InterActive: Inter-Layer Activeness Propagation

Lingxi Xie[1†] * Liang Zheng[2†] Jingdong Wang[3] Alan Yuille[4] Qi Tian[5]
[1,4]Department of Statistics, University of California, Los Angeles, Los Angeles, CA, USA
[3]Microsoft Research, Beijing, China
[4]Departments of Cognitive Science and Computer Science, Johns Hopkins University, Baltimore, MD, USA
[2,5]Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA
[1]198808xc@gmail.com [2]liangzheng06@gmail.com
[3]jingdw@microsoft.com [4]alan.l.yuille@gmail.com [5]qitian@cs.utsa.edu

Abstract

An increasing number of computer vision tasks can be tackled with deep features, which are the intermediate outputs of a pre-trained Convolutional Neural Network. Despite the astonishing performance, deep features extracted from low-level neurons are still below satisfaction, arguably because they cannot access the spatial context contained in the higher layers. In this paper, we present **InterActive***, a novel algorithm which computes the* **activeness** *of neurons and network connections. Activeness is propagated through a neural network in a top-down manner, carrying high-level context and improving the descriptive power of low-level and mid-level neurons. Visualization indicates that neuron activeness can be interpreted as spatial-weighted neuron responses. We achieve state-of-the-art classification performance on a wide range of image datasets.*

1. Introduction

We have witnessed a big revolution in computer vision brought by the deep Convolutional Neural Networks (C-NN). With powerful computational resources and a large amount of labeled training data [8], a differentiable function for classification is trained [23] to capture different levels of visual concepts organized by a hierarchical structure. A pre-trained deep network is also capable of generating deep features for various tasks, such as image classification [20][9], image retrieval [38][48] and object detection [15].

Although deep features outperform conventional image representation models such as Bag-of-Visual-Words (BoVW), we note that the deep feature extraction process only involves forward propagation: an image is rescaled into a fixed size, input into a pre-trained network, and the intermediate neuron responses are summarized as visual features. As we shall see in Section 3.1, such a method ignores important high-level visual context, causing both a "big" problem and a "small" problem (see Figure 1). These problems harm the quality of the deep features, and, consequently, visual recognition accuracy.

In this paper, we present **InterActive**, a novel deep feature extraction algorithm which integrates high-level visual context with low-level neuron responses. For this, we measure the *activeness* of neuron connections for each specified image, based on the idea that a connection is more important if the network output is more sensitive to it. We define an unsupervised probabilistic distribution function over the high-level neuron responses, and compute the *score function* (a concept in statistics) with respect to each connection. Each neuron obtains its *activeness* by collecting the activeness of the related connections. InterActive increases the receptive field size of low-level neurons by allowing the supervision of the high-level neurons. We interpret neuron activeness in terms of spatial-weighted neuron responses, and the visualization of neuron weights demonstrates that visually salient regions are detected in an unsupervised manner. More quantitatively, using the improved InterActive features, we achieve state-of-the-art image classification performance on several popular benchmarks.

The remainder of this paper is organized as follows. Section 2 briefly introduces related works. The InterActive algorithm is presented in Section 3. Experiments are shown in Section 4, and we conclude this work in Section 5.

2. Related Works

Image classification is a fundamental problem in computer vision. In recent years, researchers have extended

*This work was partly done when Lingxi Xie and Liang Zheng were interns at MSR. They contributed equally. This work was supported by NIH Grant 5R01EY022247-03, ONR N00014-12-1-0883 and ARO grant W911NF-15-1-0290, Faculty Research Gift Awards by NEC Labs of America and Blippar, and NSFC 61429201. We thank John Flynn, Xuan Dong, Jianyu Wang, Junhua Mao and Zhuotun Zhu for discussions.

1063-6919/16 $31.00 © 2016 IEEE
DOI 10.1109/CVPR.2016.36

the conventional tasks [24][11] to fine-grained [33][43][34], and large-scale [16][46][8] cases.

The Bag-of-Visual-Words (BoVW) model [6] represents each images with a high-dimensional vector. It typically consists of three stages, *i.e.*, descriptor extraction, feature encoding and feature summarization. Due to the limited descriptive power of raw pixels, local descriptors such as SIFT [30] and HOG [7] are extracted. A visual vocabulary is then built to capture the data distribution in feature space. Descriptors are thereafter quantized onto the vocabulary as compact feature vectors [53][44][35][49], and summarized as an image-level representation [24][12][58]. These feature vectors are post-processed [50], and then fed into a machine learning tool [10][1][48] for evaluation.

