Multi-Scale Spatially-Asymmetric Recalibration for Image Classification

Yan Wang[1]*, Lingxi Xie[2]*, Siyuan Qiao[2],
Ya Zhang[1](✉), Wenjun Zhang[1], Alan L. Yuille[2]

[1] Cooperative Medianet Innovation Center, Shanghai Jiao Tong University
[2] Department of Computer Science, The Johns Hopkins University
tiffany940107@gmail.com, 198808xc@gmail.com, siyuan.qiao@jhu.edu,
{ya_zhang,zhangwenjun}@sjtu.edu.cn, alan.l.yuille@gmail.com

Abstract. Convolution is *spatially-symmetric*, *i.e.*, the visual features are independent of its position in the image, which limits its ability to utilize contextual cues for visual recognition. This paper addresses this issue by introducing a *recalibration* process, which refers to the surrounding region of each neuron, computes an importance value and multiplies it to the original neural response. Our approach is named **multi-scale spatially-asymmetric recalibration** (MS-SAR), which extracts visual cues from surrounding regions at *multiple scales*, and designs a weighting scheme which is *asymmetric in the spatial domain*. MS-SAR is implemented in an efficient way, so that only small fractions of extra parameters and computations are required. We apply MS-SAR to several popular building blocks, including the residual block and the densely-connected block, and demonstrate its superior performance in both CIFAR and ILSVRC2012 classification tasks.

Keywords: Large-scale image classification, convolutional neural networks, multi-scale spatially asymmetric recalibration

1 Introduction

In recent years, deep learning has been dominating in the field of computer vision. As one of the most important models in deep learning, the convolutional neural networks (CNNs) have been applied to various vision tasks, including image classification [19], object detection [7], semantic segmentation [23], boundary detection [41], *etc.* The fundamental idea is to stack a number of linear operations (*e.g.*, convolution) and non-linear activations (*e.g.*, ReLU [24]), so that a deep network has the ability to fit very complicated distributions. There are two prerequisites in training a deep network, namely, the availability of large-scale image data, and the support of powerful computational resources.

The first two authors contributed equally. This work was supported by the High Tech R&D Program of China 2015AA015801, NSFC 61521062, STCSM 18DZ2270700 and 2018 CSC-IBM Future Data Scientist Scholarship Program (Y-100), the NSF award CCF-1317376, and ONR N00014-15-1-2356. We thank Huiyu Wang for discussions.

Convolution is the most important operation in a deep network. A window is slid across the image lattice, and a number of small convolutional kernels are applied to capture local visual patterns. This operation suffers from a weakness of being *spatially-symmetric*, which assumes that visual features are independent of their spatial position. This limits the network's ability to learn from contextual cues (*e.g.*, an object is located upon another) which are often important in visual recognition. Conventional networks capture such spatial information by stacking a number of convolutions and gradually enlarging the receptive field, but we propose an alternative solution which equips *each* neuron with the ability to refer to its contexts at multiple scales efficiently.

Our approach is named **multi-scale spatially asymmetric recalibration** (MS-SAR). It quantifies the importance of each neuron by a score, and multiplies it to the original neural response. This process is named *recalibration* [13]. Two features are proposed to enhance the effect of recalibration. First, the importance score of each neuron is computed from a local region (named a *coordinate set*) covering that neuron. This introduces the factor of spatial position into recalibration, leading to the desired *spatially-asymmetric* property. Second, we relate each neuron to multiple coordinate sets of different sizes, so that the importance of that neuron is evaluated by incorporating *multi-scale* information. The conceptual flowchart of our approach is illustrated in Figure 1.

In practice, the recalibration function (taking inputs from the coordinate sets and outputting the importance score) is the combination of two linear operations and two non-linear activations, and we allow the parameters to be learned from training data. To avoid heavy computational costs as well as a large amount of extra parameters to be introduced, we first perform a regional pooling over the coordinate set to reduce the spatial resolution, and use a smaller number of outputs in the first linear layer to reduce the channel resolution. Consequently, our approach only requires small fractions of extra parameters and computations beyond the baseline building blocks.

We integrate MS-SAR into two popular building blocks, namely, the residual block [11] and the densely-connected block [15], and empirically evaluate its performance in two image classification tasks. In the CIFAR datasets [18], our approach outperforms the baseline networks, the ResNets [11] and the DenseNets [15]. In the ILSVRC2012 dataset [29], we also compare with SENet [13], a special case of our approach with single-scale spatially-symmetric recalibration and demonstrate the superior performance of MS-SAR. In all cases, the extra computational overhead brought by MS-SAR does not exceed 1%.

The remainder of this paper is organized as follows. Section 2 briefly reviews the previous literatures on image classification based on deep learning, and Section 3 illustrates the MS-SAR approach and describes how we apply it to different building blocks. After extensive experimental results are shown in Section 4, we conclude this work in Section 5.

2 Related Work

2.1 Convolutional Neural Networks for Visual Recognition

Deep convolutional neural networks (CNNs) have been widely applied to computer vision tasks. These models are based on the same motivation to learn and organize visual features in a hierarchical manner. In the early years, CNNs were verified successful in simple classification problems, in which the input image is small yet simple (*e.g.*, MNIST [20] and CIFAR [18]) and the network is shallow (*i.e.* with 3–5 layers). With the emerge of large-scale image datasets [4][22] and powerful computational resources such as GPUs, it is possible to design and train deep networks for recognizing high-resolution natural images [19]. Important technical advances involve using the piecewise-linear ReLU activation [24] to prevent under-fitting, and applying Dropout [32] to regularize the training process and avoid over-fitting.

Modern deep networks are built upon a handful of building blocks, including convolution, pooling, normalization, activation, element-wise operation (sum [11] or product [36]), *etc.* Among them, convolution is considered the most important module to capture visual patterns by template matching (computing the inner-product between the input data and the learned templates), and most often, we refer to the depth of a network by the maximal number of convolutional layers along any path connecting the input to the output. It is believed that increasing the depth leads to better recognition performance [34][31][11][3][15]. In order to train these very deep networks efficiently, researchers proposed batch normalization [17] to improve numerical stability, and highway connections [33][11] to facilitate visual information to be propagated faster. The idea of automatically learning network architectures was also explored [38][47].

