Geometric Neural Phrase Pooling: Modeling the Spatial Co-occurrence of Neurons

Lingxi Xie[1], Qi Tian[2], John Flynn[3], Jingdong Wang[4], Alan Yuille[5]

[1,5]Center for Imaging Science, The Johns Hopkins University, Baltimore, MD, USA
[2]Department of Computer Science, University of Texas at San Antonio, TX, USA
[3]Department of Statistics, University of California, Los Angeles, CA, USA
[4]Microsoft Research, Beijing, China
[1]198808xc@gmail.com [2]qitian@cs.utsa.edu
[3]john_flynn@mac.com [4]jingdw@microsoft.com [5]alan.l.yuille@gmail.com
Codes: http://bigml.cs.tsinghua.edu.cn/~lingxi/Projects/GNPP.html

Abstract. Deep Convolutional Neural Networks (CNNs) are playing important roles in state-of-the-art visual recognition. This paper focuses on modeling the spatial co-occurrence of neuron responses, which is less studied in the previous work. For this, we consider the neurons in the hidden layer as *neural words*, and construct a set of *geometric neural phrases* on top of them. The idea that grouping neural words into neural phrases is borrowed from the Bag-of-Visual-Words (BoVW) model. Next, the **Geometric Neural Phrase Pooling** (GNPP) algorithm is proposed to efficiently encode these neural phrases. GNPP acts as a new type of hidden layer, which punishes the isolated neuron responses after convolution, and can be inserted into a CNN model with little extra computational overhead. Experimental results show that GNPP produces significant and consistent accuracy gain in image classification.

Keywords: Image Classification, Convolutional Neural Networks, Spatial Co-occurrence of Neurons, Geometric Neural Phrase Pooling

1 Introduction

We have witnessed a significant revolution in computer vision brought by the deep Convolutional Neural Networks (CNNs). With powerful computational resources (*e.g.*, GPUs) and a large amount of labeled training data (*e.g.*, [1]), a hierarchical structure containing different levels of visual concepts is constructed and trained [2] to produce impressive performance on large-scale visual recognition tasks [3]. A pre-trained deep network is also capable of generating deep features for various tasks, such as image classification [4][5], image retrieval [6][7] and object detection [8][9].

CNN is composed of several stacked layers, each of which contains a number of neurons. We argue that modeling the co-occurrence of neuron responses is important, whereas less studied in the previous work. For this, we define a set of *geometric neural phrases* on the basis of the hidden neurons, and propose

the **Geometric Neural Phrase Pooling** (GNPP) algorithm to encode them efficiently. GNPP can be regarded as a new type of layer, and inserted into a network with little computational overhead (*e.g.*, 1.29% and 2.52% extra time and memory costs in the experiments on **ImageNet**). We explain the behavior of GNPP by noting that it punishes the isolated neuron responses, and that the isolated responses are often less reliable than clustered ones, especially in the high-level network layers. Experimental results show that adding GNPP layers boosts image classification accuracy significantly and consistently. Later, we will discuss the benefits brought by the GNPP layer from different points of view, showing that GNPP produces better internal representation, builds latent connections, and accelerates the network training process.

The remainder of this paper is organized as follows. Section 2 briefly introduces related work. Section 3 introduces the GNPP layer, and Section 4 shows experimental results. We discuss the benefits brought by adding GNPP layers in Section 5. Finally, we conclude this work in Section 6.

2 Related Work

2.1 The Bag-of-Visual-Words Model

The Bag-of-Visual-Words (BoVW) model [10] represents each image as a high-dimensional vector. It typically consists of three stages, *i.e.*, descriptor extraction, feature encoding and feature aggregation.

Due to the limited descriptive ability of raw pixels, handcrafted descriptors such as SIFT [11], HOG [12] or other variants [13] are extracted. The set of local descriptors on an image is denoted as $\mathcal{D} = \{(\mathbf{d}_m, \mathbf{l}_m)\}_{m=1}^{M}$, where M is the number of descriptors, $\mathbf{d}_m$ is the description vector and $\mathbf{l}_m$ is the 2D location of the m-th word. A visual vocabulary or codebook is then built to capture the data distribution in feature space. The codebook is a set of codewords: $\mathcal{B} = \{\mathbf{c}_b\}_{b=1}^{B}$, in which B is the codebook size and each codeword has the same dimension with the descriptors. Each descriptor $\mathbf{d}_m$ is then quantized onto the codebook as a visual word $\mathbf{f}_m \in \mathbb{R}_{\geqslant 0}^{B}$. Effective feature quantization algorithms include sparse coding [14][15] and high-dimensional encoding [16][17][18]. $\mathcal{F} = \{(\mathbf{f}_m, \mathbf{l}_m)\}_{m=1}^{M}$ is the set of visual words. Finally, these words are aggregated as an image-level representation vector [19][20]. These Image-level vectors are then normalized and fed into machine learning algorithms [21] for training and testing, or used in some training-free image classification algorithms [22][7].

2.2 Geometric Phrase Pooling

The basic unit in the BoVW model is a *visual word*, *i.e.*, a quantized local descriptor. Dealing with individual visual words does not consider the spatial co-occurrence of visual features. To this end, researchers propose *visual phrase* [23][24] as a mid-level data structure connecting low-level descriptors and high-level visual concepts [25]. A visual phrase is often defined as a group of

neighboring visual words [25][26]. It can be used to filter out the false matches in object retrieval [24][27], or improve the descriptive ability of visual features for image classification [26][28].

Geometric Phrase Pooling (GPP) [26] is an efficient algorithm for extracting and encoding visual phrases. GPP starts from constructing, for each visual word, a *geometric visual phrase*, which is a group of visual words: $\mathcal{G}_m = (\mathbf{f}_m, \mathbf{l}_m) \cup \left\{ \left(\mathbf{f}_m^{(k)}, \mathbf{l}_m^{(k)} \right) \right\}_{k=1}^{K}$. In $\mathcal{G}_m$, $(\mathbf{f}_m, \mathbf{l}_m)$ is the *central word*, and all the other K words are *side words*, located in a small neighborhood $\mathcal{N}_m$ of the central position $\mathbf{l}_m$. GPP encodes each geometric visual phrase $\mathcal{G}_m$ by adding the maximal response of the side words to the central word: $\mathbf{p}_m = \mathbf{f}_m + \max_{k=1}^{K} \left\{ s_m^{(k)} \times \mathbf{f}_m^{(k)} \right\}$, where $\max_{k=1}^{K} \{\cdot\}$ denotes dimension-wise maximization. Note that the central word is not included in the maximization. $s_m^{(k)}$ is the smoothing weight of the k-th side word in $\mathcal{G}_m$. Most often, $s_m^{(k)}$ is determined by the Euclidean distance between $\mathbf{l}_m^{(k)}$ and $\mathbf{l}_m$, *e.g.*, $s_m^{(k)} = \exp\left\{ -\tau \times \left\| \mathbf{l}_m - \mathbf{l}_m^{(k)} \right\|_2 \right\}$, where $\tau > 0$ is the predefined smoothing parameter. Note that, at least in theory, the GPP algorithm can be applied to any data with a spatial attribute.