The Convolutional Neural Network (CNN) serves as a hierarchical model for large-scale visual recognition. It is based on that a network with enough neurons is able to fit any complicated data distribution. In past years, neural networks were shown to be effective for simple recognition tasks [25]. More recently, the availability of large-scale training data (*e.g.*, ImageNet [8]) and powerful GPUs makes it possible to train deep CNNs [23] which significantly outperform BoVW models. A CNN is composed of several stacked layers, in each of which responses from the previous layer are convoluted and activated by a differentiable function. Hence, a CNN can be considered as a composite function, and is trained by back-propagating error signals defined by the difference between supervised and predicted labels at the top level. Recently, efficient methods were proposed to help CNNs converge faster [23] and prevent over-fitting [17][18][52]. It is believed that deeper networks produce better recognition results [40][41].

The intermediate responses of CNN, or the so-called deep features, serve as efficient image description [9], or a set of latent visual attributes. They can be used for various vision applications, including image classification [20], image retrieval [38][48], object detection [15][14] and object parsing [45]. A discussion of how different CNN configurations impact deep feature performance is available in [4].

Visualization is an effective method of understanding CNNs. In [54], a *de-convolutional* operation was designed to capture visual patterns on different layers of a pre-trained network. [39] and [2] show that different sets of neurons are activated when a network is used for detecting different visual concepts. The above works are based on a *supervised* signal on the output layer. In this paper, we define an *unsupervised* probabilistic distribution function on the high-level neuron responses, and back-propagate it to obtain the activeness of low-level neurons. Neuron activeness can also be visualized as spatial weighting maps. Computing neuron activeness involves finding the relevant contents on each network layer [31][5], and is related to recovering low-level details from high-level visual context [29].

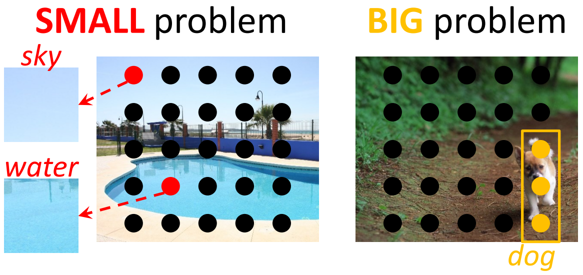

Figure 1. Examples showing the "big" problem and the "small" problem of deep feature extraction (best viewed in color PDF). Left image (label: *swimming pool*): the receptive regions of two red neurons are visually similar, but correspond to different semantics (*sky* vs. *water*), implying that the receptive field of low-level neurons is often too *small* to capture contexts. Right image (label: a *dog* species): only the yellow neurons are located on the object, but standard deep features used in classification are pooled over all neurons, most of which are irrelevant, suggesting that the pooling region is often too *big* compared to the semantic region.

3. Inter-Layer Activeness Propagation

3.1. Motivation

We start with deep features extracted from a pre-trained CNN. Throughout this paper, we will use the very deep **VGGNet** [40] with 19 convolutional layers. This produces competitive performance to **GoogLeNet** [41], and outperforms **AlexNet** [23] significantly. We also adopt the same notation for layers used in **VGGNet**, *e.g.*, *conv-3-3*, *pool-5* and *fc-7*. All the referred neuron responses are ReLU-processed, *i.e.*, negative values are replaced by 0.

One of the popular deep feature extraction approaches works as follows: an image is warped (resized) to the same size as the input of a pre-trained network (*e.g.* 224×224 in **VGGNet**), then fed into the network, and the responses at an intermediate layer (*e.g.*, *fc-6*) are used for image representation. A key observation of [40] is that recognition accuracy is significantly boosted if the input images are not warped. In what follows, we resize an image, so that the number of pixels is approximately 512^2, both width and height are divisible by 32 (the down-sampling ratio of **VGGNet**), and the aspect ratio is maximally preserved. Using this setting, we obtain a 3D data cube at each layer (even for *fc-6* and *fc-7*), and perform average-pooling or max-pooling to aggregate it as image representation. We emphasize that such a simple resizing modification gives significant improvement in recognition accuracy. For example, with features extracted from the *fc-6* layer, the classification accuracy is 83.51%, 61.30% and 93.54% on the **Caltech256**, **SUN-397** and **Flower-102** datasets, whereas features extracted from warped images only report 80.41%, 53.06% and 84.89%, respectively. On the *pool-5* layer, the numbers are 81.40%, 55.22% and 94.70% for un-warped input images, and 77.46%, 48.19% and 86.87% for warped ones, also showing significant improvement.