Image classification lays the foundation of other vision tasks. The pre-trained networks can be used to extract high-quality visual features for image classification [5], instance retrieval [27], fine-grained object recognition [45][39] or object detection [8], surpassing the performance of conventional handcraft features. Another way of transferring knowledge learned in these networks is to fine-tune them to other tasks, including object detection [7][28], semantic segmentation [23][1], boundary detection [41], pose estimation [35][25], *etc.* A network with stronger classification results often works better in other tasks.

2.2 Spatial Enhancement for Deep Networks

One of the most important factor of deep networks lies in the spatial domain. Although the convolution operation is naturally invariant to spatial translation, there still exist various approaches aimed at enhancing the ability of visual recognition by introducing different *priors* into deep networks.

In an image, the relationship between two features is often tighter when their spatial locations are closer to each other. An efficient way of modeling such distance-sensitive information is to perform spatial pooling [10], which explicitly splits the image lattice into several groups, and ignores the diversity of features in the same group. This idea is also widely used in object detection to summarize visual features given a set of regional proposals [7][28].

On the other hand, researchers also noticed that spatial importance (saliency) is not uniformly distributed in the spatial domain. Thus, various approaches were designed to discriminate the important (salient) features from others. Typical examples include using gradient back-propagation to find the neurons that contribute most to the classification result [43][39], introducing saliency [30][26] or attention [2] into the network, and investigating local properties (*e.g.*, smoothness [37]). We note that a regular convolutional layer also captures local patterns in the spatial domain, but (i) it performs linear template matching and so cannot capture non-linear properties (*e.g.*, smoothness), meanwhile (ii) it often needs a larger number of parameters and heavier computational overheads.

In this work, we consider a *recalibration* approach [13], which aims at revising the response of each neuron by a spatial weight. Unlike [13], the proposed approach utilizes multi-scale visual information and allows different weights to be added at different spatial positions. This brings significant accuracy gains.

3 Our Approach

3.1 Motivation: Why Spatial Asymmetry is Required?

Let $\mathbf{X}$ be the output of a convolutional layer. This is a 3D cube with $W \times H \times D$ entries, where W and H are the width and height, indicating the spatial resolution, and D is the depth, indicating the number of convolutional kernels. According to the definition of convolution, each element in $\mathbf{X}$, denoted by $x_{w,h,d}$, represents the intensity of the d-th visual pattern at the coordinate (w,h), which is obtained from the inner-product of the d-th convolutional kernel and the input region corresponding to the coordinate (w,h).

Here we notice that convolution performs *spatially-symmetric* template matching, in which the intensity $x_{w,h,d}$ is independent of the spatial position (w,h). We argue that this is not the optimal choice. In visual recognition, we often hope to learn contextual information (*e.g.*, feature d_1 often appears upon feature d_2), and so the *spatially-asymmetric* property is desired. To this end, we define $\mathcal{S}_{w,h}$ to be the *coordinate set* containing the neighboring coordinates of (w,h) (detailed in the next subsection). We aim at computing a new response $\tilde{x}_{w,h,d}$ by taking into consideration all neural responses in $\mathcal{S}_{w,h} \times \{1,2,\ldots,D\}$, where × denotes the Cartesian product. Our approach is related but different from several existing approaches.

- First, we note that a standard convolution can learn contexts in a small local region, *e.g.*, $\mathcal{S}_{w,h}$ is a 3×3 square centered at (w,h). Our approach can refer to multiple $\mathcal{S}_{w,h}$'s at different scales, capturing richer information and being more computationally efficient than convolution.
- The second type works in the spatial domain, which uses the responses in the set $\mathcal{S}_{w,h} \times \{d\}$ to compute $\tilde{x}_{w,h,d}$. Examples include the Spatial Pyramid

Pooling (SPP) [10] layer which set regular pooling regions and ignored feature diversity within each region, and the Geometric Neural Phrase Pooling (GNPP) [37] layer which took advantage of the spatial relationship of neighboring neurons (it also assumed that spatially closer neurons have tighter connections) to capture feature co-occurrence. But, both of them are non-parameterized and work in each channel individually, which limited their ability to adjust feature weights.

- Another related approach is called feature recalibration [13], which computed $\tilde{x}_{w,h,d}$ by referring to the visual cues in the entire image lattice, *i.e.*, the set $\{(w,h)\}_{w=1,h=1}^{W,H} \times \{1,2,\ldots,D\}$ was used. This is still a spatially-symmetric operation. As we shall see later, our approach is a generalized version and produces better visual recognition performance.

3.2 Formulation: Spatially-Asymmetric Recalibration

Given the neural responses cube $\mathbf{X}$ and the coordinate set $\mathcal{S}_{w,h}$ at (w,h), the goal is to compute a revised intensity $\tilde{x}_{w,h,d}$ with spatial information taken into consideration. We formulate it as a weighting scheme $\tilde{x}_{w,h,d} = x_{w,h,d} \times z_{w,h,d}$, in which $z_{w,h,d} = f_d(\mathbf{X}, \mathcal{S}_{w,h})$ and $f_d(\cdot)$ is named the *recalibration function* [13]. This creates a weighting cube $\mathbf{Z}$ with the same size as $\mathbf{X}$ and propagate $\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{Z}$ to the next network layer. We denote the D-dimensional feature vector of $\mathbf{X}$ at (w,h) by $\mathbf{x}_{w,h} = [x_{w,h,1};\ldots;x_{w,h,D}]^\top$, and similarly for $\tilde{\mathbf{x}}_{w,h}$ and $\mathbf{z}_{w,h}$.

