IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 12, 2019, accepted October 6, 2019. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.2946425

Scalenet: A Convolutional Network to Extract Multi-Scale and Fine-Grained Visual Features

JINPENG ZHANG[1,2], JINMING ZHANG[3], GUYUE HU[1,2], YANG CHEN[1], AND SHAN YU[1,2,4]

[1]Brainnetome Center and National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[4]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Beijing 100190, China

Corresponding author: Jinpeng Zhang (zhjphust@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0105203, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences (CAS) under Grant XDBS01040200. The work of S. Yu was supported in part by the Hundred-Talent Program of CAS.

ABSTRACT Many convolutional neural networks have been proposed for image classification in recent years. Most tend to decrease the plane size of feature maps stage-by-stage, such that the feature maps generated within each stage show the same plane size. This concept governs the design of most classification networks. However, it can also lead to semantic deficiency of high-resolution feature maps as they are always placed in the shallow layers of a network. Here, we propose a novel network architecture, named ScaleNet, which consists of stacked convolution-deconvolution blocks and a multipath residual structure. Unlike most current networks, ScaleNet extracts image features by a cascaded deconstruction-reconstruction process. It can generate scale-variable feature maps within each block and stage, thereby realizing multiscale feature extraction at any depth of the network. Based on the CIFAR-10, CIFAR-100, and ImageNet datasets, ScaleNet demonstrated competitive classification performance compared to state-of-the-art ResNet. In addition, ScaleNet exhibited a powerful ability to capture strong semantic and fine-grained features on its high-resolution feature maps. The code is available at https://github.com/zhjpqq/scalenet.

INDEX TERMS Image classification, convolutional neural networks, ResNet, deconvolution.

I. INTRODUCTION

With the development of deep convolutional neural networks, image classification has achieved considerable progress in recent years. The early convolutional neural networks (CNNs) used for classification, such as NiN [1], VGG [2], Inception [3], [4], DSN [5], and HighwayNet [6], faced issues of vanishing/exploding gradients [7], [8]. Fortunately, with the development of ResNet [9] and DenseNet [10], these problems have been largely alleviated. ResNet [9] solved this problem by adding summation-skip-connections between different layers in each block, leading to the extension of many ResNet variants [11]–[24]. DenseNet [10] resolved the issue by constructing concatenation-skip-connections between a layer and all previous layers in a block, with various models developed subsequently [25]–[30]. Instead of the manual design method used in above networks, network architecture search (NAS) finds the optimal architecture by auto searching several key hyper-parameters of a baseline network, as seen in [23], [31]–[39].

The associate editor coordinating the review of this manuscript and approving it for publication was Guitao Cao.

However, to extract high-level semantic information, most classification networks tend to decrease feature size (*i.e.*, plane size of feature maps, same below) and increase feature channels stage-by-stage until a 1D feature vector is generated. Thus, they sacrifice feature size to promote feature channels, resulting in feature dimension reduction. As a consequence, high-/low-resolution feature maps always occupy the shallow/deep layers of a network, respectively. On the other hand, the semantic information contained in feature maps usually increases in strength with network depth [40]. Therefore, high-resolution feature maps will always have weak semantic representation, whereas low-resolution feature maps will always have strong semantic representation. In other words, there exists an intrinsic trade-off between feature size and feature semantics for these CNNs. As such, the construction of a network that can eliminate this kind of compromise is important.

Inspired by the above consideration, we propose a novel network architecture, named ScaleNet. ScaleNet contains stacked convolution-deconvolution blocks and a multipath residual structure. Different from ResNet [9] and DenseNet [10], in which feature size remains unchanged within each stage, ScaleNet can generate scale-variable

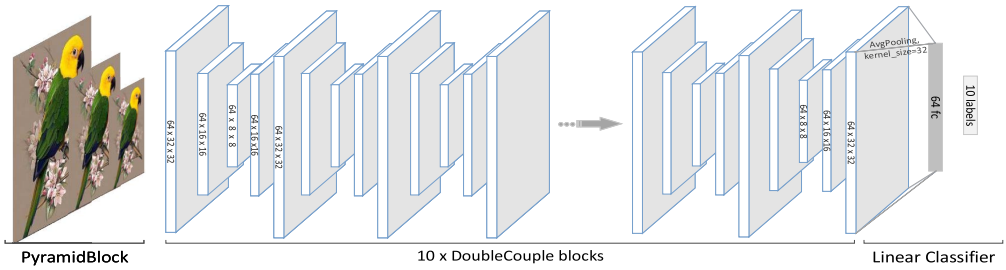


FIGURE 1. Feature-maps generated by ScaleNet44-D_{10}(2.0M) on CIFAR-10.

feature maps within each stage and block, as shown in Fig.1. Based on this, ScaleNet can generate high-resolution feature maps in very deep layers, thus contributing to the ability to capture fine-grained visual features with strong semantic information. In addition, compared with the original residual structure used in ResNet [9], [12], our multipath residual structure has a stronger ability for both feature forward-propagation and gradient back-propagation, and can therefore improve parameter efficiency.

In summary, this paper makes the following contributions:

- Our network can generate scale-variable feature maps within each block and stage and can therefore realize multiscale feature extraction at any depth level of the network.
- Our network can learn fine-grained features with strong semantic representation for high-resolution feature maps.
- Our network has a multipath residual structure, which can further improve feature forward-propagation ability and gradient back-propagation ability of CNNs.

II. RELATED WORKS

Based on the significant progress of AlexNet [41], research on CNNs has shown considerable advancement. However, early networks [1]–[6] were impeded by vanishing/exploding gradient [7], [8]. ResNet [9] relieved this problem to a large extent by adding skip-connections between different layers in each residual block, with the subsequent emergence of many ResNet variants. For example, pre-active-ResNet [12] uses identity mappings and adjusts the order of convolution, batch normalization, and ReLU to make training easier. WResNet [17] proposes weighted residuals to create a direct path. ELU-ResNet [42] and PELU-ResNet [43] propose the use of Exponential Linear Unit (ELU) and Parametric Exponential Linear Unit (PELU), respectively. Stochastic-Depth-ResNet [11] recommends a stochastic depth strategy for ResNet. ResNet-in-ResNet (RiR) [14] presents a generalized residual architecture. ResNeXt [15] introduces the split-transform-merge strategy to ResNet. WideResNet [44] decreases the depth and increases the width of ResNet. CRMNet [45] augments ResNet with a long short-term memory mechanism. ResNet-of-ResNet (RoR) [13] adds level-wise skip-connections to ResNet to achieve better results. Attention-ResNet [16] introduces an attention mechanism to guide feature learning. Res2Net [46] constructs hierarchical residual-like connections within each single residual block to generate multiscale feature maps. Scale-Aggregation-Net [24] proposes a multiscale aggregation strategy for residual learning to enhance feature representation and reduce computation complexity. In this paper, we propose a multipath residual structure to replace the original single-path residual structure proposed in ResNet [9], which can further improve network feature forward-propagation and gradient back-propagation abilities.

Deconvolution has been well discussed in [47]–[49]. It can expand the plane size of an input tensor, and therefore often appears in vision tasks that need planar reconstruction. In these tasks, deconvolution layers are often attached to a classification backbone to reconstruct their feature maps, as seen in DeconvNet [50], DSSD [51], RRC [52], and HourglassNet [53]. In HourglassNet [53], each block is also built by a symmetrical convolution layer and deconvolution layer and the whole architecture is built by several stacked blocks. However, different from HourglassNet, our ScaleNet does not reduce the plane size of the feature maps in each block to 1 x 1 and therefore can always maintain high resolution of the feature maps.