Compared to the large input image size (approximately 512^2 pixels), the receptive field of a neuron on an intermediate layer is much smaller. For example, a neuron on the *pool-4*, *pool-5* and *fc-6* layers can *see* 124×124, 268×268 and 460×460 pixels on the input image, respectively, while its effective receptive field is often much smaller [54][56]. We argue that small receptive fields cause the following problems: (1) a low-level neuron may not see enough visual context to make prediction, and (2) there may be many irrelevant neurons which contaminate the image representation. We name them the "small" problem and the "big" problem, respectively, as illustrated in Figure 1.

Both the above problems can be solved if low-level neurons receive more visual information from higher levels. In the network training process, this is achieved by error back-propagation, in which low-level neurons are supervised by high-level neurons to update network weights. In this section, we present **InterActive**, which is an unsupervised method allowing back-propagating high-level context on the testing stage. InterActive involves defining a probabilistic distribution function (PDF) on the high-level neuron responses, and computing the *score function* which corresponds to the *activeness* of network connections. As we will see in Section 3.4, this is equivalent to adding spatial weights on low-level neuron responses.

3.2. The Activeness of Network Connections

Let a deep CNN be a mathematical function $\mathbf{h}\big(\mathbf{X}^{(0)};\boldsymbol{\Theta}\big)$, in which $\mathbf{X}^{(0)}$ denotes the input image and $\boldsymbol{\Theta}$ the weights over neuron connections. There are in total L layers, and the response on the t-th layer is $\mathbf{X}^{(t)}$ ($t = 0$ indicates the input layer). In our approach, $\mathbf{X}^{(t)}$ is a vector of length $W_t \times H_t \times D_t$, where W_t, H_t and D_t denote the width, height and depth (number of channels), respectively. $x_{w,h,d}^{(t)}$ is a neuron on the t-th layer. The connections on the t-th layer, $\boldsymbol{\theta}^{(t)}$, are a matrix of $(W_t \times H_t \times D_t) \times (W_{t+1} \times H_{t+1} \times D_{t+1})$ elements, where $\theta_{w,h,d,w',h',d'}^{(t)}$ connects neurons $x_{w,h,d}^{(t)}$ and $x_{w',h',d'}^{(t+1)}$. Let $\mathcal{U}_{w,h,d}^{(t)}$ be the set of neurons on the $(t+1)$-st layer that are connected to $x_{w,h,d}^{(t)}$, and $\mathcal{V}_{w',h',d'}^{(t+1)}$ be the set of neurons on the t-th layer that are connected to $x_{w',h',d'}^{(t+1)}$. Hence, the convolution operation can be written as:

$$x_{w',h',d'}^{(t+1)} = \sigma\left[\sum_{(w,h,d)\in\mathcal{V}_{w',h',d'}^{(t+1)}} x_{w,h,d}^{(t)} \cdot \theta_{w,h,d,w',h',d'}^{(t)} + b\right], \tag{1}$$

where $b = b_{w',h',d'}^{(t+1)}$ is the bias term, and $\sigma[\cdot]$ is the ReLU activation: $\sigma[\cdot] = \max(\cdot, 0)$.

We study the PDF on the T-th layer $f\big(\mathbf{x}^{(T)}\big)$ by sampling, where $\mathbf{x}^{(T)} = \Big(x_1^{(T)}, \ldots, x_{D_T}^{(T)}\Big)^\top$ is the averaged neuron response vector over all spatial positions:

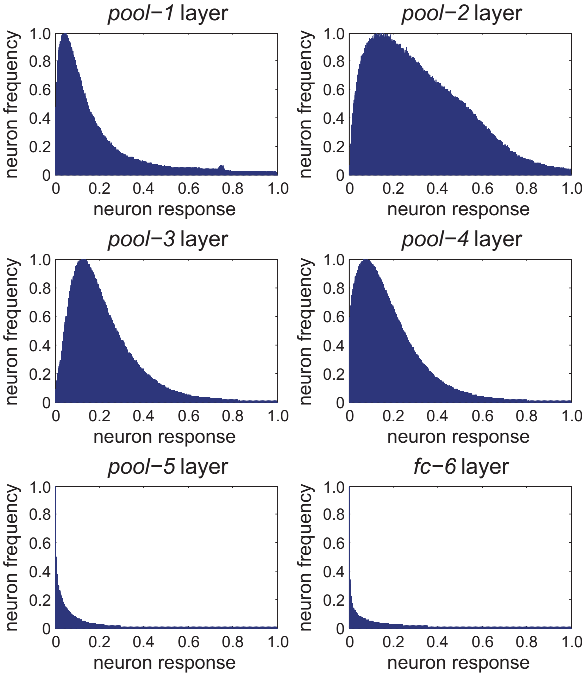

Figure 2. The statistics of neuron responses on different layers. For better visualization, we have filtered all the 0-responses and normalized the neuron responses and the neuron frequency.