Let the set of all spatial positions be $\mathcal{P} = \{(w,h)\}_{w=1,h=1}^{W,H}$. The coordinate set of each position is a subset of $\mathcal{P}$, *i.e.*, $\mathcal{S}_{w,h} \in 2^{\mathcal{P}}$ where $2^{\mathcal{P}}$ is the power set of $\mathcal{P}$. Each coordinate set $\mathcal{S}_{w,h}$ defines a corresponding feature set $\mathbf{X}_{\mathcal{S}_{w,h}} = [\mathbf{x}_{w',h'}]_{(w',h') \in \mathcal{S}_{w,h}}$, and we abbreviate $\mathbf{X}_{\mathcal{S}_{w,h}}$ as $\mathfrak{X}_{w,h}$. Thus, $z_{w,h,d} = f_d(\mathbf{X}, \mathcal{S}_{w,h})$ can be rewritten as $z_{w,h,d} = f_d(\mathfrak{X}_{w,h})$. This means that, for two spatial positions (w_1,h_1) and (w_2,h_2), $\mathbf{z}_{w_1,h_1}$ can be impacted by $\mathbf{x}_{w_2,h_2}$ if and only if $(w_2,h_2) \in \mathcal{S}_{w_1,h_1}$, and vice versa. It is common knowledge that if two positions (w_1,h_1) and (w_2,h_2) are close in the image lattice, *i.e.*, $\|(w_1,h_1)-(w_2,h_2)\|_1$ is small[3], the relationship of their feature vectors is more likely to be tight. Therefore, we define each $\mathcal{S}_{w,h}$ to be a continuous region[4] that covers (w,h) itself.

We provide two ways of defining $\mathcal{S}_{w,h}$, both of which are based on a scale parameter K. The first one is named the *sliding* strategy, in which $\mathcal{S}_{w,h} = \{(w',h') \mid \|(w,h)-(w',h')\|_1 \leqslant T\}$, where $T = \sqrt{WH}/K$ is the threshold of distance. The second one is named the *regional* strategy, which partitions the image lattice into $K \times K$ equally-sized regions, and $\mathcal{S}_{w,h}$ is composed of all positions falling in the same region with it. The former is more flexible, *i.e.*, each position has a unique spatial region set, and so there are $W \times H$ different sets, while the latter reduces this number to K^2, which slightly reduces the computational costs (see Section 3.5).

[3] Constraining $\|(w_1,h_1)-(w_2,h_2)\|_1$ results in a square region which is more friendly in implementation than constraining $\|(w_1,h_1)-(w_2,h_2)\|_2$.

[4] By continuous we mean that $\mathcal{S}_{w,h}$ equals to the smallest convex hull that contains it, *i.e.*, there are no holes in this region.

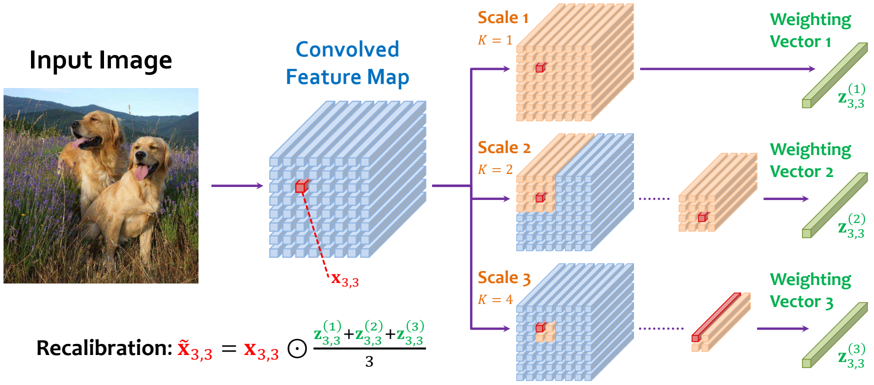

Fig. 1. Illustration of multi-scale spatially-asymmetric recalibration (MS-SAR). The feature vector for recalibration is marked in red, and the spatial coordinate sets at different scales are marked in yellow, and the weighting vectors are marked in green. For the first and second scales, for better visualization, we copy the neural responses used for recalibration. This figure is best viewed in color.

It remains to determine the form of the recalibration function $f_d(\mathfrak{X}_{w,h})$. The major consideration is to reduce the number of parameters to alleviate the risk of over-fitting, and reduce the computational costs (FLOPs) to prevent the network from being much slower. We borrow the idea of adding both spatial and channel bottlenecks for this purpose [13]. $\mathfrak{X}_{w,h}$ is first down-sampled into a single vector using average pooling, *i.e.*, $\mathbf{y}_{w,h} = \left|\mathcal{S}_{w,h}\right|^{-1} \sum_{(w,h)\in\mathcal{S}_{w,h}} \mathbf{x}_{w,h}$, and passed through two fully-connected layers: $z_{w,h,d} = \sigma_2[\boldsymbol{\Omega}_{2,d} \cdot \sigma_1[\boldsymbol{\Omega}_1 \cdot \mathbf{y}_{w,h}]]$. Here, both $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_{2,d}$ are learnable weight matrices, and $\sigma_1[\cdot]$ and $\sigma_2[\cdot]$ are activation functions which add non-linearity to the recalibration function. The dimension of $\boldsymbol{\Omega}_1$ is $D' \times D$ $(D' < D)$, and that of $\boldsymbol{\Omega}_{2,d}$ is $1 \times D'$. This idea is similar to using channel bottleneck to reduce computations [11]. $\sigma_1[\cdot]$ is a composite function of batch normalization [17] followed by ReLU activation [24], and $\sigma_2[\cdot]$ replaces ReLU with sigmoid so as to output a floating point number in $(0, 1)$.

