Multi-Scale Densely Connected Convolutional Neural Networks

Jinpeng Zhang[1,2], Jinming Zhang[3], Guyue Hu[1,2], and Shan Yu[1,2]

[1] Institute of Automation, Chinese Academy of Sciences (CAS),
95 Zhongguancun East Road, Beijing 100049, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
{zhangjinpeng2015,huguyue2016}@ia.ac.cn, shan.yu@nlpr.ia.ac.cn
[3] State Key Laboratory for Manufacturing Systems Engineering,
Xi'an Jiaotong University, Xi'an 710049, China
zhang.jin.ming@stu.xjtu.edu.cn

Abstract. Convolutional neural networks have got great progress in image classification in recent years. These networks usually have a common characteristic: several processing stages are connected in series; the plane size of feature-maps usually keep unchanged in the same stage, but will be reduced continuously stage by stage. This kind of idea basically governs the design of most classification networks, instead, we propose a brand new network architecture in this paper[†], called *Multil-scale DenseNet*. It is built by many densely stacked convolution-deconvolution coupled blocks and a densely forward-connected structure like DenseNet. Different from most mainstream classification networks, our networks can generate scale-variable feature-maps within each processing stage, and therefore can realize multi-scale feature extraction at any depth level of networks. Classification experiments on CIFAR10 and CIFAR100 show that our networks have good classification performance. The code will be available at https://github.com/alicococo.

Keywords: convolutional neural networks · DenseNet · multi-scale.

1 Introduction

In recent years, deep convolutional neural networks have been successfully applied to many computer vision tasks, such as image classification, object detection, semantic segmentation, face recognition, scene segmentation, super resolution reconstruction, image caption generation and so on. Image classification always plays a foundation role among them and the resulted networks can be used as feature extractors for these vision tasks.

The early classification networks, such as NiN [18], VGG [30], Inception [34–36], DSN [17], HighwayNet [32] and so on, all face the problem of vanishing/exploding gradients [1, 5]. ResNet [6] solves this problem to a large extend by adding skip connections between different layers. Based on ResNet, there

[†] A student paper.

appear many variants [3, 7, 12, 22, 25, 27, 28, 37, 39, 41, 42, 45] . DenseNet [11] gives an another solution for this problem, in which skip connections are densely constructed between a layer and its all previous layers. There are also many variants [8–10, 20, 40, 44] based on DenseNet.

As we all known, multi-scale analysis is a core issue in many computer vision tasks, such as image object detection, semantic segmentation, super resolution reconstruction and so on. The networks designed for these vision tasks often work on a classification backbone network, which is pre-trained on image classification dataset. However, in order to dock with the final linear classifier, classification networks are always designed to reduce the plane size of feature-maps continuously until a 1D feature vector is generated. In other words, the classification networks tend to promote feature channels at the expense of feature scales. So, an extra multi-scale enhancement module is often introduced to the classification backbone to generate new multi-scale feature-maps for the vision tasks that heavily depend on multi-scale features. For example, the notable FPN [19], RefineDet [46], DSSD [4], PFPNet [13], RefineDet [46], STDN [47] and FishNet [33] all introduce their different forms of multi-scale enhancement modules. This strategy is feasible but more like a remedial approach to accommodate the original classification backbone. Alternatively, another natural idea is to design an architecture natively containing enough multi-scale feature-maps.

Based on the above consideration, we propose a brand new network architecture in this paper, called *Multi-scale DenseNet*, briefly named as *Scale-DenseNet* or *ScaleNet*. Our architecture contains successively stacked convolution-deconvolution coupled blocks and a multi-way densely connected structure. Different with ResNet [6] and DenseNet [11], in which the plane size of feature-maps keeps the same from the start layer to the end layer within each stage, in our architecture, each stage can generate feature-maps with different plane sizes. In other words, our networks can produce multi-scale feature-maps at any depth level of the network.