III. NETWORK ARCHITECTURE

A. CONV-DECONV COUPLE

Our network architecture is shown in Fig.2. The network contains three successive processing stages (*i.e.*, stages 1, 2, and 3), each containing several stacked isomorphic blocks (surrounded by dotted boxes), with each block being a conv-deconv couple. There are two kinds of conv-deconv couples in Fig.2, Conv-DeConv couple and Conv-Conv-DeConv-DeConv couple, which are named SingleCouple and DoubleCouple, respectively. SingleCouple is constructed by two stacked conv-unit and deconv-unit. DoubleCouple is constructed by two SingleCouple blocks, with one embedded in the other. These two couples can be used independently to build a network, and can also be mixed to build a network, as shown in Fig.2. In addition, the lowercases (conv or deconv) refer to the convolution and deconvolution operations in general sense, while the uppercases (Conv or DeConv) refer to the convolution and deconvolution units in Fig.2.

1) HYPER-PARAMETERS

All Conv1 and Conv2 units in Fig.2 have the same settings:*i.e.*, kernel size, 3; stride, 2; padding, 1; dilation, 1;

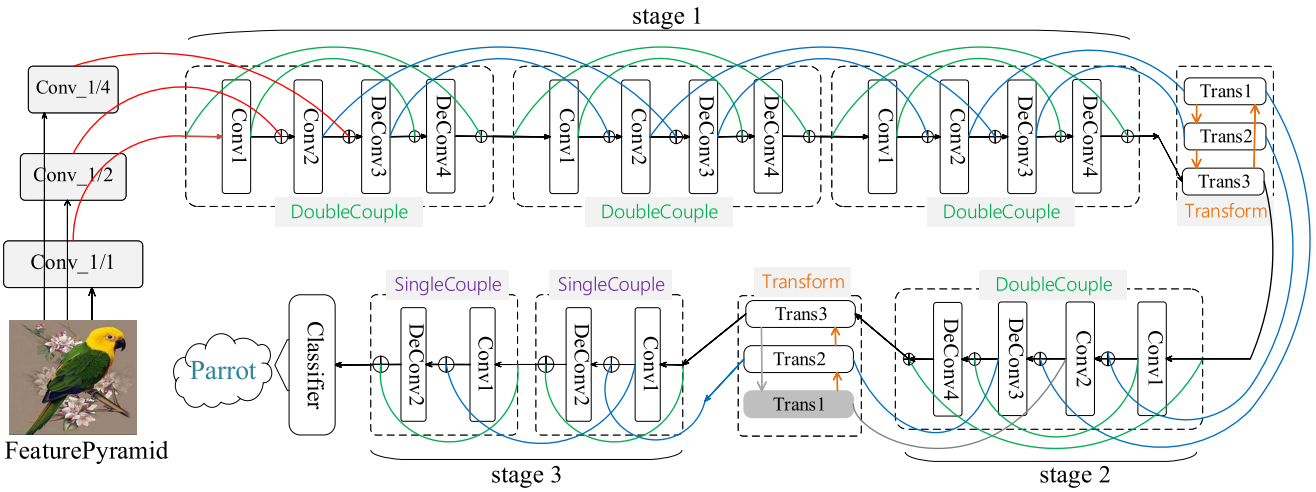


FIGURE 2. Architecture of ScaleNet. ⊕ is summation sign.

and no bias. All deconv-units in Fig.2, including DeConv2, DeConv3, and DeConv4, have the same settings:*i.e.*, kernel size, 3; stride, 2; padding, 1; dilation, 1; output padding, 1; and no bias. For each conv-unit and deconv-unit, BatchNorm and ReLU are attached to form BatchNorm-ReLU-Conv or BatchNorm-ReLU-DeConv, respectively.

2) FEATURE DIMENSIONS

In a conv-deconv block, the input feature maps are first down-sampled by one or two (one for SingleCouple, two for DoubleCouple, same below) conv-units, and then up-sampled by one or two deconv-units. Because conv-units and deconv-units are always pair-coupled and have the same stride of 2, the output plane size of a block remains the same as its input plane size. In addition, if conv-units change (increase or decrease) the feature channels to some value, the coupled deconv-units will restore the feature channels to the original value. Thus, the output feature channels of a block remain the same as the input feature channels. Therefore, a conv-deconv block can always ensure its output dimensions are the same as its input dimensions. Furthermore, a processing stage, constructed by stacking many isomorphic conv-deconv blocks, can also ensure the output dimensions are the same as the input dimensions.

3) TRANSFORM BLOCKS

Transform blocks are placed between every two adjacent stages to halve the feature size and increase the feature channels. As shown in Fig.2, there are two transform blocks between stages 1 and 2 and between stages 2 and 3. They contain three (or two) independent transform units (Trans 1, 2, and 3) to first process their parallel inputs independently. The feature maps with the same plane size are then concatenated as the final outputs. The yellow arrows represent the concatenation operations. Each transform unit contains two conv-units, the first has a kernel size of 3, stride of 1, and padding of 1, whereas the second has a kernel size of 2, stride of 2, and padding of 0. BatchNorm and ReLU are attached to form BatchNorm-ReLU-Conv.

4) NAMING RULES

If a processing stage adopts DoubleCouple blocks, it is denoted with a postfix '−D'; if SingleCouple blocks are adopted, the stage is denoted with a postfix '-S'. For example, a network with three stages and 40 layers is denoted as ScaleNet40-$D_5D_3S_2$, which means that its first stage contains five DoubleCouple blocks ($-D_5$), second stage contains three DoubleCouple blocks (D_3), and the third stage contains two SingleCouple blocks (S_2). A depth of 40 indicates that the total number of convolution, deconvolution, and linear layers is 40.

B. MULTIPATH RESIDUAL STRUCTURE

The overall view of our multipath residual structure is shown in Fig.2 and includes the green skip-connections within each conv-deconv block and the blue skip-connections across every two adjacent conv-deconv blocks. We call this residual structure a multipath residual structure. If we only retain the outermost green skip-connection within each block and remove all other skip-connections, it is the same as pre-active-ResNet [12], *i.e.*, a single-path residual structure.

We denote the three successive DoubleCouple blocks in **stage.1** (see Fig.2) as B_{i-1}, B_i, B_{i+1}. For block B_i, the weights and outputs of every unit (including Conv1, Conv2, DeConv3, and DeConv4) can be denoted as $(C_1^i, C_2^i, D_3^i, D_4^i)$ and $(x_1^i, x_2^i, x_3^i, x_4^i)$, respectively. The output of the last summing node in this block can be denoted as X^i. We can then integrate BatchNorm and ReLU into the Conv/DeConv unit and obtain the following expressions to analyze the relationships among the three blocks.