We share $\boldsymbol{\Omega}_1$ over all $f_d(\mathfrak{X}_{w,h})$'s, but use an individual $\boldsymbol{\Omega}_{2,d}$ for each output channel. Let $\boldsymbol{\Omega}_2 = \left[\boldsymbol{\Omega}_{2,1}^{\top}; \ldots; \boldsymbol{\Omega}_{2,D}^{\top}\right]^{\top}$, and thus the recalibration function is:

$$\mathbf{z}_{w,h} = \mathbf{f}(\mathfrak{X}_{w,h}) = \sigma_2\left[\boldsymbol{\Omega}_2 \cdot \sigma_1\left[\boldsymbol{\Omega}_1 \cdot \frac{1}{\left|\mathcal{S}_{w,h}\right|} \cdot \sum_{(w,h)\in\mathcal{S}_{w,h}} \mathbf{x}_{w,h}\right]\right]. \qquad (1)$$

3.3 Multi-Scale Spatially Asymmetric Recalibration

In Eqn (1), the coordinate set $\mathcal{S}_{w,h}$ determines the region-of-interest (ROI) that can impact $\mathbf{z}_{w,h}$. There is the need of using different scales to evaluate the importance of each feature. We achieve this goal by defining multiple coordinate sets for each spatial position.

Let the total number of scales be L. For each $l = 1, 2, \ldots, L$, we define the scale factor $K^{(l)}$, construct the coordinate set $\mathcal{S}_{w,h}^{(l)}$ and the feature set $\mathfrak{X}_{w,h}^{(l)}$, and compute $\mathbf{z}_{w,h}^{(l)}$ using Eqn (1). The weights from different scales are averaged: $\mathbf{z}_{w,h} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{z}_{w,h}^{(l)}$. Using the matrix notation, we write **multi-scale spatially-asymmetric recalibration** (MS-SAR) as:

$$\tilde{\mathbf{X}}_{w,h} = \mathbf{X} \odot \mathbf{Z} = \mathbf{X} \odot \frac{1}{L}\sum_{l=1}^{L}\mathbf{Z}^{(l)}. \qquad (2)$$

The configuration of this an MS-SAR is denoted by $\mathcal{L} = \left\{K^{(l)}\right\}_{l=1}^{L}$. When $\mathcal{L} = \{1\}$, MS-SAR degenerates to the recalibration approach used in the Squeeze-and-Excitation Network (SENet) [13], which is single-scaled and spatially-symmetric, *i.e.*, each pair of spatial positions can impact each other, and $\mathbf{z}_{w,h}$ is the same at all positions. We will show in experiments that MS-SAR produces superior performance than this degenerated version.

3.4 Applications to Existing Building Blocks

MS-SAR can be applied to each convolutional layer individually. Here we consider two examples, which integrate MS-SAR into a residual block [11] and a densely-connected block [15], respectively. The modified blocks are shown in Figure 2. In a residual block, we only recalibrate the second convolutional layer, while in a densely-connected block, this operation is performed before each convolved feature vector is concatenated to the main feature vector.

Another difference lies in the input of the recalibration function. In the residual block, we simply use the convolved response map for "self recalibration", but in the densely-connected block, especially in the late stages, we note that the main vector is of a much higher dimensionality and thus contains multi-stage visual information. Therefore, we compute the recalibration function using the main vector. We name this option as *multi-stage recalibration*. In comparison to *single-stage recalibration* (input the convolved vector to the recalibration function), it requires more parameters as well as computations, but also leads to better classification performance (see Section 4.2).

3.5 Computational Costs

Let $\mathbf{X}$ be a $W \times H \times D$ cube, and the input of convolution also have D channels, then the number of parameters of convolution is $9D^2$ (assuming the convolution kernel size is 3×3). Given that MS-SAR is configured by $\mathcal{L} = \left\{K^{(l)}\right\}_{l=1}^{L}$, the learnable parameters come from two weight matrices $\boldsymbol{\Omega}_1$ $(D' \times D)$ and $\boldsymbol{\Omega}_2$ $(D \times D')$, and so there are $2DD'$ extra parameters for each scale, and $2LDD'$ for all L scales. We set $D' = D/L$ so that using multiple scales does not increase the total number of parameters.

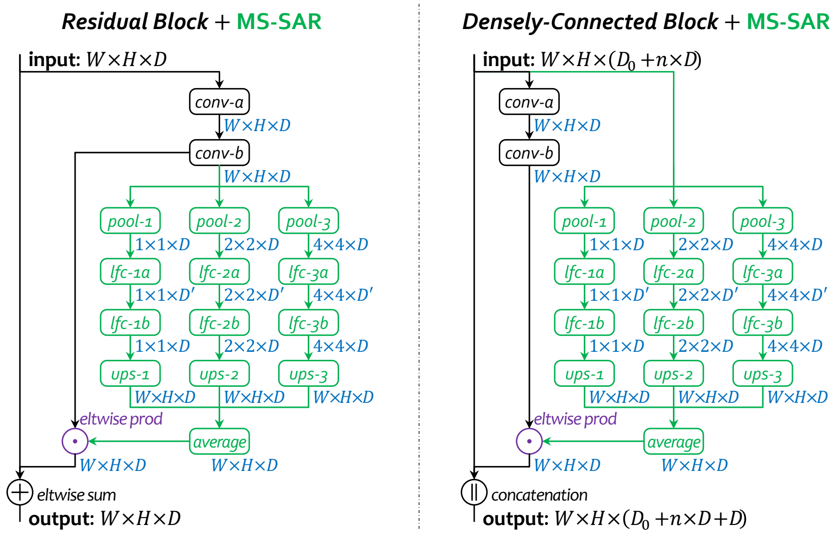

Fig. 2. Applying MS-SAR (green parts) to a residual block (left) or one single step in a densely-connected block (right). In both examples we set $\mathcal{L} = \{1, 2, 4\}$. Here, *pool* indicates a $\frac{W}{K} \times \frac{H}{K}$ regional pooling, *lfc* is a local fully-connected layer (1×1 convolution), and *ups* performs up-sampling by duplicating each element for $\frac{W}{K} \times \frac{H}{K}$ times. The feature map size is labeled for each cube. This figure is best viewed in color.