In summary, we make the following contributions in this paper:

- We propose a novel architecture for classification which can generate scale variable feature-maps within its each processing stage.
- We propose a multi-way densely connected structure which further strengthens feature forward-propagation and gradient back-propagation.
- It is the first time for deep convolutional neural networks to realize multi-scale feature extraction at any depth level of networks.
- It is the first public work to apply densely stacked convolution-deconvolution coupled blocks in deep convolutional neural networks for image classification.

2 Related Work

Image classification is a very classical vision task, and since the significant progress of AlexNet [15], the researches on this area begin to blowout. The NiN [18], VGG [30], Inception [34–36], DSN [17], HighwayNet [32] are the typical ones in the early years. All of them try to make networks deeper and deeper

to get a strong learning ability, but are impeded by the problem of vanishing/exploding gradients [1, 5].

ResNet [6] solves this problem to a large extent by adding shortcut connections between different layers in each residual block. Each connection performs a summing operation for two layers. Based on ResNet, there appear many other network architectures [3, 7, 12, 22, 25, 27, 28, 37, 39, 41, 42, 45]. DenseNet [11] solves this problem by connecting each layer to every other layer in a feed-forward fashion, which can further alleviate the vanishing-gradient problem. Each connection in DenseNet performs a concatenation operation for two layers. Many other networks based on DenseNet are proposed. Log-DenseNet [8] increases the back-propagation distances among layers from 1 to $(1 + \log_2 L)$, and uses $L \log_2 L$ total connections instead of $O(L^2)$. MSDNet [9] generates and maintains coarse level features throughout the network and inter-connects them with dense connectivity. CondenseNet [10] combines dense connectivity with learned group convolution to improve the classification efficiency. SparseNet [20] reduces connections of a L-layer DenseNet from $O(L^2)$ to $O(L)$, and simultaneously increase depth, width and connections of neural networks in a more efficient way. MCA-DenseNet [40] proposes an adaptive multi-scale convolution aggregation module for DenseNet. MFR-DenseNet [44] improves the representation power of the DenseNet by adaptively recalibrating the channel-wise feature responses and explicitly modeling the interdependencies between the features of different convolutional layers.

Deconvolution, also called transpose convolution, has been carefully discussed in [2, 24, 29]. It can expand the plane size of an input tensor, and therefore often appears in the vision tasks that need planar reconstruction, such as object detection, segmentation, super-resolution reconstruction and so on. In these tasks, deconvolution layers are often attached to the tail of a classification backbone to restore the plane size of feature-maps. DeconvNet [23], DSSD [4] are the representatives of this method. On the contrary, deconvolution hardly appears in classification task. Because classification networks are expected to output a 1D feature vector to dock with a linear classifier, they trend to decrease the plane size of feature-maps continuously instead of increasing it.

As far as we known, this is the first public work to apply densely stacked convolution-deconvolution coupled blocks to classification task. Based on this design, our networks realize the multi-scale feature extraction at any depth level of the whole network for the first time.

3 Our Method

In this section, we will introduce our network architecture and evaluate it on CIFAR10 and CIFAR100 [14]. In order to facilitate the writing, we use the following shorthands: convolution is abbreviated as conv or Conv, deconvolution is abbreviated as deconv or DeConv, batch normalization is abbreviated as BatchNorm. For example, a convolution unit will be abbreviated as conv-unit or Conv-unit, and a deconvolution unit will be abbreviated as deconv-unit or DeConv-unit.