According to the data flow in these three blocks (B_{i-1}, B_i, B_{i+1}), we have:

$$X^i = x_4^i + X^{i-1} \tag{1}$$

$$\begin{aligned} X^{i+1} &= x_4^{i+1} + X^i \\ &= (A + D)X^i + Bx_3^i + Cx_2^i + X^i \end{aligned} \tag{2}$$

$$\begin{aligned} inside, \quad A &= D_4^{i+1} D_3^{i+1} C_2^{i+1} C_1^{i+1} \\ B &= D_4^{i+1} D_3^{i+1} C_2^{i+1} \\ C &= D_4^{i+1} D_3^{i+1} \\ D &= D_4^{i+1} C_1^{i+1} \end{aligned}$$

Then we can calculate the residual of block B_{i+1}:

$$Res_{single}^{i+1} = X^{i+1} - X^i = AX^i = AX^{i-1} + Ax_4^i \tag{3}$$

$$\begin{aligned} Res_{multi}^{i+1} &= X^{i+1} - X^i \\ &= AX^i + (DX^i + Bx_3^i + Cx_2^i) \\ &= Res_{single}^{i+1} + DX^{i-1} + Dx_4^i + Bx_3^i + Cx_2^i \end{aligned} \tag{4}$$

where, Res_{single}^{i+1} refers to the residual of a single-path residual structure and Res_{multi}^{i+1} refers to the residual of a multipath residual structure. Comparing equations (3) and (4), if we disable D, B and C by changing the skip-connections from multipath to single-path, then Res_{multi}^{i+1} will equal Res_{single}^{i+1}. Obviously, this demonstrates that the multipath residual structure is an extension of the original residual structure and can thereby spread the residual calculation of a block to more previous layers.

IV. EXPERIMENTS AND RESULTS

A. DATASETS AND DATA AUGMENTATION

The CIFAR-10 and CIFAR-100 datasets [54] both contain 50k training images and 10k test images with a size of $3 \times 32 \times 32$. The CIFAR-10 dataset consists of 10 categories, whereas the CIFAR-100 dataset consists of 100 categories. As per [5], [9], [10], [12], the original training set of both CIFAR-10 and CIFAR-100 are split into 45k for training and 5k for validation. A standard data augmentation scheme, as used in [5], [9], [10], [12], is adopted for their training: *i.e.*, four pixels are padded on each side and a 32 x 32 crop is randomly sampled from the padded image or its horizontal flip. Validation is carried out at the end of each training epoch. Testing is carried out on a single view of the original 32×32 image, and the final test error rate is reported on the 10k test sets.

The ImageNet2012 [55] dataset consists of 1.2 million training images and 50k validation images, containing 1000 categories and variable sizes. The test error rate is reported on its 50k validation images (denoted as ImageNet-1k val). For training on ImageNet, a standard data augmentation scheme provided by PyTorch is adopted, in which a random-sized crop ($0.08 \sim 1.0$ of original size) and random aspect ratio ($3/4 \sim 4/3$ of original aspect ratio) is first made on an image or its horizontal flip, after which the crop is resized to 224×224. The color augmentation and normalization used in other studies [9], [10], [12], [56] are also adopted. For testing on ImageNet, we adopt 1-crop testing and 10-crop testing, in which the crop size is 224×224 on a 256×256 image and 320×320 on a 350×350 image. In both testing situations, the crops are normalized by the mean and variance of ImageNet.

B. LEARNING SCHEDULE

Both CIFA-10 and CIFA-100 are trained on 2× NVIDIA TitanX GPUs with a batch size of 128. The optimizer is a stochastic gradient descent (SGD) with a learning rate starting from 0.1 and divided by 10 at the 240^{th} and 270^{th} epochs, respectively, with 300 training epochs in total. The weight decay and momentum are 0.0001 and 0.9, respectively. ImageNet is trained on 4× NVIDIA 1080Ti GPUs with a batch size of 256, and the learning schedule is the same as that in [9], [10], [12], [56]. The optimizer is SGD with a learning rate starting from 0.1 and divided by 10 at every 30 epochs, with a total of 100 training epochs. The weight decay and momentum are 0.0001 and 0.9, respectively. The weights are initialized as in [57] and BatchNorm as in [58]. For all our networks, the loss functions are cross entropy loss and no dropout is adopted.

C. RESULTS ON CIFAR-10 AND CIFAR-100

Several ScaleNets with different parameter scales are designed for CIFAR-10 and CIFAR-100. The experimental results are shown in Table.1, together with current results from several other papers, especially ResNets and its variants. Our best results are marked in bold and the best overall results are marked in blue.

1) ACCURACY

For the CIFAR-10 dataset, our deep ScaleNet with a distinct parameter reduction of 30% (1.2M parameters and 222 layers) achieves better performance (5.39%) than that of ResNet (6.41%, 1.7M), SD-ResNet (5.43%, 1.7M), pre-ResNet (5.46%, 1.7M), ELU-ResNet (5.62%, 1.7M), and PELU-ResNet (5.37%, 1.7M). Our deep ScaleNet with 1.7M parameters and 444 layers achieves a far better error rate (4.96%) than that of ResNet (6.41%, 1.7M), SD-ResNet (5.43%, 1.7M), pre-ResNet (5.46%, 1.7M), RiR (5.01%, 10.3M), ELU-ResNet (5.62%, 1.7M), PELU-ResNet (5.37%, 1.7M), FitResNet (5.39%, 2.5M), and Attention-ResNet (4.99%, 1.9M). Meanwhile, our wide ScaleNet with 1.7M parameters and 98 layers also shows an approximate error rate of 5.06%. Furthermore, the deep ScaleNet with 10M parameters achieves better performance (4.46%, 1432L) than that of SD-ResNet(4.91%, 1202L, 10M), Pre-ResNet (4.62%, 1001L, 10M), Wide ResNet (4.81%, 16L, 11M), RiR (5.01%, 18L), and CRMN (4.65%, 40M+).

For the CIFAR-100 dataset, our deep ScaleNet with a substantial parameter reduction of 30% (1.2M parameters) performs better (24.13%) than ResNet (27.22%, 1.7M), SD-ResNet (24.58%, 1.7M), pre-ResNet (24.33%,1.7M), ELU-ResNet (26.55%, 1.7M), PELU-ResNet (25.04%, 1.7M), and FitResNet (27.66%, 2.5M). In addition, the wide

TABLE 1. Classification performance of ScaleNets on CIFAR-10 and CIFAR-100. *Params* refers to the parameter scales of networks, with a unit of M (1M means 10e6).

Model	Depth	Params (M)	CIFAR-10+	CIFAR-100+
FractalNet [59]	21	38.6	5.22	23.30
With dropout/drop-path	21	38.6	4.60	23.73
ResNet [9]	110	1.7	6.61	-
Reported by [11]	110	1.7	6.41	27.22
	1202	19.4	7.93	-
ResNet-	110	1.7	5.23	24.58
Stochastic-Depth [11]	1202	10.2	4.91	-
ELU-ResNet [42]	110	1.7	5.62	26.55
PELU-ResNet [43]	110	1.7	5.37	25.04
Wide ResNet [44]	16	11.0	4.81	22.07
	28	36.5	4.17	20.50
pre-active-ResNet [12]	164	1.7	5.46	24.33
	1001	10.2	4.62	22.71
Attention-ResNet [16]	92	1.9	4.99	21.71
	236	5.1	4.14	21.16
	452	8.6	3.90	20.45
FitResNet [60]	-	2.5M	5.84	27.66
ResNet-in-ResNet [14]	18	10.3	5.01	22.90
CRMN [45]	28	40	4.65	20.35
ResNet-of-ResNet [13]	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
DenseNet-BC [10]	100	0.8	4.51	22.27
	100	7.0	4.10	20.20
	250	15.3	3.62	17.60
Wide ScaleNet:				
ScaleNet-D	32	1.0	5.50	25.18
ScaleNet-DDS	98	1.7	5.06	23.24
	90	10.0	5.10	21.97
	157	15.0	**4.29**	**21.33**
Deep ScaleNet:				
ScaleNet-DDS	632	1.0	5.48	28.08
ScaleNet-DS	222	1.2	5.39	24.13
ScaleNet-DDS	444	1.7	4.96	25.25
	1432	10.0	4.46	23.37
	2000	15.0	4.53	-

ScaleNet with 1.7M parameters achieves a much lower error rate (23.24%) than the above models. Our wide ScaleNet with 10M parameters also attains a lower error rate (21.97%) than that of Wide ResNet (22.07%, 11M) and RiR (22.90%, 10M). Thus, our results from CIFAR-10 and CIFAR-100 confirm the marked advantage of ScaleNet over ResNet [9], SD-ResNet [11], Wide-ResNet [44], pre-ResNet [12], ELU-ResNet [42], PELU-ResNet [43], FitResNet [60], and RiR [14].