The extra computations (FLOPs) brought by MS-SAR is related to the strategy of defining the coordinate sets. We first consider the *sliding* strategy, in which each position (w, h) has a different feature set $\mathfrak{X}_{w,h}$. The spatial average pooling over the feature sets of all positions takes around WHD FLOPs[5]. Then, each D-dimensional vector $\mathbf{y}_{w,h}$ is passed through two matrix-vector multiplications, and the total FLOPs is $2WHDD'$. For the *regional* strategy, the difference lies in that the number of unique feature sets is $K^{(l)2}$ at the l-th scale. By sharing computations, the total FLOPs of the fully-connected layers is decreased to $2K^{(l)2}DD'$. For all L scales, the extra FLOPs is $2LWHDD'$ for the *sliding* strategy and $2DD'\sum_{l=1}^{L} K^{(l)2}$ for the *regional* strategy, respectively.

Note that in both ResNets and DenseNets, MS-SAR is applied to half of convolutional layers, and so the fractions of extra parameters and FLOPs are relatively small. We will report the detailed numbers in experiments.

[5] This is implemented by the idea of *partial sum*. For each channel, we compute $T_{w,h} = \sum_{w'=1}^{w} \sum_{h'=1}^{h} \sum_{d=1}^{D} x_{w',h',d}$ for each position (w, h) – using a gradual accumulation process, this takes WHD sum operations for all D channels. Then we have $\sum_{w=w_1}^{w_2} \sum_{h=h_1}^{h_2} \sum_{d=1}^{D} x_{w,h,d} = T_{w_2,h_2} - T_{w_1-1,h_2} - T_{w_2,h_1-1} + T_{w_1-1,h_1-1}$, which takes $O(WH)$ sum operations for all spatial position (w, h)'s.

Approach	C10	C100	Network	C10	C100	FLOPs	Params
Lee *et al.*, 2015 [21]	7.97	34.57	RN-20	8.61	31.87	40.8M	0.27M
He *et al.*, 2016 [11]	6.61	27.22	RN-20*	**7.61**	**31.09**	40.9M	0.28M
Huang *et al.*, 2016 [16]	5.23	24.58	RN-32	7.51	30.63	69.1M	0.46M
He *et al.*, 2016 [12]	4.62	22.71	RN-32*	**6.68**	**29.41**	69.3M	0.48M
Zagoruyko *et al.*, 2016 [42]	4.17	20.50	RN-56	6.97	29.07	125.7M	0.85M
Han *et al.*, 2017 [9]	3.48	17.01	RN-56*	**6.04**	**27.71**	126.0M	0.89M
Huang *et al.*, 2017 [14]	3.40	17.40	DN-100	4.67	22.45	252.5M	0.80M
Zhang *et al.*, 2017 [46]	3.25	19.25	DN-100*	**4.16**	**21.13**	253.3M	0.99M
Gastaldi *et al.*, 2017 [6]	2.86	15.85	DN-190	3.46	17.34	7.95G	25.8M
Zhang *et al.*, 2017 [44]	2.70	16.80	DN-190*	**3.32**	**16.92**	7.98G	32.7M

Table 1. Comparison of classification error rates (%) on the CIFAR10 and CIFAR100 datasets. The left three columns list several recent work, and the right part compares our approach with the baselines. "RN" and "DN" denotes "ResNet" and "DenseNet". An asterisk sign (*) indicates that MS-SAR is added. For all ResNets, the error rates are averaged from 3 individual runs. All FLOPs and numbers of parameters are computed on the experiments on CIFAR10. The difference in these numbers between the CIFAR10 and CIFAR100 experiments are ignorable.

4 Experiments

4.1 The CIFAR Datasets

We first evaluate MS-SAR on the CIFAR datasets [18] which contain tiny RGB images with a fixed spatial resolution of 32×32. There are two subsets with 10 and 100 object classes, referred to as CIFAR10 and CIFAR100, respectively. Each set has 50,000 training samples and 10,000 testing samples, both of which are evenly distributed over all (10 or 100) classes.

We choose different baseline network architectures, including the deep residual networks (ResNets) [11] with 20, 32 and 56 layers and the densely-connected networks (DenseNets) [15] with 100 and 190 layers. MS-SAR is applied to *each* residual block and densely-connected block, as illustrated in Figure 2. We choose the *regional* strategy to construct coordinate sets, use $\mathcal{L} = \{1, 2, 4\}$ and set $D' = D/3$. For other options, see ablation studies in the next subsection.

We follow the conventions to train these networks from scratch. The standard SGD with a weight decay of 0.0001 and a Nesterov momentum of 0.9 are used. In the ResNets, we train the network for 160 epochs with mini-batch size of 128. The base learning rate is 0.1, and is divided by 10 after 80 and 120 epochs. In the DenseNets, we train the network for 300 epochs with a mini-batch size of 64. The base learning rate is 0.1, and is divided by 10 after 150 and 225 epochs. Adding MS-SAR does not require any of these settings to be modified. In the training process, the standard data-augmentation is used, *i.e.*, each image is padded with a 4-pixel margin on each of the four sides. In the enlarged 40×40 image, a subregion with 32×32 pixels is randomly cropped and flipped with a probability of 0.5. No augmentation is used at the testing stage.