3.1 Network Architecture

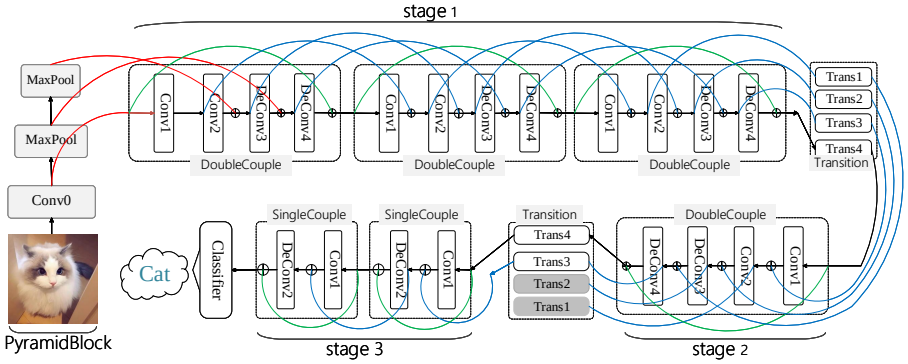

Fig. 1. An architecture demo of ScaleNet. ⊕:concatenation operation of two tensors.

Conv-Deconv Coupled Block: Fig.1 shows an architecture demo of our networks. The demo contains three successive processing stages (stage1, stage2 and stage3), each stage contains several stacked isomorphic blocks (surrounded by dotted boxes), and each block is a conv-deconv couple. There are two kinds of conv-deconv couples, *Conv-Deconv* couple and *Conv-Conv-Deconv-Deconv* couple, which are named as *SingleCouple* and *DoubleCouple* respectively. SingleCouple is constructed by two stacked conv-unit and deconv-unit. DoubleCouple is constructed by two SingleCouple blocks, in which one is embedded into another. All of them have been shown in Fig.1. These two kinds of couples can be independently used to build a network, and can also be mixed to build a network. Fig.1 shows a mixed usage.

Hyper Parameters: The hyper parameters of conv-deconv couples are as follows. All conv-units, including Conv1 and Conv2 in Fig.1, have the same parameters: kernel size of 3, stride of 2, padding of 1, dilation of 1, and no bias. All deconv-units, including DeConv2, DeConv3 and DeConv4 in Fig.1, have the same parameters: kernel size of 3, stride of 2, padding of 1, dilation of 1, output padding of 1 and no bias. For each Conv unit and DeConv unit, a BatchNorm unit and a ReLU unit will be attached to them. The order of Conv (or DeConv), BatchNorm and ReLU is the same as pre-active-ResNet [7], which is BatchNorm-ReLU-Conv (or BatchNorm-ReLU-DeConv), not the traditional Conv-BatchNorm-ReLU used in ResNet [6]. In the following sections, these hyper parameters will be used in all our networks on CIFAR10 and CIFAR100.

Feature Dimensions: In a conv-deconv couple, input feature-maps will be firstly downsampled by one or two (one for SingleCouple, two for DoubleCouple, the same below) conv-units, and then upsampled by one or two deconv-units. Because conv-unit and deconv-unit are always coupled into a pair and have the same stride of 2, the output plane size of a block can always keep the same with its input plane size. As a result, a processing stage, constructed by isomorphic conv-deconv blocks, can also assure its output plane size be same with its input plane size. The output feature channels of each conv-unit and deconv-unit in conv-deconv blocks are called as **growth-rate** (denoted as K), a concept borrowed from DenseNet [11] to describe the growing rules of feature-channels. **The grow-rates (K) of all conv-deconv blocks in a network are the same**. For each conv-unit and deconv-unit in conv-deconv blocks, its final output is the concatenation of its own output and the outputs of all previous blocks. The concatenation operations have been shown by skip connections (green lines and blue lines) in Fig.1. In addition, the first conv-deconv block in the first stage has some difference with other conv-deconv blocks. Because this block is in the starting position and no concatenation follows its first conv-unit, so the feature channels of this first conv-unit will be manually specified as $C + K$. C is the output feature channels of PyramidBlock. Therefore, if a processing stage contains N DoubleCouple blocks and has an input feature channels of C, the final output feature channels of this stage will be $C + N \times K$.