2) OVERFITTING

The training and validation curves of a very deep ScaleNet with 2000 layers are shown in Fig.3. It contains 272 DoubleCouple blocks in the first stage, indicating that the feature maps in this stage will continuously vary 272 times in [32 × 32, 16 × 16, 8 × 8, 16 × 16, 32 × 32]. Its second stage contains 200 DoubleCouple blocks, indicating that the feature maps in this stage will vary 200 times in [16 × 16, 8 × 8, 4 × 4, 8 × 8, 16 × 16]. The third stage contains 50 SingleCouple blocks and the feature maps in this stage will vary 50 times in [8 × 8, 4 × 4, 8 × 8]. From the train-loss and val-loss curves in Fig.3, we can clearly see slight over-fitting in the training stage at the iteration of 64k. However, the val-prec curve still maintains a slow rise and does not worsen with the rise of val-loss, suggesting that over-fitting is within an acceptable range. Though it is an extremely deep network,

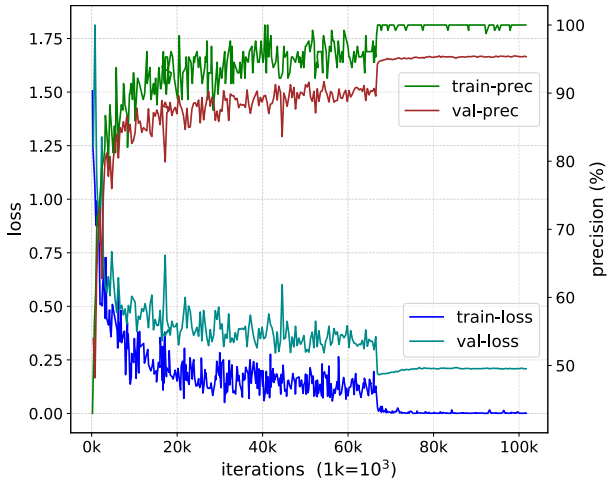


FIGURE 3. ScaleNet2000-$D_{272}D_{200}S_{50}$ (15M, 95.47%) on CIFAR-10: training loss curve (*train-loss*), validation loss curve (*val-loss*), training precision curve (*train-prec*), and validation precision curve (*val-prec*).

even deeper than ResNet (1202L, 10.2M, 7.93%, in Table.1) by 1.6×, serious over-fitting does not exist, and it achieves a much lower error rate of 4.53%.

D. RESULTS ON IMAGENET

We compared ScaleNet with ResNet [9] and DenseNet [10] on ImageNet [55] classification tasks. The ResNets and DenseNets are the pre-trained models provided by PyTorch. Table.2 shows the evaluation results of the ScaleNets on ImageNet-1k val based on 320 × 320 crops from 350 × 350 images. Fig.4 shows the top-1 validation error rates on ImageNet-1k val as a function of learned parameters and floating-point operations (FLOPs), as well as their inference time and inference memory on a single TitanX GPU. Evaluation is carried out on 224 × 224 crops from 256 × 256 images.

As seen in Fig.4 (a, b), our ScaleNets significantly outperform the ResNets under relatively small parameter scales; however, the DenseNets perform better than the ResNets and ScaleNets in regard to parameter efficiency and FLOPs. For any classification network, inference time and memory are two irreplaceable aspects reflecting computation costs. Both parameter scales and FLOPs are reflected on these two variables. Thus, we conducted an experiment to comprehensively compare the inference time and memory of different networks, with results shown in Fig.4 (c). As seen in this figure, the ScaleNets perform similarly to ResNets, but outperform DenseNets in regard to both inference time and memory. Even the largest ScaleNet-58(30M) is better than the smallest DenseNet-121(8M) for both inference time and memory. Thus, under the trade-off of efficiency and accuracy, ScaleNet performance is competitive against these state-of-the-art networks.

The experimental method of Fig.4(c). We evaluat networks with a fake input image (random tensor with a shape of 1 × 3 × 224 × 224) on a single NVIDIA TitanX GPU for 5000 cycles and count the running time and GPU memory usage. Because

TABLE 2. 1-crop and 10-crop validation on ImageNet-1k val.

Model	Depth	Params(M)	FLOPs(G)	Top-1 Error		Top-5 Error	
				1-crop	10-crops	1-crop	10-crops
ScaleNet-D_{25}	103	5.1	4.8	27.10	26.14	9.15	8.42
ScaleNet-D_8D_8	68	10.1	4.8	25.57	24.66	7.71	7.04
ScaleNet-$D_3D_3S_3$	44	9.4	2.4	26.98	25.80	8.58	8.01
ScaleNet-$D_3D_3S_3$	38	15.0	3.0	25.32	24.20	7.81	7.10
ScaleNet-$D_4D_4S_4$	48	20.1	4.0	24.11	23.15	7.01	6.50
ScaleNet-$D_4D_5S_5$	54	25.0	4.6	23.75	22.77	6.87	6.39
ScaleNet-$D_4D_5S_6$	59	30.5	5.4	**23.46**	**22.30**	**6.77**	**6.12**

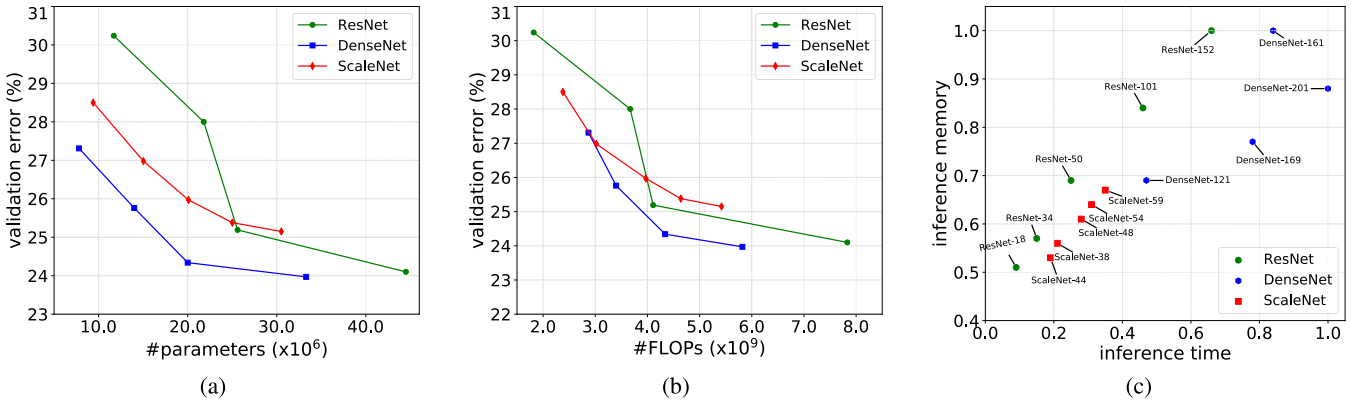


FIGURE 4. Top-1 validation error rate as a function of learned parameters (a) and FLOPs (b). Comparison of inference memory and inference time (c).