$$x_d^{(T)} = \frac{1}{W_T \times H_T} \sum_{w=0}^{W_T-1} \sum_{w=0}^{H_T-1} x_{w,h,d}^{(T)}. \tag{2}$$

We use the **Caltech256** dataset which contains 30607 natural images to simulate the distribution. We simply assume that all the D_T elements in $\mathbf{x}^{(T)}$ are nearly independent, and summarize all the $30607 \times D_T$ elements by 1D histograms shown in Figure 2. We can observe that there are typically fewer neurons with large responses. Therefore, we can assume that the PDF of high-level neurons has the following form: $f\big(\mathbf{x}^{(T)}\big) = C_p \cdot \exp\Big\{-\big\|\mathbf{x}^{(T)}\big\|_p^p\Big\}$, where p is the *norm* and C_p is the normalization coefficient.

In statistics, the *score function* indicates how a likelihood function depends on its parameters. The score function has been used to produce discriminative features from generative models [19], *e.g.*, as of in Fisher vectors [35]. It is obtained by computing the gradient of the log-likelihood with respect to the parameters. Given an image $\mathbf{X}^{(0)}$, we compute the intermediate network output $\mathbf{X}^{(T)}$, the response vector $\mathbf{x}^{(T)}$ using (2), and the likelihood $f^{(T)} \doteq f\big(\mathbf{x}^{(T)}\big)$. Then we compute the score function with respect to $\boldsymbol{\theta}^{(t)}$ to measure the **activeness** of each network connection in $\boldsymbol{\theta}^{(t)}$:

$$\frac{\partial \ln f^{(T)}}{\partial \boldsymbol{\theta}^{(t)}} = \frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(t+1)}} \cdot \frac{\partial \mathbf{X}^{(t+1)}}{\partial \boldsymbol{\theta}^{(t)}}, \tag{3}$$

where $\mathbf{X}^{(t+1)}$ is taken as the intermediate term since it directly depends on $\boldsymbol{\theta}^{(t)}$. The two terms on the right-handed side are named the *layer-score* and the *inter-layer activeness*, respectively.

3.2.1 The Layer Score

We first compute the layer score $\frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(t+1)}}$. From the chain rule of differentiation we have:

$$\frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(t+1)}} = \frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(T)}} \cdot \frac{\partial \mathbf{X}^{(T)}}{\partial \mathbf{X}^{(t+1)}} \tag{4}$$

The second term on the right-handed side, *i.e.*, $\frac{\partial \mathbf{X}^{(T)}}{\partial \mathbf{X}^{(t+1)}}$, can be easily derived by network back-propagation as in the training process. The only difference is that the gradient on the top (T-th) layer is defined by $\frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(T)}}$. From $\mathbf{x}^{(T)}$ defined in (2) and $f^{(T)} = C_p \cdot \exp\left\{-\left\|\mathbf{x}^{(T)}\right\|_p^p\right\}$, we have:

$$\frac{\partial \ln f^{(T)}}{\partial \mathbf{X}^{(T)}} = -\frac{p}{W_T \times H_T} \cdot \left(\mathbf{x}^{(T)}\right)^{p-1} \cdot \frac{\partial \mathbf{x}^{(T)}}{\partial \mathbf{X}^{(T)}} \tag{5}$$

where $\left(\mathbf{X}^{(T)}\right)^{p-1}$ is the element-wise $(p-1)$-st power of the vector. In particular, when $p = 1$, the layer score is proportional to an all-one vector $\mathbf{1}^{W_T \times H_T \times D_T}$; when $p = 2$, each of the $W_T \times H_T$ sections is proportional $\mathbf{x}^{(T)}$.

3.2.2 The Inter-Layer Activeness

Next we compute the inter-layer activeness $\frac{\partial \mathbf{X}^{(t+1)}}{\partial \boldsymbol{\theta}^{(t)}}$. Consider a single term $\frac{\partial x^{(t+1)}_{w',h',d'}}{\partial \theta^{(t)}_{w,h,d,w',h',d'}}$, direct differentiation of (1) gives:

$$\frac{\partial x^{(t+1)}_{w',h',d'}}{\partial \theta^{(t)}_{w,h,d,w',h',d'}} = x^{(t)}_{w,h,d} \cdot \mathbb{I}_{x^{(t+1)}_{w',h',d'} > 0} \cdot \mathbb{I}_{(w',h',d') \in \mathcal{U}^{(t)}_{w,h,d}}, \tag{6}$$

where $\mathbb{I}.$ is the indicator whose value is 1 when the conditional term is true and 0 otherwise.