2.3 Convolutional Neural Networks

The Convolutional Neural Network (CNN) serves as a hierarchical model for large-scale visual recognition. It is based on the observation that a network with enough neurons is able to fit any complicated data distribution. In past years, neural networks were shown effective for simple recognition tasks [29]. More recently, the availability of large-scale training data (*e.g.*, ImageNet [1]) and powerful GPUs make it possible to train deep CNNs [2] which significantly outperform BoVW models. A CNN is composed of several stacked layers. In each of them, responses from the previous layer are convoluted with a filter bank and activated by a differentiable non-linearity. Hence, a CNN can be considered as a composite function, which is trained by back-propagating error signals defined by the difference between supervision and prediction at the top layer. Efficient methods were proposed to help CNNs converge faster and prevent over-fitting, such as ReLU activation [2], batch normalization [30] and regularization [31][32]. It is believed that deeper networks produce better recognition results [33][34][35].

The intermediate responses of CNNs, *i.e.*, the so-called deep features, serve as effective image descriptions [5], or a set of latent visual attributes [36]. They can be used for various types of vision tasks, including image classification [4][37], image retrieval [6][7] and object detection [8]. A discussion of how different CNN configurations impact deep feature performance is available in [38].

3 Geometric Neural Phrase Pooling

This section presents the Geometric Neural Phrase Pooling (GNPP) algorithm and its application to improve the CNN model.

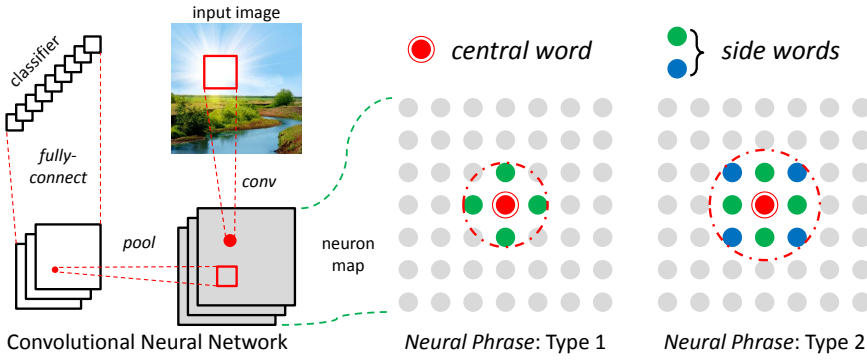

Fig. 1: Left: the architecture of a toy CNN model. A geometric neural phrase is defined on the basis of a set of neural words. Right: two types of neighborhood used in this work (best viewed in color PDF). The green side words are weighted by σ and the blue ones by σ^2, where σ is the smoothing parameter.

3.1 The GNPP Layer

We start with a hidden layer $\mathbf{X}$ in the CNN model. $\mathbf{X}$ is a 3D neuron cube with $W \times H \times D$ neurons, where W, H and D are the width, height and depth of the cube. The response of each neuron corresponds to the inner-product of a local patch in the previous layer and a filter (convolutional kernel). We naturally consider the data as a set of D-dimensional *visual words* indexed over a 2D spatial domain. We denote the set as $\mathcal{X} = \{\mathbf{x}_{w,h}\}_{w=1,h=1}^{W,H}$, in which $\mathbf{x}_{w,h} \in \mathbb{R}^D_{\geqslant 0}$ for each w and h. The spatial domain coordinate (w, h) is not the same as the pixel coordinate (a, b) in the original image, but they are linearly corresponded.

A *geometric neural phrase* is defined as $\mathcal{G}_{w,h} = \{\mathbf{x}_{w',h'} \mid \mathbf{x}_{w',h'} \in \mathcal{N}_{w,h}\}$, where $\mathcal{N}_{w,h}$ is the neighborhood of $\mathbf{x}_{w,h}$. Given the number of side words K, we can rewrite it as $\mathcal{G}_{w,h} = \mathbf{x}_{w,h} \cup \left\{\mathbf{x}_{w,h}^{(k)}\right\}_{k=1}^{K}$, where $\mathbf{x}_{w,h}$ is the *central word*, and all the others in $\mathcal{G}_{w,h}$ are *side words*. For simplicity, we consider two fixed types of neighborhood, shown in Figure 1. If the central word is located on the boundary of the neuron map, the side words outside the map are simply ignored.

The **Geometric Neural Phrase Pooling** (GNPP) algorithm computes an updated neural response for each geometric neural phrase $\mathcal{G}_{w,h}$ individually:

$$\mathbf{z}_{w,h} = \frac{1}{2}\left[\mathbf{x}_{w,h} + \max_{k=1}^{K}\left\{s_{w,h}^{(k)} \times \mathbf{x}_{w,h}^{(k)}\right\}\right], \tag{1}$$

where $\max_{k=1}^{K}\{\cdot\}$ is the maximization over K side words. Note that the central word is not included in the maximization. We add the coefficient $\frac{1}{2}$ to approximately preserve the average scale of neuron responses. We define a smoothing parameter $\sigma \in (0, 1]$. A side word is weighted by either $s_{w,h}^{(k)} = \sigma$ or $s_{w,h}^{(k)} = \sigma^2$, according to the relative position to the central word. Of course, one can modify

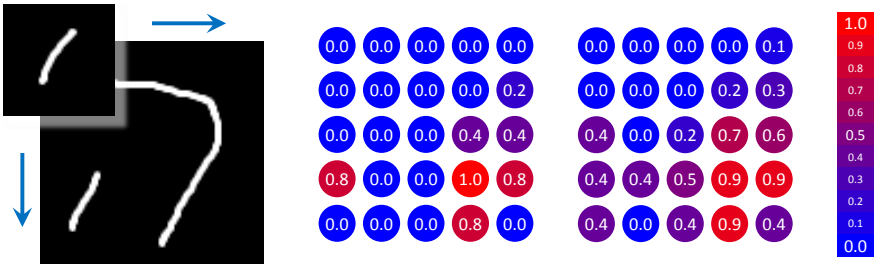

Fig. 2: A conceptual illustration of GNPP (best viewed in color PDF). Left: an image is convoluted with a template. Middle: the original one-dimensional neuron responses. Right: the responses after GNPP (type 1, $\sigma = 1.0$). The isolated high response (around the bottom-left corner) is decreased and smoothed, while the clustered high responses (around the bottom-right corner) are preserved.

the definition of both neighborhood and weights, *e.g.*, using a larger neighborhood or assigning smaller weights on side words, but these minor changes do not impact much on the performance (see Section 4.3).

