ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1760

Stochastic Area Pooling for Generic Convolutional Neural Network

Zhidong Deng[1] and **Zhenyang Wang**[2] and **Shiyao Wang**[3]

Abstract. This paper proposes a novel SAPNet model that incorporates a stochastic area pooling (SAP) method with a generic stacked T-shaped CNN architecture. In our SAP method, pooling area is randomly transformed and max pooling operation is then conducted on such areas, which are no longer regular identical fixed upright squares. It can be viewed as feature-level augmentation, substantially reducing model parameters while keeping generalization ability of CNN almost unchanged. Furthermore, we present a generic CNN architecture that structurally resembles three stacked T-shaped cubes. In such architecture, the number of kernels in convolutional layer preceding any pooling layer is doubled and all learnable weight layers are combined with batch normalization and dropout with a small ratio. Finally, on CIFAR-10, CIFAR-100, MNIST, and SVHN datasets, the experimental results show that our SAPNet requires fewer parameters than regular CNN models and still achieves superior recognition performances for all the four benchmarks.

1 Introduction

Data augmentation is a simple and useful way to improve recognition and generalization capabilities of CNN. It can help expand data space and further enhance diversity of input images. In this paper, we take advantage of such idea to feature-level augmentation to explore expansion on feature map space. Accordingly, we propose a novel stochastic area pooling (SAP) to improve feature representation capabilities of CNN. Instead of pooling on fixed pooling areas, pooling areas in our SAP method could be randomly translated, scaled, and rotated with a fluctuation. Jaderberg *et al*. [3] introduce similar idea in their spatial transformer network, which is to transform the input feature map. The difference is that the transformations in spatial transformer network are a deterministic function of the input and the transformation parameters and are explicitly optimized over, while the transformations in our SAP method are randomly drawn from pre-specified distributions that are not optimized over. In particular, Jaderberg *et al*.aim at generating scale and rotation invariance features by introducing an addition learning-based spatial transformer network, which requires a large amount of parameters and computational cost. SAP, however, plays the role of feature-level augmentation, having only a little extra computational burden.

Meanwhile, aiming to simplify the design procedure of new convolutional models, we present a generic CNN architecture that structurally looks like three stacked T-shaped cubes. In such architecture, the number of kernels in convolutional layer just preceding any pooling layer is doubled to overcome representational bottleneck [8]. All learnable weight layers are embedded with regularization of batch normalization, ReLU, and dropout with a small ratio of 0.1. Actually, all SAPNets with different convolutional kernel sizes have the same structures and similar parameter settings, which are demonstrated to be stable, reliable, and efficient.

[1] State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science, Tsinghua University, Beijing 100084, China, email: michael@tsinghua.edu.cn
[2] Tsinghua University, email: crazycry2010@gmail.com
[3] Tsinghua University, email: sy-wang14@mails.tsinghua.edu.cn

2 SAPNet Method

2.1 SAP Layer

SAP consists of the two consecutive steps: area selection and pooling operation. As the first step, a pooling area is generated using random affine transformation. Second, regular pooling operations such as max or average pooling are conducted on such randomly transformed areas, thus giving rise to output of SAP layer.

Instead of regular pooling on fixed square, the random pooling area of SAP is transformed by an affine mapping as shown in Figure 1(a).Suppose that θ denotes angle of rotation, $[c_x, c_y]$ coordinates of a center point, $[o_x, o_y]$ a shift vector, and $[s_x, s_y]$ scaling factors. Consequently, we form a 7-tuple of parameters $\{\theta, c_x, c_y, o_x, o_y, s_x, s_y\}$ so as to express such an affine transform, i.e.,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \cos\theta & s_y \sin\theta & t_x \\ -s_y \sin\theta & s_x \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

$$t_x = c_x(1 - s_x \cos\theta) - c_y s_y \sin\theta + o_x \tag{2}$$

$$t_y = c_x(1 + s_y \sin\theta) - c_y s_x \cos\theta + o_y \tag{3}$$

Actually, pooling area in SAP is produced by assigning a random distribution to each parameter. The Gaussian distribution that each parameter must satisfy is given below,

$$\theta \sim N(0, \sigma_\theta), \tag{4}$$

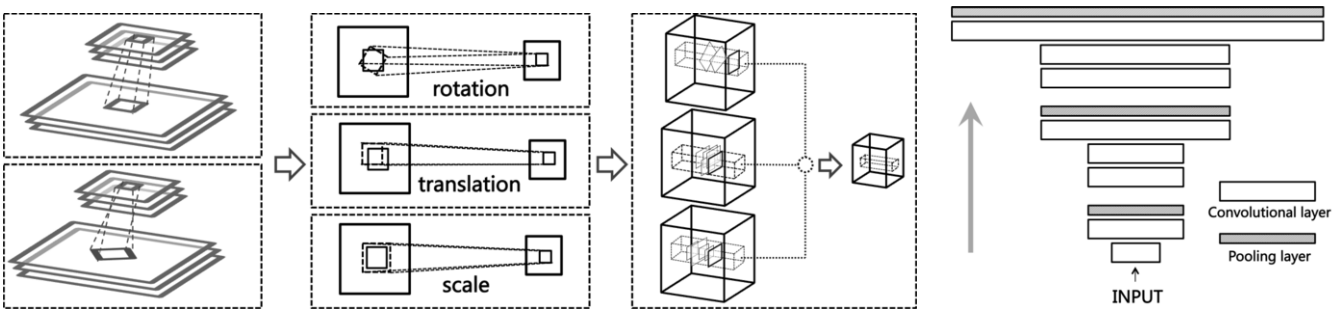
$$c_x, c_y \sim N(0.5d, \sigma_c) \tag{5}$$

$$s_x, s_y \sim N(1, \sigma_s) \tag{6}$$

where standard deviations σ_θ, σ_c and σ_s are called hyper-parameters, which should be pre-specified before training, and d indicates the height or width of incoming feature map.