Naming Rules: In the following sections, if a stage adopts DoubleCouple blocks, it will be denoted with a postfix '-D'; while if SingleCouple blocks are adopted, the stage will be denoted with a postfix '-S'. For example, a ScaleNet with 3 stages and 40 layers can be denoted as ScaleNet40-$D_5D_3S_2$-$C_{24}K_{12}$, which means that its first stage contains 5 DoubleCouple blocks (-D_5), the second stage contains 3 DoubleCouple blocks (D_3), and the third stage contains 2 SingleCouple blocks (S_2). C_{24} refers that the input feature channels of its first stage is 24. K_{12} refers that the growth-rate of all conv-deconv blocks is 12. The depth 40 means that the total number of convolution layers, deconvolution layers and fully connected layers in this network is 40.

Transition Blocks: Several transition blocks are placed between every two adjacent stages to halve the feature plane size and reduce the feature channels from one stage to the next stage. Fig.1 shows two transition blocks between stage1 and stage2, stage2 and stage3. In DenseNet [11], skip connections only exist within a stage, and its transition blocks are placed between every two adjacent stages to reduce the channels and plain size of feature-maps. Differently, the skip connections in our architecture not only exist within a stage but also across different stages. So, there are four (or two) transition units in our transition blocks, and they process their parallel inputs independently. This has been shown by 'Trans1', 'Trans2', 'Trans3', and 'Trans4' in Fig.1. For the transition block between two adjacent DoubleCouple blocks, it has four transition units; while for the transition block between two adjacent DoubleCouple block and

SingleCouple block, it has two transition units. Each transition unit contains a conv-unit(kernel size of 1, stride of 1, padding of 1, and no bias) and a average pooling unit (kernel size of 2, stride of 2 and padding of 0).

Multi-way Skip Connection: There are two kinds of skip connections: summing mode proposed in ResNet [6] and concatenating mode proposed in DenseNet [11]. We use concatenating mode. Fig.1 shows the overall view of skip connections in our architecture, where each circle embedded with '+' represents a concatenation operation node.

We borrow the idea of **growth-rate**(denoted as K) from DenseNet [11] and let each conv/deconv unit in conv-deconv blocks produces K-channels feature-maps. Just like DenseNet, the growth rate K regulates how much new information each conv/deconv unit contributes to the network global state. The value of K doesn't affect the network depth, but affect the network width and thus can change the amount of network learnable parameters.

Each conv/deconv unit in conv-deconv blocks is followed by a concatenation operation node. The feature-maps after each concatenation node includes two parts: one comes from conv/deconv unit, the other comes from skip connection. Because of adopting the concatenating mode, as more and more conv-deconv blocks are stacked in the same stage, the amount of the output channels of the stage will increase rapidly, which is not conducive to building deeper networks. Therefore, the transition block placed after each stage will play a role to reduce the channels and plain size of feature-maps. If a stage outputs m feature-maps, the following transition block will output $\theta \cdot m$ feature-maps, where θ refers to the compression factor and $0 \leq \theta \leq 1$. When $\theta = 1$, the number of feature-maps across transition block remains unchanged. Following DenseNet [11], we set $\theta = 0.5$ in all our networks.

FeaturePyramid Block: A preprocessing block, named 'PyramidBlock' in Fig.1, is designed to extract multi-scale feature-maps from raw images, whose output plane sizes are 1/1, 1/2, 1/4 of the input image size. This PyramidBlock can expediently dock with the following conv-deconv block, and thus let the whole network works on feature pyramids. The concrete form of PyramidBlock needs to be customized according to the dataset used. Fig.1 shows a PyramidBlock used in CIFAR10 and CIFAR100, which contains three successive processing nodes. The first node contains a convolution layer with kernel size of 3, stride of 1, padding of 1 and no bias; the second node contains a max-pooling layer with kernel size of 2, stride of 2, padding of 0; the third node also contains a max-pooling layer with kernel size of 2, stride of 2, padding of 0.

4 Experiments and Results

In this section we will make experiments on CIFAR10 and CIFAR100 [14], and report the results on their validation sets.