GNPP averages neuron responses over the central word and the maximal candidate among all side words. Although this looks like the behavior of a local smoother, we emphasize that GNPP is intrinsically different from other smoothers such as vanilla Gaussian blur. Gaussian blur can be formulated as convoluting the input data with a fixed kernel. Applying Gaussian blur after a convolutional layer is similar to using larger kernels, where some weights are not independent to each other. As expected, adding Gaussian blur does not obtain accuracy gain. We add a vanilla Gaussian blur layer before each pooling layer of the **LeNet**, and test it on **CIFAR10**. The baseline error rate is $17.07\% \pm 0.15\%$, and the network with Gaussian blur reports $17.05\% \pm 0.13\%$. On the other side, the network with GNPP reports $14.78\% \pm 0.17\%$ (see Section 4.2). In summary, **GNPP does something that a linear smoother cannot do**.

Since GNPP does not change the dimension (W, H and D) of the neuron cube, we can regard GNPP as an intermediate network layer, *i.e.*, the **GNPP layer**. Although the GNPP layer can, at least in theory, appear anywhere, **we only insert it between a convolutional layer and a pooling layer**, due to the reason to be elaborated in the next subsection.

3.2 Modeling the Spatial Co-occurrence

We show that GNPP is an implicit way of punishing *isolated* neuron responses. Therefore, GNPP works well on the assumption that *clustered* neuron responses are more reliable than *isolated* ones. In this subsection, we will elaborate that such an assumption is better satisfied on the high-level layers of a CNN.

To start, we note that the computation in (1) is carried out in parallel over the D channels. Without loss of generality, we only consider a single channel in a

hidden layer, and our conclusion remains valid for the entire layer (containing D channels). In other words, we can simplify to the situation where we are dealing with $W \times H$ one-dimensional visual words.

In a CNN, neuron responses in one layer are generated by convolution. Convoluting data with the kernel can be regarded as template matching on different spatial locations. After ReLU activation [2], the preserved positive neuron responses correspond to those local patches with high similarity to the template. Figure 2 shows a toy example of (ReLU-activated) convolution results, in which we can find some *clustered* high responses and some *isolated* ones. Since GNPP averages the neuron responses over the central word and the maximal candidate of side words, the *clustered* responses are approximately preserved, while the *isolated* ones are punished. A toy example is shown in Figure 2.

We explain why clustered responses are more reliable, especially on a high-level layer, where the isolated responses often correspond to unexpected random noise [39]. This is because high-level convolutional kernels are highly "specialized", *i.e.*, they often represent concrete visual concepts, *e.g.*, *car wheel* or *aircraft nose* [39]. Meanwhile, as the network level goes up, the receptive field of a neuron becomes larger (*e.g.*, a neuron on the *conv-5* layer of the **AlexNet** can "see" 163×163 pixels on the input image), and neighboring neurons share more and more common visual information (*e.g.*, the overlapping rate of two neighboring neurons on the *conv-5* layer is 90.2%). Thus, if a positive neuron response is caused by the correct match of a visual concept, its neighboring neurons are also likely to be activated, leading to a cluster of positive neuron responses. Oppositely, if it is caused by some random noise, its neighboring neurons may not be activated, and this isolated response shall be punished.

In conclusion, the core idea of GNPP is to model the spatial co-occurrence of neurons produced by a convolutional layer, or find reliable features by punishing the isolated responses which are more likely to be unexpected random noise. We note that pooling, when applied right after GNPP, is an efficient way of aggregating these rectified neuron responses. Therefore, in this work, we only insert GNPP between a convolutional layer and a pooling layer.

3.3 Comparison to Other Work

The GNPP algorithm is inspired by the GPP algorithm which originates from the BoVW model (see Section 2.2). GPP models the spatial context of visual words, and GNPP models the spatial co-occurrence of neural words. In the BoVW model, GPP can only be applied before a max-pooling layer, but GNPP can be inserted anywhere into the CNN model. In the **SVHN** and **CIFAR** experiments, we also show that GNPP cooperates well with the average-pooling layers.

The GNPP layer is related to the Spatial Pyramid Pooling (SPP) layer [40] and the Region-Of-Interest (ROI) pooling layer [41]. However, the motivation and working mechanism of GNPP are quite different from these two layers. The goal of GNPP is to punish isolated neuron responses and improve the descriptive power of every single neuron, while the SPP layer and the ROI pooling layer aim at summarizing local neurons into a regional description. The basic unit in the

GNPP layer is a single neuron, and pooling is performed on a small set of its neighboring neurons, whereas both the SPP layer and the ROI pooling layer work on image regions. Finally, we point out that GNPP can be integrated with other network layers to further improve the recognition performance.

4 Experiments

In this section, we show the experimental results of inserting the GNPP layer into different CNN models. We first observe the performance by evaluating relatively shallow networks on small datasets, then use our conclusions to inform the application of GNPP to deeper networks and the large-scale database.

4.1 The MNIST and SVHN Datasets

MNIST [42] is one of the most popular datasets for handwritten digit recognition. It contains 60,000 training and 10,000 testing samples, uniformly distributed over 10 categories (digits 0–9). All the samples are 28×28 grayscale images. We use a modified version (2 convolutional layers) of the **LeNet** [29] as the baseline. With abbreviated notation, the network configuration is written as:
`{C5(S1P0)@20-MP2(S2)}{C5(S1P0)@50-MP2(S2)}{FC500}{FC10}.`
Here, `C5(S1P0)@20` denotes a convolutional layer with 20 kernels of size 5×5, spatial stride 1 and padding width 0, `MP2(S2)` is a max-pooling layer with pooling region 2×2 and spatial stride 2, and `FC500` is a fully-connected layer with 500 outputs. All the convolution results are activated by ReLU [2], and we use the softmax loss function. In the later experiments, we will directly use the same notations. We apply 20 training epochs with learning rate 10^{-3}, followed by 4 epochs with learning rate 10^{-4}, and another 1 epoch with learning rate 10^{-5}. We test each network five times individually with different initialization and report the averaged error rate and standard deviation.