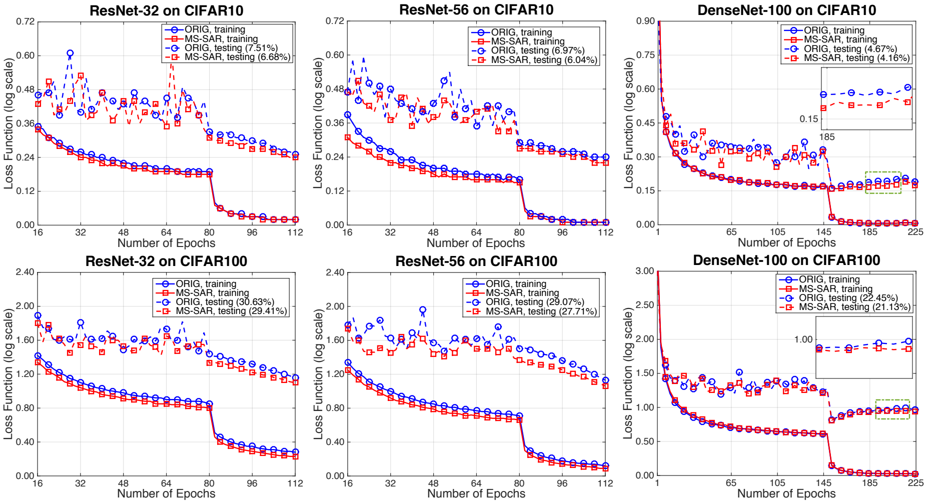

Fig. 3. The curves of different networks, with and without MS-SAR. All the curves on ResNet-32 and ResNet-56 are averaged over 3 individual runs.

Classification results are summarized in Table 1. One can observe that MS-SAR improves the baseline classification accuracy consistently and significantly. In particular, in terms of the relative drop in error rates, almost all these numbers are higher than 10% on CIFAR10 (except for DenseNet-190), and higher than 4% on CIFAR100 (except for ResNet-20 and DenseNet-190). The highest drop is over 10% on CIFAR10 and over 5% on CIFAR100. We note that these improvements are produced at the price of higher model complexities. The additional computational costs are very small for both the ResNets (*e.g*, $\sim 0.3\%$ extra FLOPs) and DenseNets (*e.g*, $\sim 0.3\%$ and $\sim 0.4\%$ extra FLOPs for DenseNet-100 and DenseNet-190, respectively), and the fractions of extra parameters are moderate ($\sim 5\%$ for the ResNets and $\sim 25\%$ for the DenseNets, respectively).

We also compare our results with the state-of-the-arts (listed in the left part of Table 1). Although some recent approaches reported much higher accuracies in the CIFAR datasets, we point out that they often used larger spatial resolutions [9], complicated network modules [46] or complicated regularization methods [6][44], and thus the results are not directly comparable to ours. In addition, we believe that MS-SAR can be applied to these networks towards better classification performance.

In Figure 3, we plot the training/testing curves of different networks on the CIFAR datasets. We find that MS-SAR effectively decreases the testing losses (and consequently, error rates) in all cases. On CIFAR10, due to the simplicity of the recognition task (10 classes), the training losses of both approaches, with and without MS-SAR, are very close to 0, but MS-SAR produces lower testing losses, giving evidence for its ability to alleviate over-fitting.

Scale			ResNet-56			DenseNet-100			
1	2	4	C10 (±std)	C100 (±std)	FLOPs	C10	C100	FLOPs	Params
			6.97 ± 0.05	29.07 ± 0.14	125.7M	4.67	22.45	252.5M	0.80M
✓			6.80 ± 0.06	28.99 ± 0.15	125.7M	4.45	21.83	252.6M	0.99M
	✓		6.55 ± 0.05	28.31 ± 0.17	125.8M	4.35	21.33	253.0M	0.99M
		✓	6.48 ± 0.06	28.74 ± 0.18	126.3M	4.39	21.79	254.3M	0.99M
✓	✓		6.38 ± 0.07	28.28 ± 0.19	125.8M	4.29	21.42	252.8M	0.99M
✓		✓	6.11 ± 0.14	28.05 ± 0.22	126.0M	4.32	21.27	253.5M	0.99M
	✓	✓	6.35 ± 0.09	28.87 ± 0.27	126.1M	4.33	21.23	253.7M	0.99M
✓	✓	✓	$\mathbf{6.04 \pm 0.11}$	$\mathbf{27.71 \pm 0.21}$	126.0M	**4.06**	**21.13**	253.3M	0.99M

Table 2. Comparison of classification error rates (%) on the CIFAR10 and CIFAR100 datasets with different scale combinations. Other specifications remain the same as in Figure 1. All results of ResNet-56 are averaged over 3 individual runs. See Section 3.5 for the reason that different scale configurations have the same number of parameters.

4.2 Ablation Study and Analysis

We first investigate the impacts of incorporating multi-scale visual information. To this end, we set $\mathcal{L}$ to be a non-empty subset of $\{1, 2, 4\}$ (7 possibilities), and summarize the results in Table 2. Compared with using a single scale, incorporating multi-scale information often leads to better classification performance (the only exception is that on DenseNet-100, $\mathcal{L} = \{2, 4\}$ works worse than $\mathcal{L} = \{2\}$, which may be caused by random noise as DenseNet-100 experiments are performed only once). Combining all three scales is always produces the best recognition performance. Provided that the extra computational costs brought by multi-scale recalibration are almost ignorable, we will use $\mathcal{L} = \{1, 2, 4\}$ in all the remaining experiments.

Next, we compare the two ways of defining coordinate sets (*sliding* vs. *regional*, see Section 3.2). In the experiments on CIFAR100, in both ResNets and DenseNets, the *regional* strategy outperforms the *sliding* strategy by $\sim 0.2\%$. The *training* accuracy using the *sliding* strategy is also decreased, giving evidence that it is less capable of fitting training data. This reveals that, although spatial asymmetry is a nice property, its degree of freedom should be controlled, so that MS-SAR, containing a limited number of parameters, does not need to fit an over-complicated distribution. Considering that the *regional* strategy requires fewer computational costs (see Section 3.5), we set it to be the default option.

Finally, we compare the *single-level* and *multi-level* recalibration methods on DenseNet-100. Detailed descriptions are in Section 3.4. Note that this is independent of the comparison between *multi-scale* and *single-scale* methods – they work on the spatial domain and the channel domain, and are complementary to each other. In the 100-layer DenseNet, *multi-level* recalibration produces 4.06% and 21.13% error rates on CIFAR10 and CIFAR100, and these numbers are 4.45% and 21.83% for *single-level* recalibration, respectively. *Multi-level* recalibration reduces the relative errors by 7.77% and 5.12%, at the price of 23.75% extra parameters and 0.3% additional FLOPs.