4.1 Datasets and Implementation Details

CIFAR10 and CIFAR100 all contain 50k training images and 10k test images with size of $3 \times 32 \times 32$. CIFAR10 has 10 categories, while CIFAR100 has 100 categories. Following [6, 7, 11, 17], the original training set of both CIFAR10 and CIFAR100 will be split to 45k for training and 5k for validation, and the validation will be carried out at the end of each training epoch. The final test error rate will be reported on their 10k test set. A standard data augmentation scheme widely used in [6,7,11,17] is adopted for both CIFAR10 and CIFAR100: 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For testing on CIFAR10 and CIFAR100, we only evaluate the single view of the original 32×32 image. Both CIFA10 and CIFA100 are trained on 2x NVIDIA TitanX GPUs with a batch size of 128. The optimizer is stochastic gradient descent (SGD) with a learning rate starting form 0.1 and divided by 10 at 180th epoch, 210th epoch respectively during a total of 240 training epochs. The weight decay and momentum are $0.0001 \sim 0.0005$ and 0.9 respectively.

4.2 Results on CIFAR10 and CIFAR100

Several ScaleNets with different parameter scales are designed for CIFAR10 and CIFAR100. The experiment results are shown in Table.1. Some state-of-art results reported by other papers, especially ResNet, DenseNet and their variants, are also shown in Table.1. All our best results are marked in boldface and the overall best results are marked in blue.

Accuracy: From Table.1, we can see that our ScaleNet112 with 5.0M parameters get a error rate of 7.73% on CIFAR10, which exceeds ResNet1202(19.4M, 7.93%) with a parameter reduction of 74%. It also exceeds Network in Network(8.81%), Deeply Supervised Net(7.97%) and equal to HighwayNet(7.72%). Meanwhile, it gets a error rate of 29.67% on CIFA100, which exceeds Deeply Supervised Net (34.57%), HighwayNet (32.39%) and ALL-CNN (33.71%). In general, the results in Table.1 clearly show that our ScaleNets have better performance than the early classical networks without skip connections. But compared to ResNet, DenseNet and their variants, our ScaleNets perform bad on both CIFAR10 and CIFAR100.

In addition, under the parameter scales of 3.1M, 5.0M and 10.0M, we use the same combination of conv-deconv blocks (10 DoubleCouple blocks in the first stage, 10 DoubleCouple blocks in the second stage and 10 SingleCouple blocks in the last stage), but use different input channels(denoted as C) and different growth-rate(denoted as K). We can see that with the widening of networks, ScaleNets get a better and better performance on both CIFAR10 and CIFAR100.

Capacity: Network capacity is usually limited by two problems: under-fitting on training data and over-fitting on test data. With the increase of parameter scale,

Table 1. the classification performance of ScaleNets on CIFAR10 and CIFAR100.