The **SVHN** dataset [43] is a larger collection of 32×32 RGB images, with 73,257 training samples, 26,032 testing samples, and 531,131 extra training samples. We split the data as in the previous methods [43], *i.e.*, preserving 6,000 images for validation, and using the remainder for training. We use Local Contrast Normalization (LCN) for data preprocessing, following [44][45][46]. We use another version of the **LeNet** with 3 convolutional layers, abbreviated as:
`{C5(S1P2)@32-MP3(S2)}{C5(S1P2)@32-AP3(S2)}{C5(S1P2)@64-AP3(S2)}{FC10}.`
Here, `AP` indicates an average-pooling layer. We apply 12 training epochs with learning rate 10^{-3}, followed by 2 epochs with learning rate 10^{-4}, and another 1 epoch with learning rate 10^{-5}. Each network is individually tested five times.

When the GNPP layer is inserted into the network, it can appear before any subset of the pooling layers. We enumerate all the possibilities, and summarize the results in Table 1. One can observe that the use of GNPP significantly improves the recognition accuracy. The relative error rates are decreased by over 20% on both datasets. Meanwhile, GNPP can be used with Dropout [31] (randomly discarding some neuron responses on the second pooling layer): on **MNIST**, the error rate is reduced from 0.72% to 0.58%.

L1	L2	T1(1.0)	T1(0.9)	T1(0.8)	T2(1.0)	T2(0.9)	T2(0.8)
		0.87 ± .02	0.87 ± .02	0.87 ± .02	0.87 ± .02	0.87 ± .02	0.87 ± .02
✓		0.72 ± .04	0.73 ± .03	0.70 ± .05	0.71 ± .06	0.71 ± .06	0.72 ± .04
	✓	0.75 ± .03	0.79 ± .02	0.77 ± .05	0.73 ± .04	0.75 ± .04	0.73 ± .05
✓	✓	**0.72** ± .03	**0.67** ± .04	**0.69** ± .04	**0.63** ± .03	**0.64** ± .03	**0.67** ± .03

(a) **MNIST**, with the 2-layer **LeNet**, no Dropout

L1	L2	T1(1.0)	T1(0.9)	T1(0.8)	T2(1.0)	T2(0.9)	T2(0.8)
		0.72 ± .03	0.72 ± .03	0.72 ± .03	0.72 ± .03	0.72 ± .03	0.72 ± .03
✓		0.59 ± .02	0.61 ± .05	0.62 ± .03	0.59 ± .03	0.59 ± .02	0.63 ± .03
	✓	0.63 ± .03	0.62 ± .07	0.64 ± .03	0.62 ± .05	0.60 ± .03	0.65 ± .03
✓	✓	**0.58** ± .05	**0.55** ± .02	**0.57** ± .02	**0.54** ± .05	**0.56** ± .04	**0.61** ± .05

(b) **MNIST**, with the 2-layer **LeNet**, Dropout ratio 0.5

L1	L2	L3	T1(1.0)	T1(0.9)	T1(0.8)	T2(1.0)	T2(0.9)	T2(0.8)
			4.63 ± .06	4.63 ± .06	4.63 ± .06	4.63 ± .06	4.63 ± .06	4.63 ± .06
✓			4.46 ± .06	4.47 ± .05	4.42 ± .09	4.42 ± .08	4.42 ± .07	4.43 ± .09
	✓		4.15 ± .08	4.18 ± .01	4.17 ± .07	4.08 ± .10	4.19 ± .07	4.20 ± .05
		✓	3.76 ± .03	3.72 ± .05	3.77 ± .06	3.53 ± .07	3.64 ± .07	3.65 ± .10
✓	✓		4.10 ± .05	4.07 ± .03	4.10 ± .05	4.10 ± .07	4.10 ± .03	4.14 ± .07
✓		✓	3.55 ± .10	3.60 ± .03	3.67 ± .06	3.47 ± .05	3.47 ± .02	3.55 ± .09
	✓	✓	**3.43** ± .06	3.52 ± .07	**3.55** ± .04	**3.41** ± .03	3.42 ± .04	3.51 ± .05
✓	✓	✓	3.46 ± .07	**3.47** ± .06	**3.55** ± .06	3.43 ± .05	**3.39** ± .01	**3.46** ± .03

(c) **SVHN**, with the 3-layer **LeNet**, no Dropout

Table 1: Classification error rates (%) on **MNIST** and **SVHN**. L1, L2 and L3 are three pooling layers, '✓' denotes that GNPP is added. T1 and T2 indicate two types of neighborhood (see Figure 1). 1.0, 0.9 and 0.8 are σ values.

4.2 The CIFAR10 and CIFAR100 Datasets

Both **CIFAR10** and **CIFAR100** datasets [47] are subsets of the 80-million tiny image database [48]. Both of them have 50,000 training and 10,000 testing samples, uniformly distributed over 10 or 100 categories. We also use the 3-layer **LeNet** as in **SVHN**, with the fully-connected layer replaced by `FC100` in **CIFAR100**. We augment the training data by randomly flipping each training image with 50% probability. We apply 120 training epochs with learning rate 10^{-3}, followed by 20 epochs with learning rate 10^{-4}, and another 10 epochs with learning rate 10^{-5}. Each network is individually tested five times.

Results with all possible GNPP settings are summarized in Table 2. Once again, GNPP improves the baseline error rate significantly: the baseline error rates on both **CIFAR10** and **CIFAR100** are reduced by more than 2%, and the relative error rate decrease are 11.25% and 6.56%, respectively.

