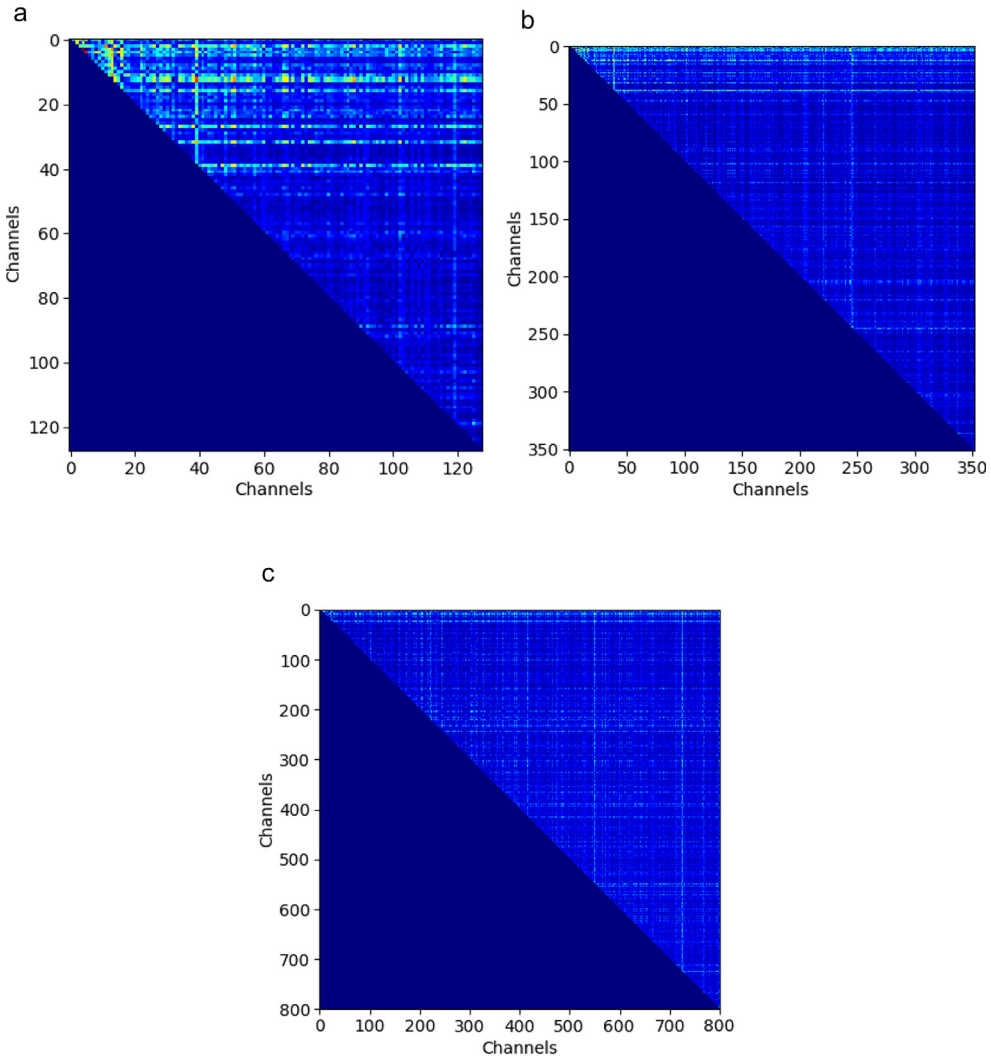

Fig. 13. KL-divergence heat matrix of output feature maps of each dense block in CondenseNet after training 1 epoch. (a) KL-divergence heat matrix of dense block 1; (b) KL-divergence heat matrix of dense block 2; (b) KL-divergence heat matrix of dense block 3.

pairs will be calculated. The model of CondenseNet and RSNet will be both trained 300 epochs (Figs. 13–15). After 300 epochs training, the top-1 error of both models will be less than 5%. And the KL divergence will be calculated on the output feature maps of each dense block, during the training process, in CondenseNet as well as RSNet. The results will be given as a heat matrix: bright color means the high difference between two feature maps, while the dark color means they are similar. Fig. 13 shows that: after training 1 epoch, the KL divergence among output feature maps of each dense block. We could find that the output feature maps of dense block 1 have more feature maps in different distribution, while the feature maps in deep layers are more similar.

After training 260 epochs, Fig. 14 shows that lots of feature maps share the low KL divergence value. In other words, the output feature maps of each dense block in CondenseNet are similar. This result is also in accordance with Sections 4.1 and 4.2, which show that some feature maps in model are redundant.

However, after training 260 epochs, the KL divergence of feature maps in RSNet is higher than that in CondenseNet, which means that RSNet has less redundant feature maps. And we could find that both in shallow and deep layers, feature maps in RSNet have the high KL divergence with each other. We could conclude that the feature maps in RSNet are more expressive. And the redundancy in CondenseNet is invalid redundancy, which could be compressed in the model. And RSNet just gives the solution to this problem (Fig. 15).