the input is a very small tensor and can be initialized in GPU memory, there is no time or memory cost for data loading and preprocessing. Therefore, the running time and GPU memory used are only related to the network inference processing. The running time is counted from the start to the end of 5000 cycles. The GPU memory is read from the NVIDIA-SMI board, which can be opened by a command line '*watch -n 1 nvidia-smi*' on Linux OS. After a very short start-up stage ($<$ 8s), the *Memory-Usage* and *GPU-Util* shown on the NVIDIA-SMI board reach their respective stable values, and thereafter remain unchanged to the end of 5000 cycles. Thus, the *GPU-Util* values are always between 96%–98% for all networks compared, which proves that the GPU always runs at full speed. Under this state, we record the *Memory-Usage* value shown on NVIDIA-SMI board as the ***inference memory*** of a network. In addition, we record the average running time of 5000 cycles as the ***inference time*** of a network. Lastly, to eliminate device impact, inference time is normalized to (0, 1] by dividing the inference time of each network by the maximum inference time of all networks. Inference memory and precision are similarly normalized to (0, 1]. The final results in Fig.4 (c) are relative values of inference time, memory, and precision based on all networks, thus providing more reasonable comparison between efficiency and accuracy.

E. ABLATION STUDIES

1) CONV-DECONV BLOCK

We use DoubleCouple blocks to build two one-stage ScaleNets for CIFAR-10, *i.e.*, ScaleNet32-D_7(1.0M) and ScaleNet44-D_{10}(2.0M). The feature maps generated by ScaleNet44-D_{10} are shown in Fig.1. This figure demonstrates that ScaleNet can completely remove the constraints between feature size and feature semantics. The high-resolution feature maps can be found at any depth level of this network. Table.3 shows the classification results, which demonstrate that ScaleNet performs much better than ResNet and pre-active-ResNet. In particular, the numbers of DeConv-units and Conv-units are almost equal in the ScaleNets, thus demonstrating the effectiveness of feature extraction based on a strong deconstruction-reconstruction process.

TABLE 3. Classification performance of two one-stage ScaleNets on CIFAR-10.

Model	Conv:DeConv	Error
ResNet110 (1.7M)	110 : 0	6.41%
pre-active-ResNet164 (1.7M)	164 : 0	5.46%
ScaleNet32(1.0M)	17 : 14	5.50%
ScaleNet44(2.0M)	23 : 20	5.25%

2) COMPARISON WITH POOLING AND INTERPOLATION

In addition to the convolution layers of stride 2 (denoted as **A**) and deconvolution layers of stride 2 (denoted as **B**) used in Fig.2, other methods can be used to resize feature maps. For example, a convolution layer of stride 1 followed by a pooling layer of stride 2 (denoted as **C**) can halve the plane size of feature maps; whereas a interpolation layer of scale 2 followed by a convolution layer of stride 1 (denoted as **D**) can double the plane size of feature maps. In C and D, the pooling and interpolation layers are used to adjust the

TABLE 4. Comparison of different up-/down-sampling methods on *Error* rates(%), *FLOPs*(G) and inference *Time*(s).

Model	Params	A + B			A + D			C + D		
		Error	FLOPs	Time	Error	FLOPs	Time	Error	FLOPs	Time
ScaleNet124-D_{30}	1.8M	5.5	**0.63**	**0.058**	5.7	0.63	0.063	**5.1**(-0.4)	1.00 (58% ↑)	0.089 (53% ↑)
ScaleNet80-$D_5D_8S_{10}$	5.0M	5.9	**0.37**	**0.025**	5.8	0.37	0.027	**5.2** (-0.7)	0.59 (59% ↑)	0.036 (44% ↑)
ScaleNet92-$D_{13}D_8$	4.0M	22.2	**0.58**	**0.041**	23.1	0.58	0.044	**21.6** (-0.6)	0.90 (55% ↑)	0.061 (49% ↑)
ScaleNet92-$D_5D_{10}S_{12}$	6.0M	24.3	**0.43**	**0.028**	24.3	0.43	0.030	**23.8** (-0.5)	0.69 (60% ↑)	0.040 (43% ↑)

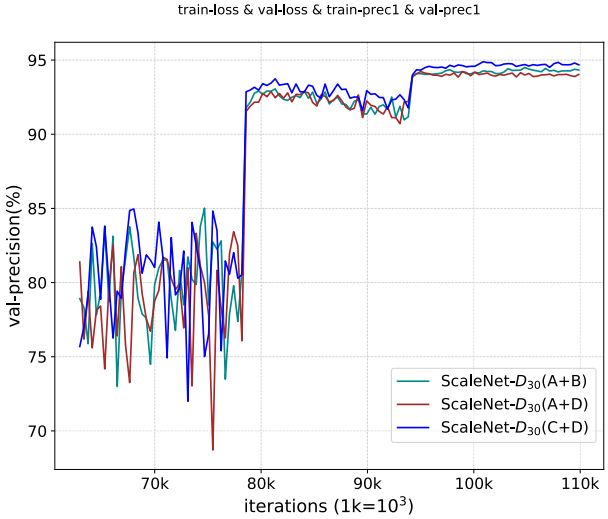


FIGURE 5. Val-precision curves of ScaleNet-D_{30} on CIFAR-10. Here, *A+B*, *A+D* and *C+D* refer to the three up-/down-sampling methods.

plane size of feature maps, whereas the convolution layers of stride 1 are used to adjust the channels of feature maps. Based on the design of C and D, ScaleNet can also ensure that both input and output display the same resolutions and channels within a block.

We compare these different up-/down-sampling methods (A, B, C, D) below. We use several standard ScaleNets with the design of A+B (denoted as ScaleNet(A+B)) as baseline networks, then first replace B with D to form ScaleNet(A+D), and then replace A with C to form ScaleNet(C+D). Table.4 shows the experimental results on CIFAR-10 and CIFAR-100, in which *Time* refers to the average inference time calculated on a single TitanX GPU with a total of 100 cycles. Fig.5 shows the validation curves of ScaleNet-D_{30} with different up-/down-sampling methods on CIFAR-10.

Comparing the two columns A+B and A+D in Table.4, we find that the two up-sampling methods (B and D) have the same parameters and FLOPs. The two ScaleNets on CIFAR-10 (ScaleNet-D_{30}, ScaleNet-$D_5D_8S_{10}$) show almost equal error rates (5.5% vs 5.7%; 5.9% vs 5.8%). Furthermore, from the two ScaleNets on CIFAR-100 (ScaleNet-$D_{13}D_8$, ScaleNet-$D_5D_{10}S_{12}$),A+B and A+D also exhibit similar performances (22.2% vs 23.1%; 24.8% vs 24.3%). This suggests that up-sampling method B is almost equal to D. Comparing the columns A+B and C+D, as well as A+D and C+D in Table.4, the models under C+D show lower error rates

TABLE 5. Comparison of different residual structures for ScaleNets on CIFAR-10 and CIFAR-100.