L1	L2	L3	T1(1.0)	T1(0.9)	T1(0.8)	T2(1.0)	T2(0.9)	T2(0.8)
			$17.07 \pm .15$	$17.07 \pm .15$	$17.07 \pm .15$	$17.07 \pm .15$	$17.07 \pm .15$	$17.07 \pm .15$
✓			$16.67 \pm .22$	$16.80 \pm .25$	$16.84 \pm .12$	$16.65 \pm .19$	$17.03 \pm .15$	$17.04 \pm .17$
	✓		$15.79 \pm .22$	$16.09 \pm .17$	$15.95 \pm .31$	$15.69 \pm .11$	$16.07 \pm .27$	$15.90 \pm .09$
		✓	$15.49 \pm .15$	$15.31 \pm .20$	$15.51 \pm .25$	$15.27 \pm .10$	$15.29 \pm .14$	$15.28 \pm .16$
✓	✓		$15.82 \pm .23$	$15.76 \pm .18$	$15.98 \pm .14$	$16.05 \pm .29$	$15.90 \pm .25$	$15.94 \pm .09$
✓		✓	$15.15 \pm .20$	$15.29 \pm .12$	$15.44 \pm .19$	$15.29 \pm .32$	$15.19 \pm .35$	$15.20 \pm .35$
	✓	✓	$\mathbf{14.92} \pm .18$	$15.00 \pm .18$	$15.15 \pm .15$	$\mathbf{14.83} \pm .25$	$14.93 \pm .20$	$14.92 \pm .16$
✓	✓	✓	$14.97 \pm .17$	$\mathbf{14.83} \pm .23$	$\mathbf{14.78} \pm .17$	$15.22 \pm .16$	$\mathbf{14.79} \pm .26$	$\mathbf{14.85} \pm .26$

(a) **CIFAR10**, with the 3-layer **LeNet**, Dropout ratio 0.2

L1	L2	L3	T1(1.0)	T1(0.9)	T1(0.8)	T2(1.0)	T2(0.9)	T2(0.8)
			$44.99 \pm .19$	$44.99 \pm .19$	$44.99 \pm .19$	$44.99 \pm .19$	$44.99 \pm .19$	$44.99 \pm .19$
✓			$44.62 \pm .17$	$44.53 \pm .45$	$44.78 \pm .06$	$44.43 \pm .29$	$44.58 \pm .36$	$44.58 \pm .52$
	✓		$43.34 \pm .23$	$43.71 \pm .19$	$43.37 \pm .26$	$43.21 \pm .23$	$43.03 \pm .27$	$43.37 \pm .30$
		✓	$43.11 \pm .24$	$42.77 \pm .37$	$42.99 \pm .24$	$42.96 \pm .32$	$42.81 \pm .38$	$43.08 \pm .39$
✓	✓		$43.99 \pm .07$	$43.63 \pm .11$	$43.50 \pm .26$	$43.38 \pm .37$	$43.34 \pm .27$	$43.46 \pm .25$
✓		✓	$42.85 \pm .38$	$42.81 \pm .27$	$42.82 \pm .29$	$43.08 \pm .27$	$42.79 \pm .34$	$42.93 \pm .22$
	✓	✓	$\mathbf{42.35} \pm .30$	$\mathbf{42.34} \pm .31$	$\mathbf{42.04} \pm .20$	$\mathbf{42.92} \pm .33$	$\mathbf{42.72} \pm .25$	$\mathbf{42.54} \pm .29$
✓	✓	✓	$42.97 \pm .29$	$42.77 \pm .36$	$42.36 \pm .18$	$43.31 \pm .34$	$42.85 \pm .18$	$42.60 \pm .36$

(b) **CIFAR100**, with the 3-layer **LeNet**, no Dropout

Table 2: Classification error rates (%) on the **CIFAR** datasets. We use the same notations as in Table 1. We apply Dropout on the simpler **CIFAR10** task.

4.3 Analysis on Small Experiments

Before we go into deeper networks and larger datasets, we conduct some preliminary analysis based on the results we already have.

First, although inserting GNPP before any pooling layers improves the performance, the most significant accuracy gain brought by a single GNPP layer is obtained by adding GNPP before the last pooling layer. This reinforces the conclusion drawn in Section 3.2, *i.e.*, GNPP works better on the high-level neuron responses. Meanwhile, on the **SVHN** and **CIFAR** datasets, adding GNPP before all three pooling layers produces inferior results to that adding GNPP before the second and third pooling layers. In the later experiments, we first add the GNPP layer before each pooling layer individually, then use the results to inform the design of the final model.

Regarding the scale of neural phrases, *i.e.*, K, we find that increasing the scale is not guaranteed to produce better recognition results. We explain this by noticing that adding a faraway side word to a neural phrase, most often, does not provide much related information but risks introducing noise to that unit. This idea can also be used to explain why a proper smoothing parameter, say, $\sigma = 0.8$, helps to reduce the contribution of faraway side words, leading to

	MNIST	SVHN	CIFAR10	CIFAR100
Zeiler *et.al* [45]	0.47	2.80	15.13	42.51
Goodfellow *et.al* [46]	0.45	2.47	9.38	38.57
Lin *et.al* [51]	0.47	2.35	8.81	35.68
Lee *et.al* [52]	0.39	1.92	7.97	34.57
Liang *et.al* [53]	**0.31**	1.77	7.09	31.75
Lee *et.al* [54]	**0.31**	1.69	6.05	32.37
BigNet (without GNPP)	0.36	2.14	7.80	31.03
BigNet (with GNPP)	**0.32**	**1.87**	**7.14**	**29.74**
WRN (without GNPP)	0.34	1.77	5.54	25.52
WRN (with GNPP)	**0.31**	**1.67**	**5.31**	**25.01**

Table 3: Comparison of the recognition error rate (%) with the state-of-the-arts. We apply data augmentation on all these datasets, but the competitors do not use it in **CIFAR100**. Without data augmentation, we report 29.92% and 29.17% error rates (using **WRN**) without and with GNPP, respectively.

better recognition performance. One may certainly try other choices such as a large neighborhood with a very small σ, but we note that the time complexity of a GNPP layer is linear to K. In the later experiments, we will directly use the first type of neighborhood ($K = 4$) with $\sigma = 0.8$.

4.4 Deeper Networks and the State-of-the-Arts

We adopt two deeper networks on the above four small datasets to compare with the state-of-the-art results. One of them (we name it as the **BigNet**) is borrowed from [49] in the Kaggle recognition competition, and other one one is the 16-layer Wide Residual Network (**WRN**) [50] with dropout. Both networks can be used in each of the four small datasets. In **CIFAR** datasets, we randomly flip the image with 50% probability. We train the **BigNet** using 6×10^6 samples with learning rate 10^{-2}, followed by 3×10^6 samples with learning rate 10^{-3} and 1×10^6 samples with learning rate 10^{-4}, respectively. We report a 7.80% error rate on **CIFAR10**, comparable to the original version [49], which uses a very complicated way of data preparation and augmentation to get a 6.68% error rate. Training our model needs about 1 hour, while the original version [49] requires 6 hours. We train the **WRN** following the original configuration in [50].

We compare our results with the state-of-the-arts in Table 3. We add GNPP before the second and the third pooling layers for **BigNet**, and the last pooling layer for **WRN**. Although the baseline is already pretty high, GNPP still improves it by a margin: on **BigNet**, the relative error rate drops are 11.11%, 12.62%, 8.46% and 4.16% on the four datasets, respectively. Without complicated tricks, our results are very competitive among these recent works. We believe that GNPP can also be applied to other powerful networks in the future.