5. Conclusion

A new CNN structure is proposed RSNet, which integrates the new bottleneck layers, learned group convolutions, compression layers and transition layers. Our new bottleneck design—Relative Squeezing—emphasizes that moderate bottleneck design could achieve higher accuracy with substantially less parameters and computations. It reduces the output channels of bottleneck layer in the shallow layer, while increases the output channels of it in the deep layer. Multiple compression layers produce more expressive feature maps, which reduce computational and parameters overhead by discarding superfluous features. In the ablation study, we investigate the effect of different squeezing ratios, different structure components and existence of shuffle layer in the presence of LGCs. RSNet achieves state-of-the-art results with less parameters as well as less compute. Thus, the design of RSNet bottleneck layer would be a good option for the application of resource limited hardware.

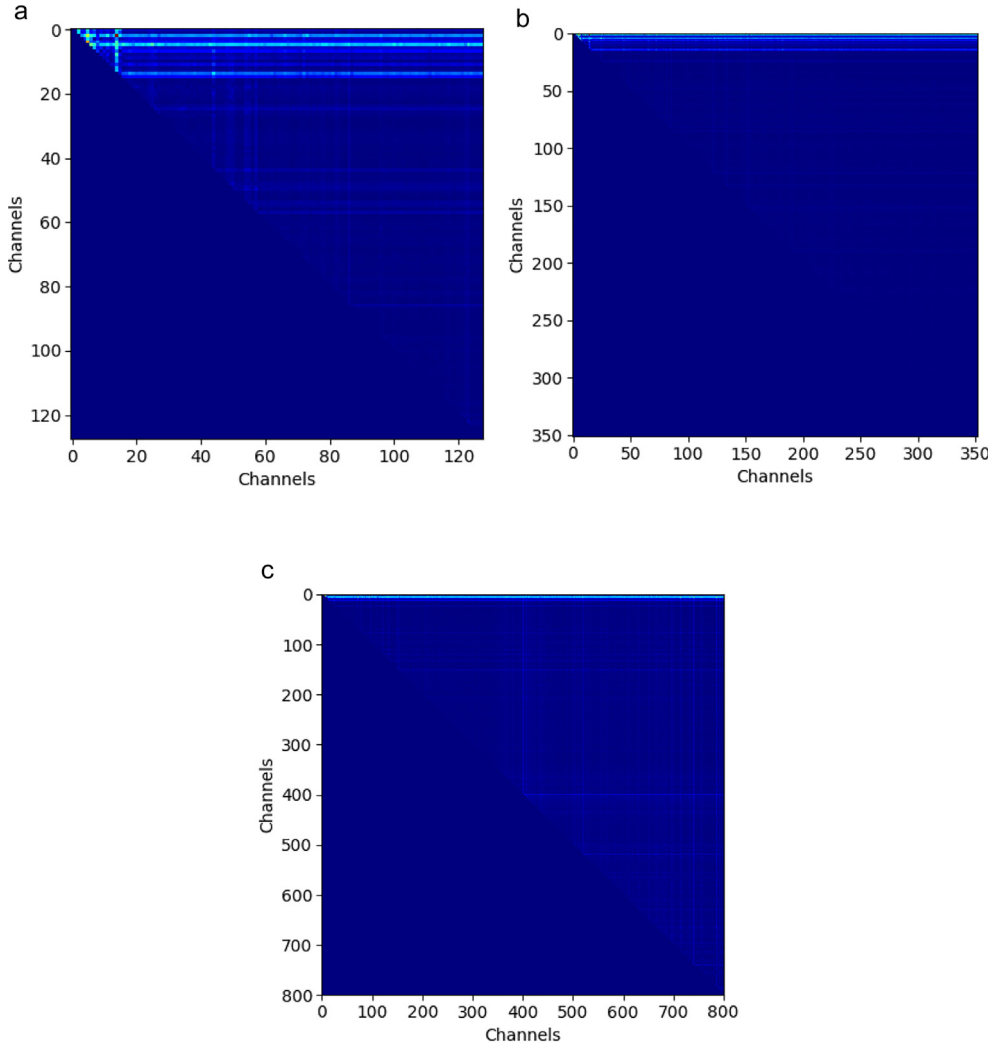

Fig. 14. KL-divergence heat matrix of output feature maps of each dense block in CondenseNet after training 260 epoch. (a) KL-divergence heat matrix of dense block 1; (b) KL-divergence heat matrix of dense block 2; (b) KL-divergence heat matrix of dense block 3.