Dataset	Model	SinglePath	MultiPath
CIFAR-10	ScaleNet32 (1.0M)	6.96%	5.50%
	ScaleNet98 (1.7M)	6.72%	5.06%
CIFAR-100	ScaleNet32 (1.0M)	29.09%	25.18%
	ScaleNet98 (1.7M)	27.02%	23.24%

but higher FLOPs and inference time than the models under A+D for both CIFAR-10 and CIFAR-100. Of note, compared with the slight decrease in error rates, computation complexity (FLOPs) and inference time increase by more than 50% and 40% respectively, thus suggesting that down-sampling method C is better than A, but exhibits a serious side-effect of fast-increasing FLOPs and inference time. In summary, this comparative experiment reveals that the conv-deconv blocks used in our ScaleNet demonstrate a better trade-off between accuracy and efficiency.

3) MULTI-PATH RESIDUAL STRUCTURE

Here, we use ScaleNet32-D_7 and ScaleNet98-$D_{20}S_5$, without any skip-connections, as two backbones, and then add two kinds of skip-connections to them, *i.e.*, 'SinglePath' and 'MultiPath'. SinglePath method means only one inner skip-connection is built within each block. It is typically used in ResNet and pre-ResNet and is shown as the outermost green line within each block in Fig.2. Our MultiPath method includes all skip-connections shown in Fig.2. As skip-connections do not add new parameters to a backbone, the resulting networks retain the same parameter scales. As shown in Table.5, MultiPath method is far superior to the SinglePath method. This is in accordance with the theoretical proof in Section III-B, which convincingly suggests that a multipath residual structure can further strengthen feature forward propagation and gradient back-propagation and can therefore significantly promote the parameter efficiency of networks.

F. FEATURE SEMANTICS ANALYSIS

Here, we use three pre-trained one-stage ScaleNets (as in Fig.1) on CIFAR-10, CIFAR-100, and ImageNet as backbones, and then attach auxiliary classifiers to the smallest feature maps generated by each of their DoubleCouple blocks. All these feature maps possess the same dimensions, and thus the attached auxiliary classifiers have the same structure. We then train the auxiliary classifiers under the same init-state and training schedule. Obviously, because the feature

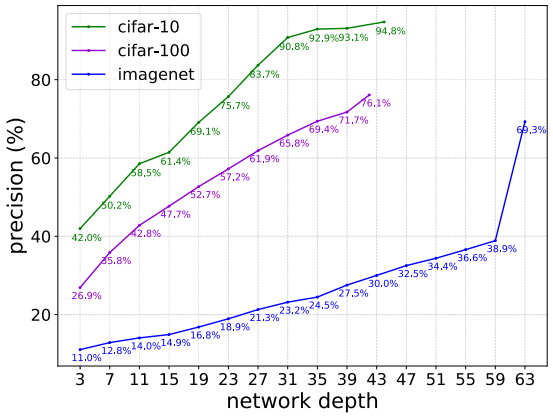


FIGURE 6. Classification results of all auxiliary classifiers and final main classifier.

maps are frozen, the training of the auxiliary classifiers is independent of each other. As seen in Fig.6, classification precision gradually increases with network depth. The auxiliary classifiers have the same potential learning ability but perform very differently according to the feature maps to which they are attached. Obviously, this reveals that the semantic information contained in the feature maps gradually increases with network depth. This further demonstrates the effectiveness of feature extraction based on a cascaded deconstruction-reconstruction process.

G. FEATURE-MAPS ANALYSIS

Here, to analyze the multiscale ability of ScaleNet, we visualize class activation maps (CAMs) using Grad-CAM [61], which is commonly applied to localize the discriminative regions for image classification. We compare the CAMs calculated on the last feature maps of ScaleNet103(5.08M, 5.26G, 30.46%), ResNet101(44.54M, 7.83G, 24.10%), and DenseNet161(28.68M, 7.79G, 23.97%), where M and G refer to parameter scales and FLOPs, respectively. The last feature maps are just ahead of the final linear classifier, which can be regarded as the final output of a CNN-based feature extractor. ResNet101 and DenseNet161 are the pre-trained models provided by PyTorch. The images used all come from ImageNet [55].

1) LEARNING FINE-GRAINED FEATURES

Fig.7 shows the CAMs of the three networks on five different images with the same input size of $3 \times 320 \times 320$. Clearly, the ScaleNet CAMs are very different from those of ResNet and DenseNet. ScaleNet tends to focus on the fine-grained details of an object, whereas ResNet and DenseNet always focus on a larger patch of an object. Under an input size of $3 \times 320 \times 320$, the plane size of the final feature maps is 10×10 for ResNet and DenseNet but is 40×40 for ScaleNet. Therefore, the final feature maps of ScaleNet exhibit much higher resolution than those of ResNet and DenseNet, thus contributing to the capture of fine-grained visual features.

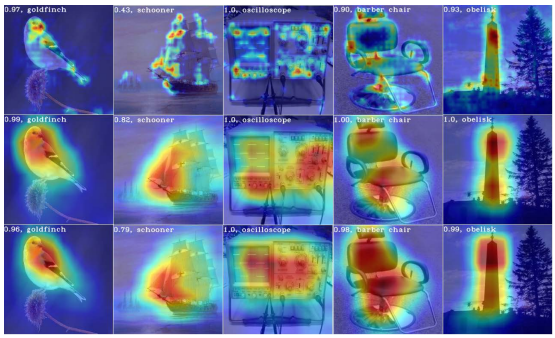


FIGURE 7. CAMs calculated on the last feature-maps of ScaleNet103 (top), ResNet101 (middle), and DenseNet161 (bottom).

2) MULTI-SCALE INPUT ANALYSIS

Here, to conduct stronger comparative experiments, we input images with a very large size range (*e.g.*, 32×32, 64×64, 128×128, 224×224, 256×256, 320×320, and 512×512) to the networks to compare their CAMs. Four different images are first stitched into a single image as the network input. The stitched image and related experimental results are shown in Fig.8, where goldfinches and fish are arranged on the diagonal.

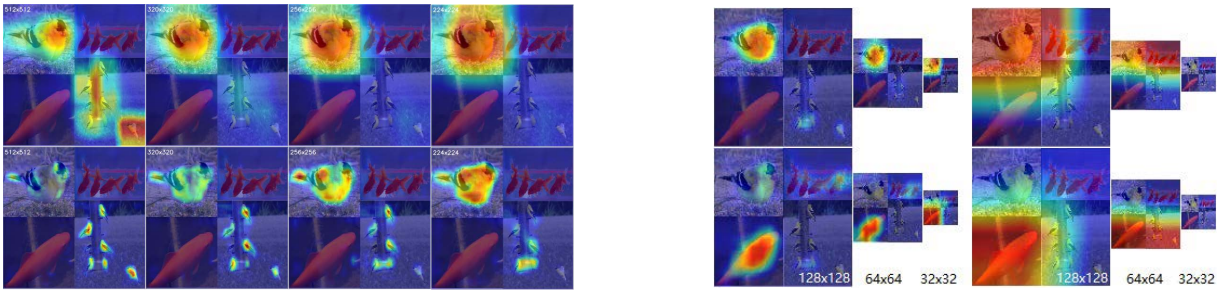
Based on comparison of the ScaleNet CAMs on birds and fish, the activated regions of the birds and fish do not interfere with each other and are semantically independent. This suggests that the representation is not a physical representation based on image textures, but a semantic representation based on object categories. For the ScaleNet CAMs, when the input size decreases from 512×512 to 32×32, the activated regions of the large bird (top-left corner of each image) vary from small details to relatively large patches, and finally to some very large patches that cover the whole bird. This reveals the continuous changing process of semantic representation from fine granularity to coarse granularity. In other words, under a very large scale range from 512×512 to 32×32, ScaleNet always finds an appropriate semantic granularity to represent the object. Compared with ScaleNet, the DenseNet CAMs in Fig.8 (a, top) and Fig.8 (c) fail to reflect such a process.