4.5 ImageNet Experiments

Finally, we evaluate our model on the **ImageNet** large-scale visual recognition task (the **ILSVRC2012** dataset [3] with 1000 categories). We use the **AlexNet** (provided by the **CAFFE** library [4]), abbreviated as:
`{C11(S4)@96-MP3(S2)}{C5(S1P2)@256-MP3(S2)}{C3(S1P1)@384}{C3(S1P1)@384}{C3(S1P1)@256-MP3(S2)}{FC4096-D0.5}{FC4096-D0.5}{FC1000}.`
The input image is of size 227×227, randomly cropped from the original 256×256 image. Following the setting of **CAFFE**, a total of 450,000 mini-batches (approximately 90 epochs) are used for training, each of which has 256 image samples, with the initial learning rate 10^{-2}, momentum 0.9 and weight decay 5×10^{-4}. The learning rate is decreased to 1/10 after every 100,000 mini-batches.

AlexNet contains three max-pooling layers, *i.e.*, *pool-1*, *pool-2* and *pool-5*. After individual tests, we only add GNPP before the last one (*pool-5*), since adding GNPP before either *pool-1* or *pool-2* causes accuracy drop. With the GNPP layer, the top-1 and top-5 recognition error rates are 42.16% and 19.24%, respectively. Comparing to the original rates (43.19% and 19.87%), GNPP boosts them by about 1.0% and 0.6%, respectively. We emphasize that the accuracy gain is not so small as it seems, especially considering that we do not introduce extra parameters and that the overall training time is only increased by 1.29%.

Although GNPP is tested on **AlexNet**, we believe it can be applied to other models, such as **VGGNet** [33], **GoogleNet** [34] and Deep Residual Nets [35].

5 Benefits of GNPP

This section presents several discussions and diagnostic experiments that help us understand the side benefits brought by the GNPP layer.

5.1 Improving Internal Representation

Here we compare the *conv-5* layer of the standard **AlexNet** with the corresponding layers in the **GNPPNet** (defined in Section 4.5). That is, we compare **AlexNet**'s *conv-5* layer with **GNPPNet**'s *conv-5* layer and *GNPP-5* layer. Each layer is a $13 \times 13 \times 256$ neuron blob corresponding to 256 convolutional kernels. We average over the 256 channels and obtain a 13×13 heatmap. To allow direct comparison with the input image (227×227), we diffuse each neuron response as a Gaussian distribution over its receptive field on the input image (the same standard deviation is used on all layers). Results are shown in Figure 3. It is observed in [39] that the activation patterns in higher convolutional layers correspond to mid-level parts. The average over filters is a crude measure that some mid-level parts are detected. Then Figure 3 shows the spatial pattern corresponding to mid-level part detection.

First note that **AlexNet**'s *conv-5* layer and **GNPPNet**'s *GNPP-5* layer are broadly similar. This is to be expected as both of them occur at corresponding places in the network architecture, *i.e.*, just before the *pool-5* layer and the fully-connected layers. We might think of the filter averages shown in Figure 3 as

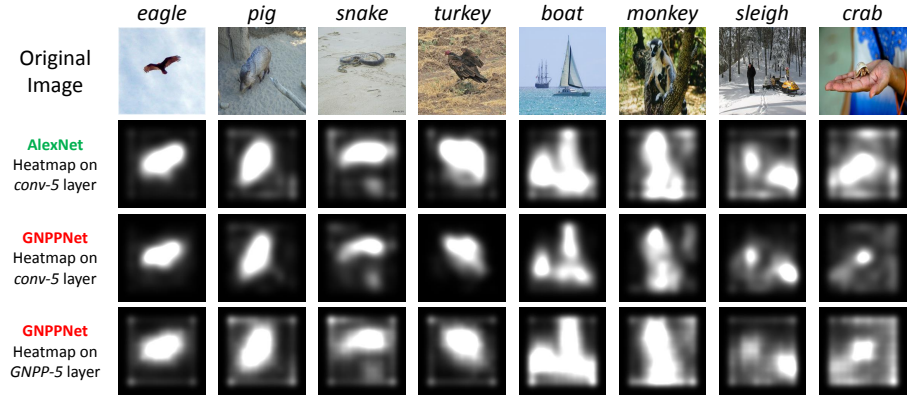

Fig. 3: Neuron response heatmaps produced by **AlexNet** and **GNPPNet**. When the background is relatively simple (*e.g.*, first two images), both methods work well. On those challenging cases, GNPP produces better saliency detection results, implying that the internal representation of CNN is improved.

spatial summaries of average scores over object parts. The higher layers in both networks combine spatial co-occurrences of parts into whole object detectors. For example, *car wheels* and *car doors* are combined into a whole *car*.

Next notice that **GNPPNet**'s *conv-5* layer is sparser and more concentrated than **AlexNet**'s *conv-5* layer. Broadly speaking the GNPP operation acts as a smoother and it is the smoothed *conv-5* layer (*i.e.*, the *GNPP-5* layer) that resembles **AlexNet**'s *conv-5* layer. The difference between **AlexNet**'s *conv-5* layer and **GNPPNet**'s *GNPP-5* layer is subtle, but we see that the *GNPP-5* layer is more diffuse corresponding to GNPP's action as local smoother.

As a result, **GNPPNet**'s *conv-5* layer produces better saliency detection results compared to **AlexNet**'s *conv-5* layer. This property can be used to extract better *deep features*. We verify our hypothesis on the **Caltech256** dataset [55]. 256-dimensional feature vectors are extracted from the *conv-5* layer by averaging over 13×13 spatial locations. The classification accuracy using the **AlexNet** is 59.36%, and **GNPPNet** improves it to 60.56%. This improvement is significant given that no extra time or memory is required for feature extraction.

In summary, applying GNPP to CNN produces better internal representation. The deep features extracted from the **GNPPNet** can also benefit other vision applications, such as image retrieval [6] and object detection [8][9].