Approach	Scale			Top-1	Top-5	FLOPs	Params
	1	2	4				
ResNet-18				30.50	11.07	1.81G	10.9M
ResNet-18+SE	✓			29.78	10.27	1.81G	13.8M
ResNet-18+MS-SAR	✓	✓	✓	**29.43**	**10.19**	1.81G	13.8M
ResNet-34				27.02	8.77	3.66G	21.7M
ResNet-34+SE	✓			26.67	8.43	3.66G	27.3M
ResNet-34+MS-SAR	✓	✓	✓	**26.15**	**8.35**	3.67G	27.4M
ResNeXt-50				22.20	6.12	3.86G	25.0M
ResNeXt-50+SE	✓			21.95	5.93	3.87G	27.5M
ResNeXt-50+MS-SAR	✓	✓	✓	**21.64**	**5.78**	3.89G	27.6M

Table 3. Comparison of top-1 and top-5 classification error rates (%) produced by different recalibration approaches (none, SE and MS-SAR) on the ILSVRC2012 dataset. All these numbers are based on our own implementation. See Section 3.5 for the reason that different scale configurations have the same number of parameters.

4.3 The ILSVRC2012 Dataset

The ILSVRC2012 dataset [29] is a subset of the ImageNet database [4], created for a large-scale visual recognition competition. It contains 1,000 categories located at different levels of the WordNet hierarchy. The training and testing sets have $\sim$ 1.3M and 50K images, roughly uniformly distributed over all classes.

The baseline network architectures include two ResNets [11] with 18 and 34 layers, and a ResNeXt [40] with 50 layers. We also compare with the Squeeze-and-Excitation (SE) module [13], which is a special case of our approach ($\mathcal{L} = \{1\}$: single-scale and spatially-symmetric). As illustrated in Figure 2, both SE and MS-SAR modules are appended after each residual block.

All these networks are trained from scratch. We follow [13] in configuring the following parameters. SGD with a weight decay of 0.0001 and a Nesterov momentum of 0.9 is used. There are a total of 100 epochs in the training process, and the mini-batch size is 1024. The learning rate starts with 0.6, and is divided by 10 after 30, 60 and 90 epochs. Again, adding MS-SAR does not require any of these settings to be modified. In the training process, we apply a series of data-augmentation techniques, including rescaling and cropping the image, randomly mirroring and rotating (slightly) the image, changing its aspect ratio and performing pixel jittering, which is same with SENet[13]. In the testing process, we use the standard single-center-crop on each image.

Results are summarized in Table 3. In all cases, MS-SAR works better than the baseline (no recalibration) and SE (single-scale spatially-symmetric recalibration). For example, based on ResNeXt-50, MS-SAR reduces the top-5 error of the baseline by an absolute value of 0.34% or a relative value of 5.56%, using $\sim$ 1% extra FLOPs and 10% extra parameters. On top of SE, the error rate drops are 0.15% (absolute) and 2.53% (relative) and the extra FLOPs and parameters are merely $\sim$ 0.5% and $\sim$ 0.4%, respectively. The training/testing curves in Figure 4 show similar phenomena as in CIFAR experiments.

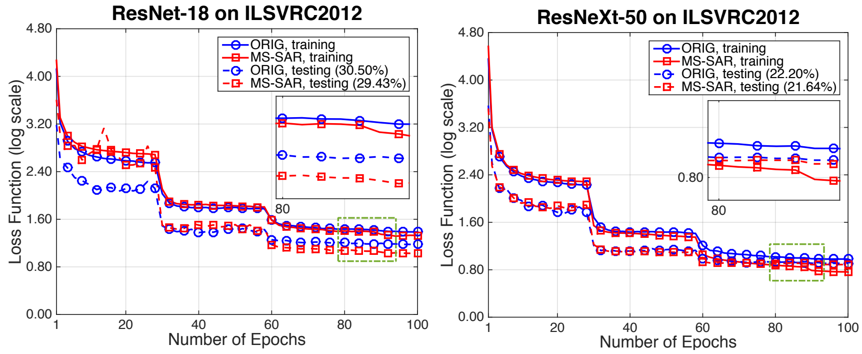

Fig. 4. The curves of different networks with and without MS-SAR on the ILSVRC2012 dataset. We zoom-in on a small part of each curve for better visualization.

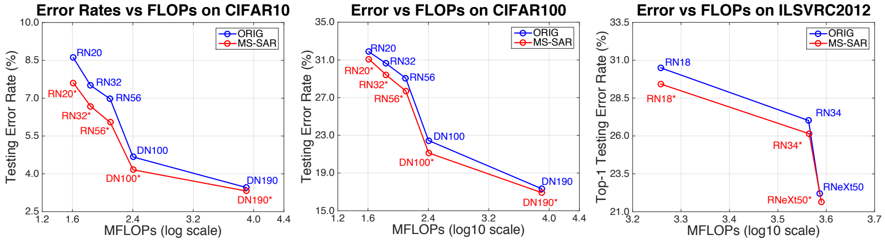

Fig. 5. The relationship between classification accuracy and computation (in FLOPs) on three datasets. RN, DN and RNeXt denote ResNet, DenseNet and ResNeXt, respectively. An asterisk sign (*) indicates that MS-SAR is added.

We also investigate the relationship between classification accuracy and computation on these three datasets. In Figure 5, we plot the testing error as the function of FLOPs, which reveals the trend that MS-SAR can achieve higher recognition accuracy under the same computational complexity.

Last but not least, we visualize spatial weights added by the MS-SAR layer in Figure 6. We present two input images containing an object (a *bird*) and a scene (a *mountain*), respectively. One can observe that, in comparison to the 1×1 weight, both 2×2 and 4×4 weights are more flexible to capture semantically meaningful regions and add higher weights. In each layer, we see some filters focus on the foreground, *e.g.*, the characteristic patterns of the *bird* and the *mountain*, while some others focus on the background, *e.g.*, the tree branch or the sky. High-level layers have low-resolution feature maps, but this property is preserved. We argue that it is the spatial asymmetry that allows the recalibration module to capture different visual information (foreground vs. background), which allows the weighted neural response ($x_{w,h,d}$) to be *dependent* to its spatial location (w, h).