Of note, the activated regions of the small birds (bottom-right of each image, shown in Fig.8 (a)) with ScaleNet exhibit much clearer activation morphology, with each small bird activated independently. However, the activated regions of these small birds (shown in Fig.8 (a, top) with DenseNet are blurry and quickly disappear as input size decreases. Comparing Fig.8 (b) with Fig.8 (c), when the input size is very small, the activated regions of the large bird and large fish with DenseNet begin to diffuse in the image plane, whereas the activated regions with ScaleNet always converge on the objects. This strongly indicates that ScaleNet possesses better cross-scale representation ability.

V. DISCUSS

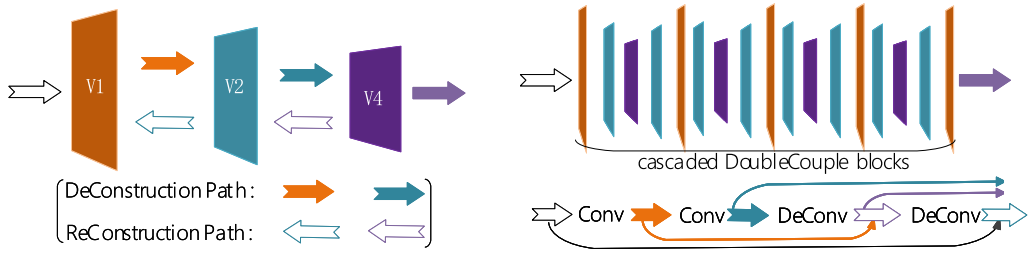
A. RELATIONSHIP WITH BIOLOGICAL VISION

In biological vision, not only is there a feature forward-propagation (**FFP**) mechanism, but also a feature



(**a**, top:DenseNet, bottom:ScaleNet) (**b**: ScaleNet) (**c**: DenseNet)

FIGURE 8. **Comparison of CAMs between ScaleNet103 and DenseNet161. All images in (a) have been resized to 300 × 300 for display, with original sizes marked.**



(a): visual cortices (b): conv-deconv blocks

FIGURE 9. **Expanding temporal information flow in visual cortices (*a*) to spatial domain of DoubleCouple blocks (*b*).**

(a): a stitched image (b): a single image

FIGURE 10. **Comparison of CAMs between ScaleNet103 (top) and DenseNet161 (bottom) on traffic lights. (a): all images have been resized to 300 × 300 for display. (b): the original image has been resized to several different sizes with its h/w ratio unchanged. In both figures, real sizes are marked at the top/bottom in white (for ScaleNet) and cyan (for DenseNet).**

back-propagation (**FBP**) mechanism [62], [63]. Fig.9 (a) shows a simple overview of the information flow among V_1, V_2 and V_4 cortices. Here, FBP refers to backward information flow from the advanced visual cortex to primary visual cortex, *e.g.*, from V_4 to V_2. It is completely unlike the forward information flow from the primary visual cortex to advanced visual cortex, *e.g.*, from V_1 to V_2. If we unfold the temporal information flow among visual cortices, then $V_1 \rightharpoonup V_2 \rightharpoonup V_4 \rightharpoonup V_2 \rightharpoonup V_1 \rightharpoonup V_2 \rightharpoonup V_4 \cdots$. Thus, to mimic this function, we can unfold the temporal information flow among visual cortices as a spatial flow among conv-deconv blocks, *i.e.*, $Cov1 \rightharpoonup Conv2 \rightharpoonup DeConv1 \rightharpoonup DeConv2 \rightharpoonup Cov1 \cdots$, as shown in Fig.9 (b). This concept is another motivation for this paper, where we introduce some biological visual mechanisms to improve CNN architecture. However, it should be pointed out that as understanding of the visual cortices is incomplete, our method can be seen as a simplified functional imitation of visual cortices.

B. UNSUPERVISED OBJECT DETECTION

Fig.10 shows the ScaleNet103 and DenseNet161 CAMs on images of traffic lights. Fig.10 (a) uses an image stitched together by 12 images, which all contain traffic lights with different scales, shapes, and illuminations. Fig.10 (b) uses

an image selected from Fig.10 (a). As shown in the figures, without the standard pipeline of mainstream detection algorithms (such as anchors sampling, bounding-boxes regression, and supervised training by ground-truth boxes), our ScaleNet exactly locates the traffic lights in an image under a very large scale range of 500 to 100. Obviously, the location ability of ScaleNet benefits from its fine-grained features and high-resolution feature maps, because under coarse-grained features and low-resolution feature maps (*e.g.*, 7×7 in DenseNet), the CAMs fail to exhibit this ability. Thus, CAMs driven by fine-grained features can accurately reflect an object's position on an image.

Based on Fig.10, CAMs are a promising method for achieving unsupervised object detection. It is conceivable that CAM workflow and high-resolution feature maps with strong semantic information can be combined to perform traversal checks and screenings on objects in an image. Because CAMs work by class labels instead of ground-truth boxes, the final detection algorithms may be able to work without ground-truth boxes. Although the results in Fig.10 show their potential, more implementation details need to be researched.

VI. CONCLUSION

In this paper, we realize image feature extraction based on an end-to-end and cascaded deconstruction-reconstruction process. A multipath residual structure is also proposed to improve the parameter efficiency of networks. Compared to the state-of-the-art networks, our ScaleNet achieves competitive performance under a trade-off between accuracy and efficiency. Furthermore, ScaleNet can remove the constraints between feature size and feature semantics, thereby realizing multiscale feature extraction at any network depth. ScaleNet combines high-resolution feature maps, fine-grained features, and strong semantic representation, thus hinting at a promising way in which to achieve unsupervised object detection.

ACKNOWLEDGMENT

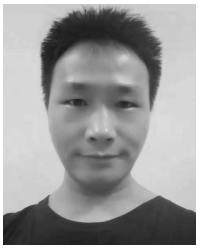
The authors thank NVIDIA's Academic Programs for donating a GPU used in the early stage of this project.