3.3. The Activeness of Neurons

With the layer score (5) and the inter-layer gradient (6), the score function with respect to $\boldsymbol{\theta}^{(t)}$ is derived to be:

$$\frac{\partial \ln f^{(T)}}{\partial \theta^{(t)}_{w,h,d,w',h',d'}} = x^{(t)}_{w,h,d} \cdot \alpha^{(t)}_{w,h,d,w',h',d'}, \tag{7}$$

where $\alpha^{(t)}_{w,h,d,w',h',d'}$ is the *importance* of the neuron $x^{(t)}_{w,h,d}$ to the connection between $x^{(t)}_{w,h,d}$ and $x^{(t+1)}_{w',h',d'}$:

$$\alpha^{(t)}_{w,h,d,w',h',d'} \doteq \mathbb{I}_{x^{(t+1)}_{w',h',d'} > 0} \cdot \mathbb{I}_{(w',h',d') \in \mathcal{U}^{(t)}_{w,h,d}} \cdot \frac{\partial \ln f^{(T)}}{\partial x^{(t+1)}_{w',h',d'}}.$$

Recall that the score function can be used as visual features. Therefore, we define the **activeness** of each neuron by accumulating the activeness of all the related connections:

$$\widetilde{x}^{(t)}_{w,h,d} = \sum_{(w',h',d') \in \mathcal{U}^{(t)}_{w,h,d}} \frac{\partial \ln f^{(T)}}{\partial \theta^{(t)}_{w,h,d,w',h',d'}}. \tag{8}$$

We summarize $\widetilde{\mathbf{X}}^{(t)} = \left\{\widetilde{x}^{(t)}_{w,h,d}\right\}^{W_t \times H_t \times D_t}$ with max-pooling (2), resulting in a D_t-dimensional **InterActive** feature vector $\widetilde{\mathbf{x}}^{(t)}$. As we will see in Section 4.2, $\widetilde{\mathbf{x}}^{(t)}$ is a discriminative representation of the input image $\mathbf{X}^{(0)}$.

The relationship between T and t can be arbitrary, provided it satisfies $T \geqslant t + 1$. In this paper, we consider two typical settings, *i.e.*, $T = L$ (L is the number of layers) and $T = t + 1$, which means that the supervision comes from the final layer (*i.e.*, *fc-7*) or its direct successor. We name them the *last* and the *next* configurations, respectively.

3.4. Visualization

Before using the InterActive features for experiments (Section 4), we note that $\widetilde{x}^{(t)}_{w,h,d}$ is a weighted version of the original neuron response $x^{(t)}_{w,h,d}$. The weighting term is:

$$\gamma^{(t)}_{w,h,d} = \sum_{(w',h',d') \in \mathcal{U}^{(t)}_{w,h,d}} \alpha^{(t)}_{w,h,d,w',h',d'}. \tag{9}$$

It counts the activated (*i.e.*, $x^{(t+1)}_{w',h',d'} > 0$) neurons on the $(t+1)$-st layer, with the importance $\frac{\partial \ln f^{(T)}}{\partial x^{(t+1)}_{w',h',d'}}$, which is supervised by a higher level (the T-th layer).

We visualize the weighting term $\gamma^{(t)}_{w,h,d}$ on the 2D image plane by defining $\widehat{\gamma}^{(t)}_{w,h} = \sum_d \gamma^{(t)}_{w,h,d}$. The weighting map is then resized to the original image size. Representative results are shown in Figure 3. We observe that spatial weighting weakly captures the interest regions, although the network is pre-trained using an independent set (*i.e.*, **ImageNet** [8]). Here, we discuss how different parameters affect the weighting terms.

First, activeness measures the contribution of each neuron to higher-level visual outputs. For a low-level neuron, if the supervision comes from the *next* layer, its receptive field is not significantly enlarged (*e.g.*, a neuron on the *pool-1* receives information from the *next* layer to increase the receptive field from 6 × 6 to 18 × 18). Therefore, it is more likely that local high-contrast regions becomes more activated, and the weighting map looks like boundary detection results. As t increases, neurons have larger receptive fields and capture less local details, thus the weighting map is more similar to saliency detection results.