5.2 Building Latent Connections

We show that GNPP builds latent connections between hidden layers in the CNN model. Consider a geometric neural phrase $\mathcal{G}_{w,h} = \mathbf{x}_{w,h} \cup \left\{ \mathbf{x}_{w,h}^{(k)} \right\}_{k=1}^{K}$. Let $\mathcal{S}_{w,h}$ be the set of neurons in the previous layer that are connected to $\mathbf{x}_{w,h}$, and $\mathcal{S}_{w,h}^{(k)}$

be the set connected to $\mathbf{x}_{w,h}^{(k)}$, $k = 1, 2, \ldots, K$. If we consider $\mathcal{G}_{w,h}$ as a *GNPP neuron*, then the set of neurons in the previous layer that are connected to it is $\mathcal{S}_{w,h} \cup \bigcup_{k=1}^{K} \mathcal{S}_{w,h}^{(k)}$. Thus, we are actually building latent neuron connections which do not exist in the original network. For example, applying GNPP (type 1) before the *pool-5* layer of the **AlexNet** increases the number of neuron connections between *conv-4* and *conv-5* from 149.5M (million) to 348.9M (on each neuron in *conv-5*, the number of connections to the previous layer increases from 9 to 21), meanwhile the number of learnable parameters remains unchanged.

To verify the benefits of latent connections, we train another version of **AlexNet**, referred to as **AlexNet2**, with the difference that the number of channels on the *conv-5* layer increases from 256 to 512. The number of neuron connections between *conv-4* and *conv-5* increases from 149.5M to 299.0M, comparable to 348.9M in **GNPPNet**. **AlexNet2** requires 9.97% extra training time and 5.58% extra GNPP memory, while the numbers for **GNPPNet** are 1.29% and 2.52%, respectively. **AlexNet2** produces 42.45% (top-5) and 19.47% (top-1) recognition error rates, which are higher than 43.19% and 19.97% reported by **AlexNet**, but lower than 42.16% and 19.24% reported by **GNPPNet**. To summarize, GNPP allows latent connections to be built in an efficient manner.

5.3 Accelerating Network Training

We show that adding GNPP layers accelerates the network training process, since GNPP allows visual information to propagate faster, like [56].

Let us investigate the case that training the 3-layer **LeNet** on the **SVHN** and **CIFAR** datasets. We are interested in the following question: if the input is a 32×32 image, which is the earliest layer containing a neuron able to "see" the entire image? Without GNPP, we need to wait until the *conv-3* layer. When GNPP is inserted before the second pooling layer, the receptive field of the neurons on the subsequent layers are increased. Consequently, some neurons in *pool-2* can already "see" the entire image. This allows some low-level and mid-level information (*e.g.*, object parts) be combined earlier.

As a result, GNPP helps the network training process converge faster. To verify, we plot the testing error rates and the loss function values throughout the training process. The results on the **SVHN** and **CIFAR100** datasets, using the **LeNet**, are shown in Figure 4. One can see that GNPP causes the error rate and loss function curves drop more quickly, especially in the early epochs. For example, in the **SVHN** dataset, the network without GNPP requires about 36,000 iterations to reach 6% error rate, while that with GNPP only needs about 15,000 iterations to get the same rate. Meanwhile, the training process reaches plateau sooner in the GNPP-equipped networks (see the error rate curve between 6–12 epochs in **SVHN**, and that between 60–120 epochs in **CIFAR100**).

With the help of GNPP, we can even train a network faster and obtain better performance. The baseline error rates on **SVHN** and **CIFAR100**, using the **LeNet**, are 4.63% and 44.99%, respectively. We train a GNPP-equipped **LeNet** with half training epochs under each learning rate, and obtain 3.78% and 43.35% error rates (the full training reports 3.55% and 42.04%).

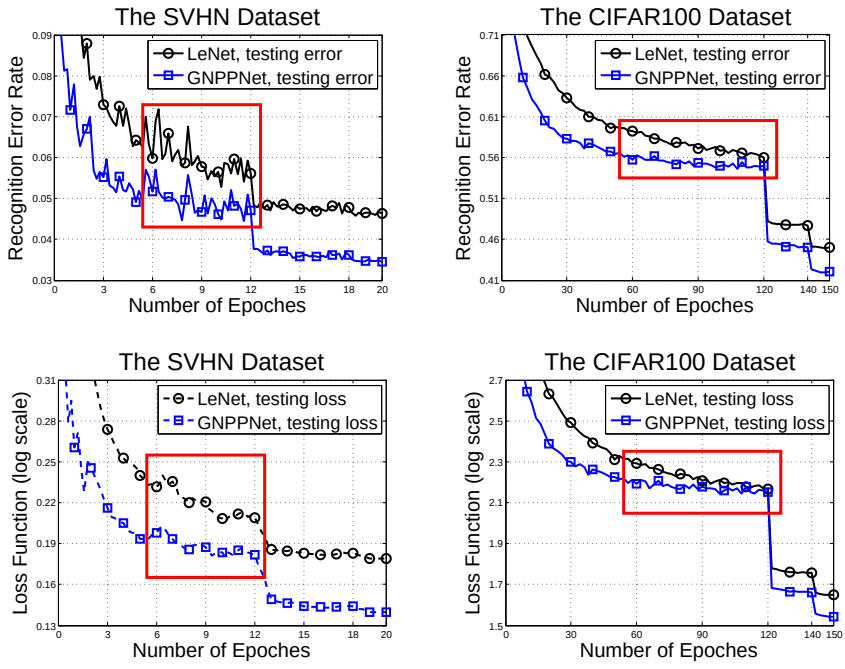

Fig. 4: Error rate and loss function curves on the **SVHN** and **CIFAR100** datasets. **GNPPNet** refers to the **LeNet** with two GNPP layers inserted before the second and third pooling layers. The curves in red frames indicate that **GNPPNet** enjoys better convergence, *i.e.*, it reaches the plateau sooner.

6 Conclusions

In this paper, we demonstrate that constructing and encoding *neural phrases* boost the performance of state-of-the-art CNNs. We insert Geometric Neural Phrase Pooling (GNPP) as an intermediate layer into the network, and show that it improves the performance of deep networks without requiring much more computational resources. GNPP can be explained as an implicit way of modeling the spatial co-occurrence of neurons. We also show that GNPP enjoys the advantage of improving the internal representation of CNN, building latent connections, and speeding up the network training process.

Our work illustrates that designing deep networks benefits from prior knowledge. In the case of GNPP, we learn that the isolated neuron responses are less reliable than the clustered ones. We hope that other prior knowledge can be useful incorporated into the CNN architecture. Meanwhile, other visual tasks, including detection, segmentation, *etc.*, may also benefit from the GNPP algorithm. The exploration of these topics is left for future works.