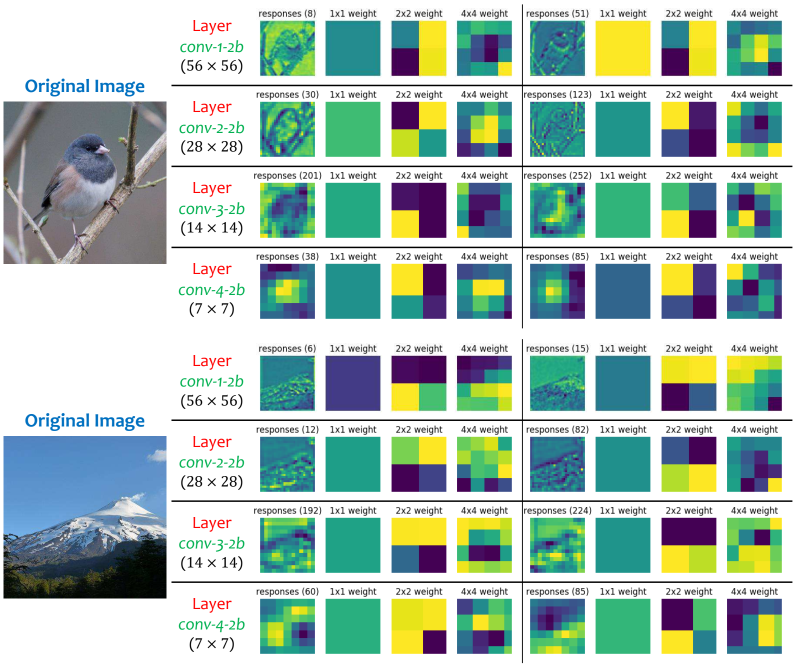

Fig. 6. Visualization of weights added by MS-SAR (best viewed in color, adjusted to the spatial resolution in each layer) to a 18-layer ResNet. The response/weight is higher if the color is closer to yellow. Each number in parentheses indicates the filter index.

5 Conclusions

In this paper, we present a module named MS-SAR for image classification. This is aimed at assigning eacg convolutional layer with the ability to incorporate spatial contexts to "recalibrate" neural responses, *i.e.*, summarizing regional information into an importance factor and multiplying it to the original response. We implement each recalibration function as the combination of a multi-scale pooling operation in the spatial domain and a linear model in the channel domain. Experiments on CIFAR and ILSVRC2012 demonstrate the superior performance of MS-SAR over several baseline network architectures.

Our work delivers two messages. First, it is not the best choice to rely on a gradually increasing receptive field (via local convolution, pooling or down-sampling) to capture spatial information – MS-SAR is a light-weighted yet specifically designed module which deals with this issue more efficiently. Second, there exists a tradeoff between diversity and simplicity – this is why *regional* pooling works better than *sliding* pooling. In its current form, MS-SAR is able to add a weight factor to each neural response (unary or linear terms), but unable to explicitly model the co-occurrence of multiple features (binary or higher-order terms). We leave this topic for future research.

References

1. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In: International Conference on Learning Representations (2016)
2. Chen, L.C., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. In: Computer Vision and Pattern Recognition (2016)
3. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. In: Advances in Neural Information Processing Systems (2017)
4. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition (2009)
5. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International Conference on Machine Learning (2014)
6. Gastaldi, X.: Shake-shake regularization. arXiv preprint arXiv:1705.07485 (2017)
7. Girshick, R.: Fast r-cnn. In: Computer Vision and Pattern Recognition (2015)
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition (2014)
9. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: Computer Vision and Pattern Recognition (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European Conference on Computer Vision (2014)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (2016)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
14. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: Train 1, get m for free. In: International Conference on Learning Representations (2017)
15. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Computer Vision and Pattern Recognition (2017)
16. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision (2016)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (2015)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
19. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

21. Lee, C., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial Intelligence and Statistics (2015)
22. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.: Microsoft coco: Common objects in context. In: European conference on computer vision (2014)
23. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Computer Vision and Pattern Recognition (2015)
24. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning (2010)
25. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision (2016)
26. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: International Conference on Computer Vision (2015)
27. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Computer Vision and Pattern Recognition (2014)
28. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (2015)
29. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
30. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
32. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
33. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (2015)
35. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Computer Vision and Pattern Recognition (2014)
36. Wang, Y., Xie, L., Liu, C., Qiao, S., Zhang, Y., Zhang, W., Tian, Q., Yuille, A.: Sort: Second-order response transform for visual recognition. In: International Conference on Computer Vision (2017)
37. Xie, L., Tian, Q., Flynn, J., Wang, J., Yuille, A.: Geometric neural phrase pooling: Modeling the spatial co-occurrence of neurons. In: European Conference on Computer Vision (2016)
38. Xie, L., Yuille, A.: Genetic cnn. In: International Conference on Computer Vision (2017)
39. Xie, L., Zheng, L., Wang, J., Yuille, A., Tian, Q.: Interactive: Inter-layer activeness propagation. In: Computer Vision and Pattern Recognition (2016)
40. Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Computer Vision and Pattern Recognition (2017)
41. Xie, S., Tu, Z.: Holistically-nested edge detection. In: International Conference on Computer Vision (2015)

42. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
43. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (2014)
44. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
45. Zhang, N., Donahue, J., Girshick, R., Darrell, T.: Part-based r-cnns for fine-grained category detection. In: European Conference on Computer Vision (2014)
46. Zhang, T., Qi, G.J., Xiao, B., Wang, J.: Interleaved group convolutions. In: Computer Vision and Pattern Recognition (2017)
47. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017)