Model	Depth	Parameters(M)	Cifar10+	Cifar100+
NiN [18]	-	-	8.81	-
ALL-CNN [31]	-	-	7.25	33.71
Deeply Supervised Net [17]	-	-	7.97	34.57
HighwayNet [32]	-	-	7.72	32.39
FractalNet [16]	21	38.6	5.22	23.30
With dropout/drop-path	21	38.6	4.60	23.73
ResNet [6]	110	1.7	6.61	-
Reported by [12]	110	1.7	6.41	27.22
	1202	19.4	7.93	-
ResNet-	110	1.7	5.23	24.58
Stochastic-Depth [12]	1202	10.2	4.91	-
ELU-ResNet [26]	110	1.7	5.62	26.55
PELU-ResNet [38]	110	1.7	5.37	25.04
Wide ResNet [43]	16	11.0	4.81	22.07
	28	36.5	4.17	20.50
pre-active-ResNet [7]	164	1.7	5.46	24.33
	1001	10.2	4.62	22.71
Attention-ResNet [39]	92	1.9	4.99	21.71
	236	5.1	4.14	21.16
	452	8.6	3.90	20.45
FitResNet [21]	-	2.5M	5.84	27.66
ResNet-in-ResNet [37]	18	10.3	5.01	22.90
CRMN [22]	28	40	4.65	20.35
ResNet-of-ResNet [45]	164	2.5	4.51	21.94
	WRN40	8.9	4.09	20.11
	WRN58	13.3	3.77	19.73
	1202	19.4	4.49	20.64
ResNeXt [41]	29	68.1	3.58	17.31
DenseNet [11]	250	15.3	3.62	17.60
	190	25.6	3.46	17.18
CondenseNet [10]	182	4.2M	3.76	18.47
MCA-DenseNet [40]	40	11.6M	5.76	22.65
MFR-DenseNet [44]	100	14.2M	3.57	18.27
SparseNet-V4 [20]	184	35.0M	3.24	16.98
ScaleNet-D_{17}-$C_{24}K_{12}$	73	1.0	8.93	32.13
ScaleNet-$D_{13}D_{13}$-$C_{24}K_{12}$	113	1.7	8.52	30.48
ScaleNet-$D_{10}D_{10}S_{10}$-$C_{24}K_{18}$	112	3.1	8.31	30.10
ScaleNet-$D_{10}D_{10}S_{10}$-$C_{24}K_{23}$	112	5.0	7.73	29.67
ScaleNet-$D_{10}D_{10}S_{10}$-$C_{36}K_{33}$	112	10.1	**7.57**	**28.12**

the training optimization of networks becomes more and more difficult (under-fitting); or test accuracy doesn't increase with the increase of training accuracy, but begins to decrease (over-fitting). From Table.1, we can see that with the increase of parameter scale, our ScaleNets get a lower and lower error rate on both CIFAR10 and CIFAR100, which suggests ScaleNets can fully utilize the increased parameters to gain stronger representational ability. Comparing the input channels (C) and growth-rate (K) used in different ScaleNets, we can see that the network accuracy can gradually increase with the increase of network width. This demonstrates that our ScaleNets have a good capacity performance on width.

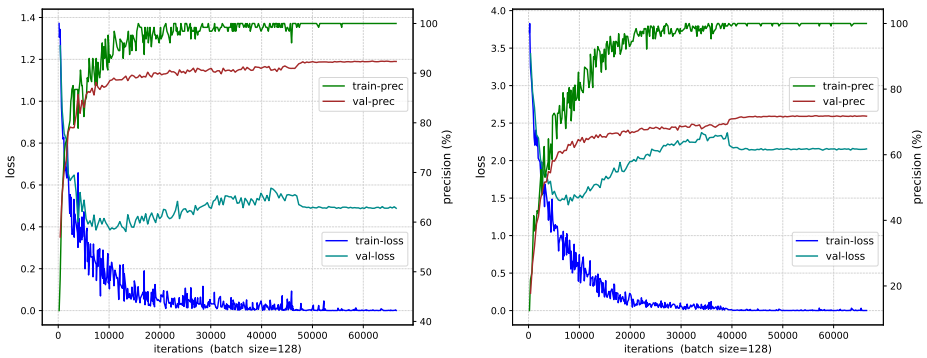

Fig. 2. ScaleNet112-$D_{10}D_{10}S_{10}$-$C_{36}K_{33}$ with 10.0M parameters on CIFAR10(*left*) and CIFAR100(*right*): training loss curve (denoted as *train-loss*), validation loss curve (denoted as *val-loss*), training precision curve (denoted as *train-prec*) and validation precision curve (denoted as *val-prec*). The validation is carried out on their 10k test images. The precision is Top-1 precision.