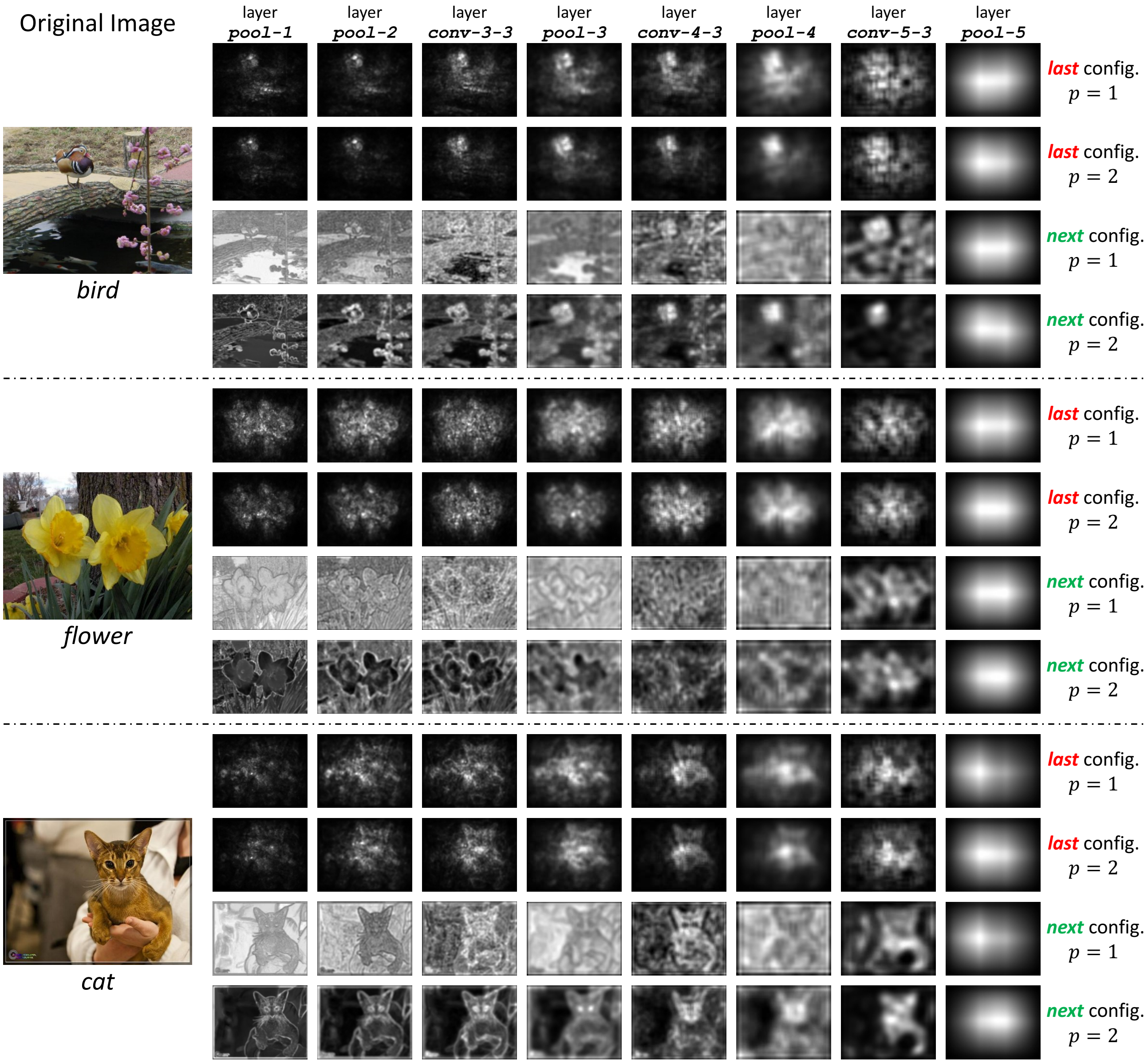

Figure 3. Typical visualization results of activeness $\widehat{\gamma}_{w,h}^{(t)}$ with different configurations. Neuron weighting maps are resized to the original image size for better visualization. Neurons with larger activeness are plotted with higher intensity values (closer to *white*). Regarding the *last* and *next* configurations, please refer to the texts in Section 3.4 for details.

Second, the *last* and *next* configurations make a big difference in activeness, especially for the low-level and mid-level neurons. Supervised by the top layer, the *last* configuration generates stable weighting maps, with the high-weight regions corresponding to the salient objects on the image. However, the output of the *next* configuration is quite sensitive to small noises, and sometimes the background regions even receive more attention than the semantic objects. As we will see in experiments (Section 4.2), the *last* configuration consistently produces higher recognition accuracy on the low-level and mid-level features.