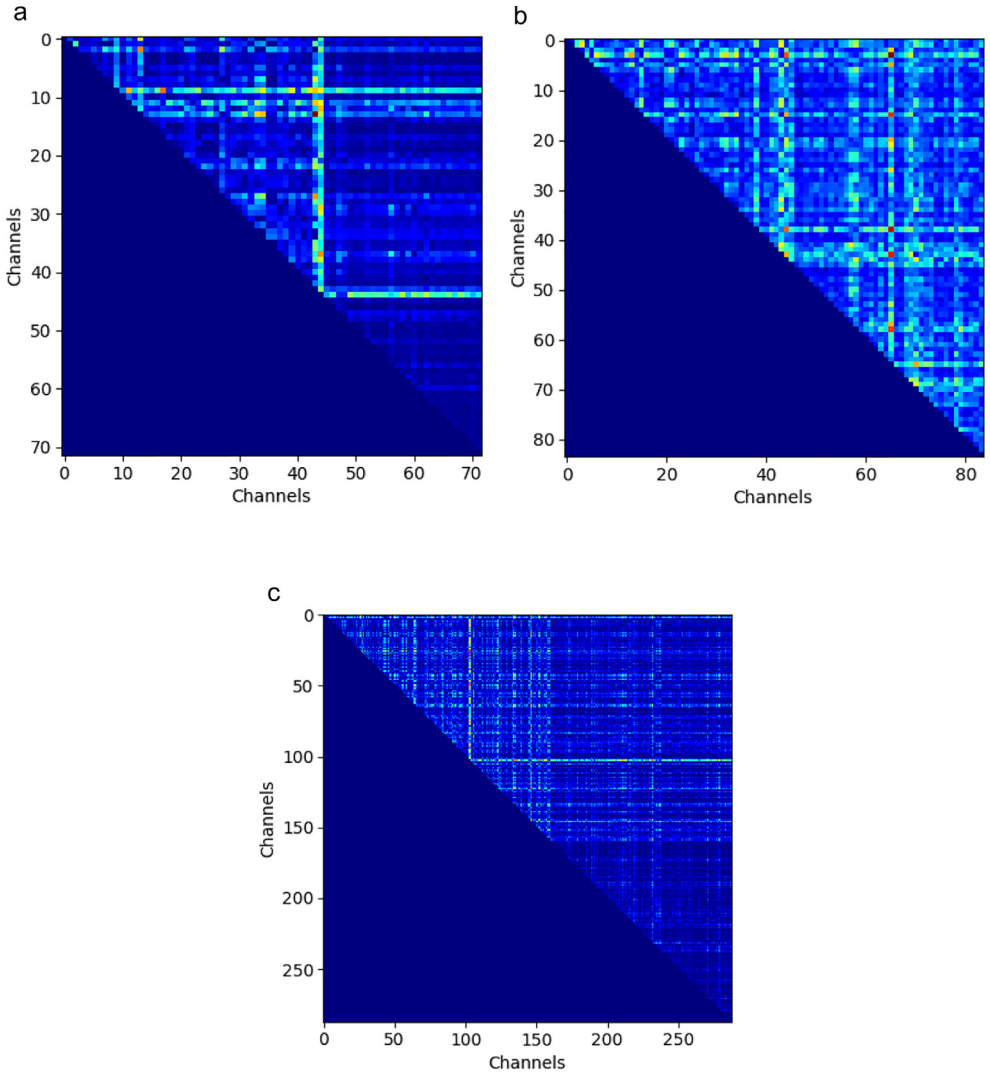

Fig. 15. KL-divergence heat matrix of output feature maps of each dense block in RSNet after training 260 epoch. (a) KL-divergence heat matrix of dense block 1; (b) KL-divergence heat matrix of dense block 5; (b) KL-divergence heat matrix of dense block 10.

Acknowledgments

This work is supported by the National Natural Science Foundation of China grant 61772052.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, 2012. pp. 1097–1105.
[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252. https://doi.org/10.1007/s11263-015-0816-y.
[3] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
[4] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-scale Image Recognition, arXiv preprint arXiv:1409.1556. 2014.
[5] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. pp. 2818–2826. https://doi.org/10.1109/CVPR.2016.308.
[7] H.T. Mustafa, J. Yang, M. Zareapoor, Multi-scale convolutional neural network for multi-focus image fusion, Image Vis. Comput. (2019) https://doi.org/10.1016/j.imavis.2019.03.001.
[8] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. pp. 4700–4708. https://doi.org/10.1109/CVPR.2017.243.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. pp. 1–9. https://doi.org/10.1109/CVPR.2015.7298594.
[10] A. Graves, Adaptive Computation Time for Recurrent Neural Networks, arXiv preprint arXiv:1603.08983. 2016.
[11] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Advances in Neural Information Processing Systems, 2015. pp. 1135–1143.
[12] Y. LeCun, J.S. Denker, S.A. Solla, Optimal brain damage, Advances in Neural Information Processing Systems, 1990. pp. 598–605.
[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning Filters for Efficient Convnets, arXiv preprint arXiv:1608.08710. 2016.
[14] B. Thomee, D.A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, L.-J. Li, YFCC100M: The New Data in Multimedia Research, arXiv preprint arXiv:1503.01817. 2015. https://doi.org/10.1145/2812802.
[15] C. Bucilu, R. Caruana, A. Niculescu-Mizil, Model compression, Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. 2006, pp. 535–541.
[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks, Advances in Neural Information Processing Systems, 2016. pp. 4107–4115.