Overfitting: In experiments, we observe serious over-fitting phenomena on both CIFAR10 and CIFAR100. Fig.2 shows the training curves and validation curves of our ScaleNet112(10.0M) on these two datasets. From Fig.2*(left)*, we can see that with the increase of training accuracy, the val-loss on CIFAR10 validation set has a rapid reduction at first, but begin to increase after 15k iterations, and then remains stable after the learning reate is reduced by 1/10. While, the val-precision increase continuously during all training process and remains stable after the learning rate is decayed by 1/10. From Fig.2*(right)* we can see that the over-fitting also exists on CIFAR100. The val-loss has a dramatical reverse after 10k iterations. But the val precision always keeps rising in the whole training process.

Considering that the densely connected structure has a strong ability of feature propagation just like what DenseNet [11] has demonstrated, our multi-way densely connected structure actually strengthen the feature propagation again.

Meanwhile, the deconvolution units also have the great ability to reconstruct the feature-maps, and therefore strengthen the feature propagation once more. So, in our ScaleNets , image details will be much easier to extract and transmit to the deep layers of networks. Therefore, the feature space learned from training images will be too fine to prevent over-fitting on test images. This cab be demonstrated by the over-fitting comparison between CIFAR10 and CIFAR100. Compared to CIFAR10, CIFAR100 is a small dataset, which have 100 classes and 500 training images for each class. So the image details contained in CIFAR100 training set can be extracted and retained more abundantly and easily.

In general, the powerful fitting ability is a double-edged sword, which means networks have sufficient learning ability on big dataset and under-fitting will not happen easily; but also means networks have excessive learning ability on small dataset and over-fitting will happen easily. We will continue to improve the network architecture in our future work to suppress its over-fitting, so as to achieve better classification performance.

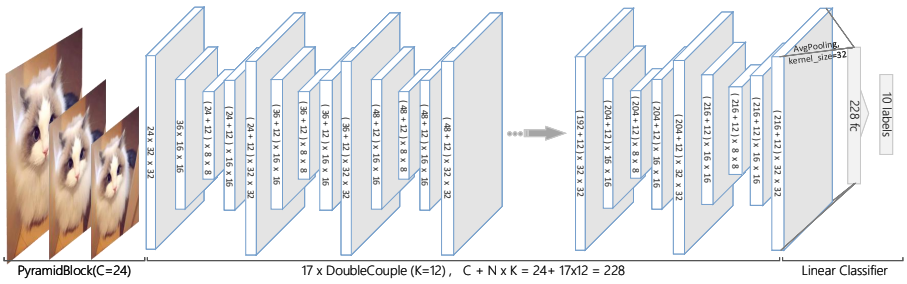

Fig. 3. The multi-scale feature-maps generated by our ScaleNet73-D_{17} on CIFAR10.

Multi-Scale Features: In order to show the multi-scale features of our architecture clearly, a ScaleNet73-D_{17}-$C_{24}K_{12}$ with 1.0M parameters are shown in Fig.3. This network has only one processing stage and 17 DoubleCouple blocks, so the feature-maps generated by it will be a repeat of $[32\times32, 16\times16, 8\times8, 16\times16, 32\times32]$. This is completely different form the most mainstream classification networks, which demonstrates that our ScaleNets have the ability to extract multi-scale feature-maps at any depth level of networks. When considering that multi-scale feature-maps are highly necessary in many computer vision tasks, the distinctive multi-scale feature-maps of our ScaleNet can provide much more imaginary space.

5 Conclusion

In this paper, we propose a novel network architecture for image classification, which contains densely stacked conv-deconv couples and a multi-way densely

connected structure. As far as we known, this is the first public work to introduce a large amount of deconvolution units for image classification. The experiments on CIFAR10 and CIFAR100 have good classification performance, which demonstrates its effectiveness. The most exciting advantage of our architecture is that it realize multi-scale analysis on any depth level of networks for the first time. As a result, it can provide much richer multi-scale feature-maps for other vision tasks at any semantic depth of networks.