We also compare different *norms*, *i.e.*, $p = 1$ vs. $p = 2$. When $p = 2$, spatial weighting rewards neurons with high responses more heavily, and the high-activeness regions become more concentrated. In general, p reflects the extent that we assume high-response neurons are more important. Although other p values can be used, we believe that $p = 1$ and $p = 2$ are sufficient to illustrate the difference and produce good performance. We also test $p \to +\infty$, which only considers the neuron with the maximal response, but the performance is inferior to that using $p = 1$ and $p = 2$.

3.5. Comparison to Related Works

Although both InterActive and network training involve gradient back-propagation, they are propagating different information. In the training process, a *supervised* loss function is defined by the difference between ground-truth and predicted outputs. In deep feature extraction, however, there is no ground-truth, so we define an *unsupervised* loss using the score function. Both methods lead to propagating high-level visual context through the network to enhance the descriptive power of low-level neurons.

Although our method and [54] share similar ideas, they are quite different. We focus on generating better image description, while [54] focuses on visualizing the network; we can visualize back-propagated neuron activeness, while [54] visualizes neuron responses; we back-propagate the activeness of all neurons, while [54] only chooses the neuron with maximal response; our method is unsupervised, while [54] is supervised (by "guessing" the label). Being unsupervised, InterActive can be generalized to many more classification problems with a different set of image classes.

In another work on object detection [2], the neural network is *told* a visual concept, and the supervised signal is back-propagated to find the most relevant neurons. InterActive performs detection in an implicit, unsupervised manner, making it feasible to be applied to image classification.

4. Experiments

4.1. Datasets and Settings

We evaluate InterActive on six popular image classification datasets. For generic object recognition, we use the **Caltech256** [16] (30607 images, 257 classes, 60 training samples for each class) dataset. For scene recognition, we use the MIT **Indoor-67** [37] (15620 images, 67 classes, 80 training samples per class) and the **SUN-397** [46] (108754 images, 397 classes, 50 training samples per class) datasets. For fine-grained object recognition, we use the Oxford **Pet-37** [34] (7390 images, 37 classes, 100 training samples per class), the Oxford **Flower-102** [33] (8189 images, 102 classes, 20 training samples per class) and the Caltech-UCSD **Bird-200** [43] (11788 images, 200 classes, 30 training samples per class) datasets.

We use the 19-layer **VGGNet** [40] (pre-trained on **ImageNet**) for deep features extraction. We use the model provided by the MatConvNet library [42] without fine-tuning. Its down-sampling rate is 32, caused by the five max-pooling layers. As described in Section 3.1, we maximally preserve the aspect ratio of the input image, constrain the width and height divisible by 32, and the number of pixels is approximately 512^2. The InterActive feature vectors are ℓ_2-normalized and sent to LIBLINEAR [10], a scalable SVM implementation, with the slacking parameter C fixed as 10.

4.2. InterActive Configurations

We evaluate the InterActive features extracted from different layers, using different *norms* p, and either the *last* or *next* configuration (please refer to Section 3.4 and Figure 3). We also compare InterActive with the original deep features with average-pooling or max-pooling. Classification results are summarized in Table 1.

We first observe the low-level and mid-level layers (from *pool-1* to *pool-4*). InterActive with the *last* configuration consistently outperforms the original deep features. Sometimes, the accuracy gain is very significant (*e.g.*, more than 30% on *conv-4-3* and *pool4* for *bird* recognition), showing that InterActive improves image representation by letting the low-level and mid-level neurons receive high-level context. Although these layers often produce low accuracy, the improvement contributes when multi-level features are combined (see Table 2). Regarding the *norm*, $p = 2$ always works better than $p = 1$. Recalling from (5) that $p = 2$ better rewards high-response neurons, we conclude that high-response neurons are indeed more important.

On the high-level neurons (*i.e.*, *pool-5* and *fc-6*), the advantage of InterActive vanishes in scene classification, and the original average-pooled features produce the best accuracy. Therefore, it is more likely that all the high-level neurons are equally important for scene understanding. On object recognition tasks, the advantage also becomes much smaller, since InterActive only provides limited increase on high-level neurons' receptive field.

The intermediate output of the t-th layer can be considered as a bunch of D_t-dimensional visual descriptors. Possible choices of feature aggregation include average-pooling and max-pooling. If each image region approximately contributes equally (such as in scene recognition), average-pooling produces higher accuracy, however in the case that semantic objects are quite small (such as on the **Bird-200** dataset), max-pooling works better. InterActive computes neuron activeness in an unsupervised manner, which provides a soft weighting scheme, or a tradeoff between max-pooling and average-pooling. By detecting interesting regions automatically, it often produces higher accuracy than both max-pooling and average-pooling.