REFERENCES

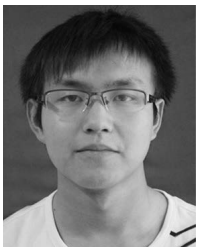
[1] M. Lin, Q. Chen, and S. Yan, "Network in network," 2014, *arXiv:1312.4400*. [Online]. Available: https://arxiv.org/abs/1312.4400
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.
[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, Jun. 2016, pp. 2818–2826.
[5] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. AISTATS*, Feb. 2015, pp. 562–570.
[6] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015, *arXiv:1505.00387*. [Online]. Available: https://arxiv.org/abs/1505.00387
[7] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[8] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, Mar. 2010, pp. 249–256.
[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, Jul. 2017, pp. 4700–4708.
[11] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*, 2016, pp. 646–661.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. ECCV*, 2016, pp. 630–645.
[13] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1303–1314, Jun. 2018.
[14] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," 2016, *arXiv:1603.08029*. [Online]. Available: https://arxiv.org/abs/1603.08029
[15] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*, Jul. 2017, pp. 1492–1500.
[16] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. CVPR*, Jul. 2017, pp. 3156–3164.
[17] F. Shen, R. Gan, and G. Zeng, "Weighted residuals for very deep networks," in *Proc. ICSAI*, Nov. 2016, pp. 936–941.
[18] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proc. CVPR*, Jul. 2017, pp. 472–480.
[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, Jun. 2018, pp. 4510–4520.
[20] Y. Shen and J. Gao, "Refine or represent: Residual networks with explicit channel-wise configuration," in *Proc. IJCAI*, Jul. 2018, pp. 2682–2688.
[21] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *Proc. CVPR*, Jul. 2017, pp. 1039–1048.
[22] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, and S. Peng, "Rectified exponential units for convolutional neural networks," *IEEE Access*, vol. 7, pp. 101633–101640, 2019.
[23] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*, 2019, pp. 1–10.
[24] Y. Li, Z. Kuang, Y. Chen, and W. Zhang, "Data-driven neuron allocation for scale aggregation networks," in *Proc. CVPR*, Jun. 2019, pp. 11526–11534.
[25] H. Hu, D. Dey, A. D. Giorno, M. Hebert, and J. A. Bagnell, "Log-DenseNet: How to sparsify a DenseNet," 2018, *arXiv:1711.00002*. [Online]. Available: https://arxiv.org/abs/1711.00002
[26] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. ICLR*, 2017, pp. 1–14.
[27] G. Huang, S. Liu, L. Van Der Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. CVPR*, Jun. 2018, pp. 2752–2761.
[28] W. Liu and K. Zeng, "SparseNet: A sparse DenseNet for image classification," 2018, *arXiv:1804.05340*. [Online]. Available: https://arxiv.org/abs/1804.05340
[29] M. Wang, J. Zhou, W. Mao, and M. Gong, "Multi-scale convolution aggregation and stochastic feature reuse for DenseNets," in *Proc. WACV*, Jan. 2019, pp. 321–330.
[30] K. Zhang, Y. Guo, X. Wang, J. Yuan, and Q. Ding, "Multiple feature reweight DenseNet for image classification," *IEEE Access*, vol. 7, pp. 9872–9880, 2019.
[31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–16.
[32] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*. [Online]. Available: https://arxiv.org/abs/1710.05941
[33] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. ECCV*, Sep. 2018, pp. 19–34.
[34] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. ICLR*, 2018, pp. 1–13.
[35] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI*, 2018, pp. 4780–4789.

[36] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. CVPR*, Jun. 2019, pp. 2820–2828.
[37] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," in *Proc. ICLR*, 2018, pp. 1–17.
[38] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.
[39] F. P. Casale, J. Gordon, and N. Fusi, "Probabilistic neural architecture search," 2019, *arXiv:1902.05116*. [Online]. Available: https://arxiv.org/abs/1902.05116
[40] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, pp. 2117–2125.
[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 60, 2012, p. 84.
[42] A. Shah, E. Kadam, H. Shah, S. Shingade, and S. Shinde, "Deep residual networks with exponential linear unit," 2016, *arXiv:1604.04112*. [Online]. Available: https://arxiv.org/abs/1604.04112
[43] L. Trottier, P. Giguère, and B. Chaib-Draa, "Parametric exponential linear unit for deep convolutional neural networks," in *Proc. ICMLA*, Dec. 2017, pp. 207–214.
[44] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: https://arxiv.org/abs/1605.07146
[45] J. Moniz and C. Pal, "Convolutional residual memory networks," 2016, *arXiv:1606.05262*. [Online]. Available: https://arxiv.org/abs/1606.05262
[46] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A new multi-scale backbone architecture," 2019, *arXiv:1904.01169*. [Online]. Available: https://arxiv.org/abs/1904.01169
[47] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang, "Is the deconvolution layer the same as a convolutional layer?" 2016, *arXiv:1609.07009*. [Online]. Available: https://arxiv.org/abs/1609.07009
[48] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016, *arXiv:1603.07285*. [Online]. Available: https://arxiv.org/abs/1603.07285
[49] A. Odena, V. Dumoulin, and C. Olah. (2016). *Deconvolution and Checkerboard Artifacts*. [Online]. Available: http://distill.pub/2016/deconv-checkerboard
[50] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. ICCV*, Dec. 2015, pp. 1520–1528.
[51] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," 2017, *arXiv:1701.06659*. [Online]. Available: https://arxiv.org/abs/1701.06659?source=post_pageec516929be93
[52] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate single stage detector using recurrent rolling convolution," in *Proc. CVPR*, Jul. 2017, pp. 5420–5428.
[53] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. ECCV*, 2016, pp. 483–499.
[54] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. Jan. 2009, vol. 1. [Online]. Available: https://www.researchgate.net/publication/306218037_Learning_multiple_layers_of_features_from_tiny_images
[55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, S. M. Bernstein, C. A. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[56] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang, "FishNet: A versatile backbone for image, region, and pixel level prediction," in *Proc. NIPS*, 2018, pp. 754–764.
[57] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. ICCV*, Dec. 2015, pp. 1026–1034.
[58] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 1–11.
[59] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," 2017, *arXiv:1605.07648*. [Online]. Available: https://arxiv.org/abs/1605.07648
[60] D. Mishkin and J. Matas, "All you need is a good init," 2015, *arXiv:1511.06422*. [Online]. Available: https://arxiv.org/abs/1511.06422
[61] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. ICCV*, Oct. 2017, pp. 618–626.
[62] C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. Dicarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS Comput. Biol.*, vol. 10, no. 12, 2014, Art. no. e1003963.
[63] K. Kar, J. Kubilius, K. Schmidt, E. B. Issa, and J. J. DiCarlo, "Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior," *Nature Neurosci.*, vol. 10, pp. 974–983, Jun. 2019.

JINPENG ZHANG received the B.S. and M.S. degrees in material science and engineering from the Huazhong University of Science and Technology (HUST), in 2011 and 2014, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, in 2019. His research interests include machine learning and computer vision.



JINMING ZHANG received the B.S. and M.S. degrees in automation from Xi'an Jiaotong University, China, in 2011 and 2014, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, in 2019. His major research interests include intelligent optimization algorithms, machine learning, and computer vision.



GUYUE HU received the B.E. degree from the Hefei University of Technology, Hefei, China, in 2016. He is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, pattern recognition, and computational neuroscience, especially in multimodal video understanding and human behavior analysis.

YANG CHEN received the B.S. degree in physics from Minzu University, in 2006, the M.S. degree in physics from Beijing Normal University, in 2013, and the Ph.D. degree in physics from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017. From 2017 to 2019, he conducted a Postdoctoral research at the Institute of Automation, Chinese Academy of Sciences (CASIA), where he has been an Assistant Professor, since 2019. His current research interests include brain-inspired computing, machine learning, and time series analysis.

SHAN YU received the B.S. and Ph.D. degrees in biology from the University of Science and Technology of China, Hefei, China, in 2000 and 2005, respectively. From 2005 to 2014, he conducted a Postdoctoral research at the Max–Planck Institute of Brain Research, Germany, from 2005 to 2008, and the National Institute of Mental Health, USA, from 2008 to 2014. In 2014, he joined the Institute of Automation, Chinese Academy of Sciences (CASIA), as a recipient of the One Hundred Talents Program. Since 2014, he has been a Professor with the National Laboratory of Pattern Recognition (NLPR), Brainnetome Center, CASIA. He has been the Deputy Director of the NLPR, since 2018. He has published over 30 peer-reviewed articles in neuroscience and other interdisciplinary fields at leading international journals such as *Nature Machine Intelligence*, the *Journal of Neuroscience*, and *eLife*. His current research interests include neuronal information processing, brain-inspired computing, and artificial intelligence.