Acknowledgements

This paper is supported in part to Prof. Alan Yuille by iARPA MICrONS contract D16PC00007 and ONR N00014-12-1-0883. This work is supported in part to Prof. Qi Tian by ARO grants W911NF-15-1-0290 and Faculty Research Gift Awards by NEC Laboratories of America and Blippar. This work is also supported in part by National Science Foundation of China (NSFC) 61429201. We thank Junhua Mao, Cihang Xie and Zhuotun Zhu for insightful discussions.

References

1. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. Computer Vision and Pattern Recognition (2009)
2. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (2012)
3. 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 (2015) 1–42
4. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: CAFFE: Convolutional Architecture for Fast Feature Embedding. ACM International Conference on Multimedia (2014)
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. International Conference on Machine Learning (2014)
6. Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN Features off-the-shelf: an Astounding Baseline for Recognition. Computer Vision and Pattern Recognition (2014)
7. Xie, L., Hong, R., Zhang, B., Tian, Q.: Image Classification and Retrieval are ONE. International Conference on Multimedia Retrieval (2015)
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Computer Vision and Pattern Recognition (2014)
9. Girshick, R.: Fast R-CNN. International Conference on Computer Vision (2015)
10. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual Categorization with Bags of Keypoints. Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision **1**(22) (2004) 1–2
11. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal on Computer Vision **60**(2) (2004) 91–110
12. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. Computer Vision and Pattern Recognition (2005) 886–893
13. Xie, L., Wang, J., Lin, W., Zhang, B., Tian, Q.: RIDE: Reversal Invariant Descriptor Enhancement. International Conference on Computer Vision (2015)
14. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. Computer Vision and Pattern Recognition (2009) 1794–1801
15. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-Constrained Linear Coding for Image Classification. Computer Vision and Pattern Recognition (2010)

16. Perronnin, F., Sanchez, J., Mensink, T.: Improving the Fisher Kernel for Large-scale Image Classification. European Conference on Computer Vision (2010)
17. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image Classification Using Super-Vector Coding of Local Image Descriptors. European Conference on Computer Vision (2010)
18. Kobayashi, T.: Dirichlet-Based Histogram Feature Transform for Image Classification. Computer Vision and Pattern Recognition (2014)
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. Computer Vision and Pattern Recognition (2006)
20. Feng, J., Ni, B., Tian, Q., Yan, S.: Geometric Lp-norm Feature Pooling for Image Classification. Computer Vision and Pattern Recognition (2011)
21. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research **9** (2008) 1871–1874
22. Boiman, O., Shechtman, E., Irani, M.: In Defense of Nearest-Neighbor Based Image Classification. Computer Vision and Pattern Recognition (2008)
23. Yuan, J., Wu, Y., Yang, M.: Discovery of Collocation Patterns: from Visual Words to Visual Phrases. Computer Vision and Pattern Recognition (2007)
24. Zhang, Y., Jia, Z., Chen, T.: Image Retrieval with Geometry-Preserving Visual Phrases. Computer Vision and Pattern Recognition (2011)
25. Zhang, S., Tian, Q., Hua, G., Huang, Q., Li, S.: Descriptive Visual Words and Visual Phrases for Image Applications. ACM Multimedia (2009)
26. Xie, L., Tian, Q., Wang, M., Zhang, B.: Spatial Pooling of Heterogeneous Features for Image Classification. IEEE Transactions on Image Processing **23**(5) (2014) 1994–2008
27. Jiang, Y., Meng, J., Yuan, J.: Randomized Visual Phrases for Object Search. Computer Vision and Pattern Recognition (2012)
28. Xie, L., Tian, Q., Hong, R., Yan, S., Zhang, B.: Hierarchical Part Matching for Fine-Grained Visual Categorization. IEEE International Conference on Computer Vision (2013)
29. LeCun, Y., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten Digit Recognition with a Back-Propagation Network. Advances in Neural Information Processing Systems (1990)
30. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. International Conference on Machine Learning (2015)
31. Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. arXiv preprint, arXiv: 1207.0580 (2012)
32. Xie, L., Wang, J., Wei, Z., Wang, M., Tian, Q.: DisturbLabel: Regularizing CNN on the Loss Layer. Computer Vision and Patter Recognition (2016)
33. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations (2015)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going Deeper with Convolutions. Computer Vision and Pattern Recognition (2015)
35. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. arXiv preprint, arXiv: 1512.03385 (2015)
36. Zhang, N., Paluri, M., Ranzato, M., Darrell, T., Bourdev, L.: PANDA: Pose Aligned Networks for Deep Attribute Modeling. Computer Vision and Pattern Recognition (2014)

37. Xie, L., Zheng, L., Wang, J., Yuille, A., Tian, Q.: InterActive: Inter-Layer Activeness Propagation. Computer Vision and Patter Recognition (2016)
38. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the Devil in the Details: Delving Deep into Convolutional Nets. British Machine Vision Conference (2014)
39. Wang, J., Zhang, Z., Premachandran, V., Yuille, A.: Discovering Internal Representations from Object-CNNs Using Population Encoding. arXiv preprint, arXiv: 1511.06855 (2015)
40. He, K., Zhang, X., Ren, S., Sun, J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. European Conference on Computer Vision (2014)
41. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems (2015)
42. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
43. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.: Reading Digits in Natural Images with Unsupervised Feature Learning. NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
44. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional Neural Networks Applied to House Numbers Digit Classification. International Conference on Pattern Recognition (2012)
45. Zeiler, M., Fergus, R.: Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. International Conference on Learning Representations (2013)
46. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout Networks. International Conference on Machine Learning (2013)
47. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto (2009)
48. Torralba, A., Fergus, R., Freeman, W.: 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(11) (2008) 1958–1970
49. Nagadomi: The Kaggle CIFAR10 Network. https://github.com/nagadomi/kaggle-cifar10-torch7/ (2014)
50. Zagoruyko, S., Komodakis, N.: Wide Residual Networks. arXiv preprint, arXiv: 1605.07146 (2016)
51. Lin, M., Chen, Q., Yan, S.: Network in Network. International Conference on Learning Representations (2014)
52. Lee, C., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-Supervised Nets. International Conference on Artificial Intelligence and Statistics (2015)
53. Liang, M., Hu, X.: Recurrent Convolutional Neural Network for Object Recognition. Computer Vision and Pattern Recognition (2015)
54. Lee, C., Gallagher, P., Tu, Z.: Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. International Conference on Artificial Intelligence and Statistics (2016)
55. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Technical Report: CNS-TR-2007-001, Caltech (2007)
56. Srivastava, R., Greff, K., Schmidhuber, J.: Training Very Deep Networks. Advances in Neural Information Processing Systems (2015)