Note that pixels within random pooling areas are usually transformed onto non-integral boundaries. Thus bilinear interpolation is required before pooling operation.

During every forward pass, a set of different affine transformation parameters are randomly produced via Equations (4), (5), and (6).



(a) Pooling areas of SAP randomly transformed (b) Three stacked T-shaped cubes

Figure 1. Schematic diagram of stochastic area pooling for generic CNN architecture with three stacked T-shaped cubes.

After that, these transformation parameters or the resulting pooling areas in the SAP layer remain unchanged and are not optimized over during either forward pass or back-propagation. Sequentially, max pooling is then done on such areas. Apparently, our SAP layer has the same forward and backward pass as that for regular CNNs with max pooling except for transformed pooling area.

In the test phase, we set the standard deviations of parameters $\sigma_\theta = 0$, $\sigma_c = 0$ and $\sigma_s = 0$ for the SAP layers, which means that we employ the same fixed upright pooling squares as that of regular pooling method.

2.2 Generic Stacked T-shaped CNN Architecture

In addition to pooling strategies, design of CNN architecture is also crucial to improvements in capabilities of hierarchical feature representation. Basically, this used to be an empirical trick. In this paper, we deliberately design a generic CNN architecture that structurally looks like three stacked T-shaped cubes (in Figure 1(b)), together with two SAP layers as separator. Note that the number of kernels in the convolutional layer preceding each pooling layer is always doubly expanded. We call such a general model as SAPNet. Specifically, every SAPNets comprises three ensembles, each of ensembles containing three convolutional layers, and one softmax layer. A total of 9 convolutional layers are separated by two SAP layers and one global average pooling layer.

In SAPNet, all convolutional layers have small receptive field of 3×3 with padding of 1 so as to preserve spatial resolution. In two SAP layers, we take 2×2 initial pooling areas with stride of 2. Following the last convolutional layer, one global average pooling is adopted. Similarly, we exploit a softmax layer as final output. Additionally, all learnable weight layers are embedded with regularization of batch normalization (BN), ReLU nonlinearity, and dropout with a small ratio of 0.1.

3 Experimental Results

We evaluated our approach on CIFAR-10, CIFAR-100, MINST, and SVHN datasets. The experimental results (Table 1) show that our SAPNet requires fewer parameters than regular CNN models. Our SAPNet yields a state of the art test error of 5.57% on CIFAR-10 with data augmentation. On CIFAR-100, the SAPNet-64 achieves a test error of 27.59%. On MNIST, the SAPNet with 0.76 million parameters receives the best result of 0.29%. Our SAPNet obtains a test error of 1.71% on SVHN benchmark, which is ranked second in all the existing machine learning models.

Table 1. Comparison with existing models on CIFAR-10, CIFAR-10 with data augmentation, CIFAR-100, MNIST, and SVHN.

Model	CIFAR-10	CIFAR-10+	CIFAR-100	MNIST	SVHN
Maxout [2]	11.68	9.38	38.57	0.45	2.47
NIN [7]	10.41	8.81	35.68	0.47	2.35
DSN [5]	9.69	7.97	34.57	0.39	1.92
RCNN [6]	8.69	7.09	31.75	0.31	1.77
Tree+Max-Avg [4]	7.62	6.05	32.37	0.31	**1.69**
ELU-Network [1]	6.55	-	**24.28**	-	-
SAPNet	**6.36**	**5.57**	27.59	**0.29**	1.71

ACKNOWLEDGEMENTS

The authors would like to be grateful to the anonymous reviewers for their valuable comments that considerably contributed to improve this paper. This work was supported in part by the National Science Foundation of China (NSFC) under Grant Nos. 91420106, 90820305, and 60775040, and by the National High-Tech R&D Program of China under Grant No. 2012AA041402.

REFERENCES

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, 'Fast and accurate deep network learning by exponential linear units (elus)', *arXiv preprint arXiv:1511.07289*, (2015).
[2] Ian J Goodfellow, David Warde-farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, 'Maxout networks', in *ICML*, pp. 1319–1327, (2013).
[3] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., 'Spatial transformer networks', in *Advances in Neural Information Processing Systems*, pp. 2008–2016, (2015).
[4] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu, 'Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree', *arXiv preprint arXiv:1509.08985*, (2015).
[5] ChenYu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu, 'Deeply-supervised nets', in *AISTATS*, pp. 562–570, (2015).
[6] Ming Liang and Xiaolin Hu, 'Recurrent convolutional neural network for object recognition', in *CVPR*, pp. 3367–3375, (2015).
[7] Min Lin, Qiang Chen, and Shuicheng Yan, 'Network in network', in *ICLR*, (2014).
[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna, 'Rethinking the inception architecture for computer vision', *arXiv preprint arXiv:1512.00567*, (2015).