[17] G. Larsson, M. Maire, G. Shakhnarovich, Fractalnet: Ultra-deep Neural Networks without Residuals, arXiv preprint arXiv:1605.07648. 2016.
[18] W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, International Conference on Machine Learning, 2015. pp. 2285–2294. https://doi.org/10.1145/2939672.2939839.
[19] F. Chollet, Xception: deep learning with depthwise separable convolutions, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. pp. 1251–1258. https://doi.org/10.1109/CVPR.2017.195.
[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee. 2009, pp. 248–255.
[21] M. Figurnov, M.D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, R. Salakhutdinov, Spatially adaptive computation time for residual networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. pp. 1039–1048. https://doi.org/10.1109/CVPR.2017.194.
[22] S. Han, H. Mao, W.J. Dally, Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv preprint arXiv:1510.00149. 2015.
[23] B. Hassibi, D.G. Stork, G.J. Wolff, Optimal brain surgeon and general network pruning, IEEE International Conference on Neural Networks, IEEE. 1993, pp. 293–299.
[24] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv preprint arXiv:1704.04861. 2017.
[25] G. Huang, S. Liu, L. Van der Maaten, K.Q. Weinberger, Condensenet: an efficient densenet using learned group convolutions, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. pp. 2752–2761. https://doi.org/10.1109/CVPR.2018.00291.
[26] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and $<$ 0.5 MB Model Size, arXiv preprint arXiv:1602.07360. 2016.
[27] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: an extremely efficient convolutional neural network for mobile devices, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. pp. 6848–6856. https://doi.org/10.1109/CVPR.2018.00716.
[28] N. Cohen, O. Sharir, A. Shashua, Deep SimNets, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. pp. 4782–4791. https://doi.org/10.1109/CVPR.2016.517.
[29] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, European Conference on Computer Vision, Springer. 2016, pp. 525–542. https://doi.org/10.1007/978-3-319-46493-0_32.
[30] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, ACM J. Emerg. Technol. Comput. Syst.(JETC) 13 (3) (2017) 32. https://doi.org/10.1145/3005348.
[31] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning Convolutional Neural Networks for Resource Efficient Inference, arXiv preprint arXiv:1611.06440. 2016.
[32] M. Lin, Q. Chen, S. Yan, Network in Network, arXiv preprint arXiv:1312.4400. 2013.
[33] T. Zhang, G.-J. Qi, B. Xiao, J. Wang, Interleaved group convolutions, Proceedings of the IEEE International Conference on Computer Vision, 2017. pp. 4373–4382. https://doi.org/10.1109/ICCV.2017.469.
[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. pp. 1492–1500. https://doi.org/10.1109/CVPR.2017.634.
[35] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, J. R. Stat. Soc. Ser. B (Stat Methodol.) 68 (1) (2006) 49–67.
[36] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Citeseer 2009, 10.1.1.222.9220.
[37] G. Huang, Y. Sun, Z. Liu, D. Sedra, K.Q. Weinberger, Deep networks with stochastic depth, European Conference on Computer Vision, Springer. 2016, pp. 646–661. https://doi.org/10.1007/978-3-319-46493-0_39.
[38] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: Hints for Thin Deep Nets, arXiv preprint arXiv:1412.6550. 2014.
[39] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for Simplicity: The All Convolutional Net, arXiv preprint arXiv:1412.6806. 2014.
[40] R.K. Srivastava, K. Greff, J. Schmidhuber, Highway Networks, arXiv preprint arXiv:1505.00387. 2015.
[41] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, European Conference on Computer Vision, Springer. 2016, pp. 630–645. https://doi.org/10.1007/978-3-319-46493-0_38.
[42] S. Zagoruyko, N. Komodakis, Wide Residual Networks, arXiv preprint arXiv:1605.07146. 2016. https://doi.org/10.5244/C.30.87.
[43] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. pp. 8697–8710. https://doi.org/10.1109/CVPR.2018.00907.
[44] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, Proceedings of the IEEE International Conference on Computer Vision, 2017. pp. 2736–2744. https://doi.org/10.1109/ICCV.2017.298.
[45] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, Proceedings of the IEEE International Conference on Computer Vision, 2017. pp. 1389–1397. https://doi.org/10.1109/ICCV.2017.155.