References

1. Bengio, Y., Simard, P.Y., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5 2**, 157–66 (1994)
2. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285 (2016)
3. Figurnov, M., Collins, M.D., Zhu, Y., Lin, X., Huang, J., Vetrov, D.P., Salakhutdinov, R.R.: Spatially adaptive computation time for residual networks. In: CVPR (2017)
4. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd : Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
5. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
7. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV (2016)
8. Hu, H., Dey, D., Giorno, A.D., Hebert, M., Bagnell, J.A.: Log-densenet: How to sparsify a densenet. arXiv preprint arXiv:1711.00002 (2018)
9. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. In: ICLR (2017)
10. Huang, G., Liu, S., van der Maaten, L., Weinberger, K.Q.: Condensenet: An efficient densenet using learned group convolutions. In: CVPR (2018)
11. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017)
12. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV (2016)
13. Kim, S.W., Kook, H.K., Sun, J.Y., Kang, M.C., Ko, S.J.: Parallel feature pyramid network for object detection. In: ECCV (2018)
14. Krizhevsky, A.: Learning multiple layers of features from tiny images. In: Tech Report (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**, 84 (2012)
16. Larsson, G., Maire, M., Shakhnarovich, G.: Fractalnet: Ultra-deep neural networks without residuals. In: ICLR (2017)
17. Lee, C.Y., Xie, S., Gallagher, P.W., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)
18. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2014)
19. Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR (2017)

20. Liu, W., Zeng, K.: Sparsenet: A sparse densenet for image classification. arXiv preprint arXiv:1804.05340 (2018)
21. Mishkin, D., Matas, J.: All you need is a good init. arXiv preprint arXiv:1511.06422 (2015)
22. Moniz, J., Pal, C.J.: Convolutional residual memory networks. arXiv preprint arXiv:1606.05262 (2016)
23. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV (2015)
24. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016). https://doi.org/10.23915/distill.00003, http://distill.pub/2016/deconv-checkerboard
25. Sandler, M.B., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
26. Shah, A., Kadam, E., Shah, H., Shinde, S.: Deep residual networks with exponential linear unit. arXiv preprint arXiv:1604.04112 (2016)
27. Shen, F., Gan, R., Zeng, G.: Weighted residuals for very deep networks. In: ICSAI (2016)
28. Shen, Y., Gao, J.: Refine or represent: Residual networks with explicit channel-wise configuration. In: IJCAI (2018)
29. Shi, W., Caballero, J., Theis, L., Huszar, F., Aitken, A.P., Ledig, C., Wang, Z.: Is the deconvolution layer the same as a convolutional layer? arXiv preprint arXiv:1609.07009 (2016)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
31. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. ICLR (2014)
32. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
33. Sun, S., Pang, J., Shi, J., Yi, S., Ouyang, W.: Fishnet: A versatile backbone for image, region, and pixel level prediction. In: NeurIPS (2018)
34. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI (2016)
35. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
37. Targ, S., Almeida, D., Lyman, K.: Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029 (2016)
38. Trottier, L., Giguère, P., Chaib-draa, B.: Parametric exponential linear unit for deep convolutional neural networks. In: ICMLA (2017)
39. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C.C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: CVPR (2017)
40. Wang, M., Zhou, J., Mao, W., Gong, M.: Multi-scale convolution aggregation and stochastic feature reuse for densenets. In: WACV (2019)
41. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
42. Yu, F., Koltun, V., Funkhouser, T.A.: Dilated residual networks. In: CVPR (2017)
43. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
44. Zhang, K., Guo, Y., Wang, X., Yuan, J., Ding, Q.: Multiple feature reweight densenet for image classification. IEEE Access **7**, 9872–9880 (2019)

45. Zhang, K., Sun, M., Han, T.X., Yuan, X., Guo, L., Liu, T.: Residual networks of residual networks: Multilevel residual networks. IEEE Transactions on Circuits and Systems for Video Technology **28**, 1303–1314 (2018)
46. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: CVPR (2018)
47. Zhou, P., Ni, B., Geng, C., Hu, J., Xu, Y.: Scale-transferrable object detection. In: CVPR (2018)