4.3. Comparison to the State-of-the-Arts

We compare InterActive with several recent works in Table 2. These algorithms also extract features from statistics-based methods, and use machine learning tools for classification. We concatenate the feature vectors of all 9 layers in Table 1 as a 6848-dimensional vector. Apart from the **Bird-200** dataset, the reported accuracy is the highest, to the best of our knowledge. Although the accuracy gain over baseline is relatively small (*e.g.*, 0.43% in **Pet-37**), we emphasize that the baseline accuracy is already very high, thanks to the improved deep feature extraction strategy. Therefore,

Layer	Model	Dims	**Caltech256**	**Indoor-67**	**SUN-397**	**Pet-37**	**Flower-102**	**Bird-200**
pool-1	Orig., AVG	64	11.12	19.96	8.52	12.09	29.36	5.10
pool-1	Orig., MAX	64	8.77	16.82	7.27	14.83	27.95	7.81
pool-1	Next, $p = 1$	64	11.01	19.97	8.62	11.60	29.11	4.95
pool-1	Next, $p = 2$	64	11.26	19.71	8.92	12.38	31.07	5.30
pool-1	Last, $p = 1$	64	12.93	20.83	9.83	20.64	32.93	8.55
pool-1	Last, $p = 2$	64	**13.14**	**21.10**	**10.02**	**21.19**	**33.58**	**9.01**
pool-2	Orig., AVG	128	21.03	31.12	18.63	20.49	45.77	8.30
pool-2	Orig., MAX	128	19.47	28.29	16.05	24.60	43.39	11.28
pool-2	Next, $p = 1$	128	20.98	30.93	18.59	19.89	45.62	8.01
pool-2	Next, $p = 2$	128	20.65	30.95	19.01	21.18	48.27	9.60
pool-2	Last, $p = 1$	128	25.84	33.24	20.25	37.29	53.72	18.52
pool-2	Last, $p = 2$	128	**26.20**	**33.47**	**20.50**	**38.42**	**54.22**	**19.43**
conv-3-3	Orig., AVG	256	26.44	36.42	22.73	27.78	49.70	10.47
conv-3-3	Orig., MAX	256	24.18	33.27	19.71	31.43	48.02	13.85
conv-3-3	Next, $p = 1$	256	27.29	36.97	22.84	28.89	50.62	10.93
conv-3-3	Next, $p = 2$	256	27.62	37.36	23.41	30.38	54.06	12.73
conv-3-3	Last, $p = 1$	256	34.50	39.40	25.84	49.41	60.53	24.21
conv-3-3	Last, $p = 2$	256	**35.29**	**39.68**	**26.02**	**50.57**	**61.06**	**25.27**
pool-3	Orig., AVG	256	29.17	37.98	23.59	29.88	52.44	11.00
pool-3	Orig., MAX	256	26.53	34.65	20.83	33.68	50.93	13.66
pool-3	Next, $p = 1$	256	29.09	38.12	24.05	30.08	52.26	10.89
pool-3	Next, $p = 2$	256	29.55	38.61	24.31	31.98	55.06	12.65
pool-3	Last, $p = 1$	256	36.96	41.02	26.73	50.91	62.41	24.58
pool-3	Last, $p = 2$	256	**37.40**	**41.45**	**27.22**	**51.96**	**63.06**	**25.47**
conv-4-3	Orig., AVG	512	49.62	59.66	42.03	55.57	76.98	21.45
conv-4-3	Orig., MAX	512	47.73	55.83	40.10	59.40	75.72	23.39
conv-4-3	Next, $p = 1$	512	51.83	60.37	43.59	59.29	78.54	25.01
conv-4-3	Next, $p = 2$	512	53.52	60.65	44.17	63.40	80.48	31.07
conv-4-3	Last, $p = 1$	512	61.62	62.45	45.43	75.29	85.91	52.26
conv-4-3	Last, $p = 2$	512	**61.98**	**62.74**	**45.87**	**77.61**	**86.08**	**54.12**
pool-4	Orig., AVG	512	60.39	66.49	49.73	66.76	85.56	28.56
pool-4	Orig., MAX	512	57.92	62.96	47.29	69.23	84.39	30.01
pool-4	Next, $p = 1$	512	60.59	66.48	49.55	66.28	85.68	28.40
pool-4	Next, $p = 2$	512	62.06	66.94	50.01	72.40	87.36	37.49
pool-4	Last, $p = 1$	512	68.20	67.20	51.04	81.04	91.22	57.41
pool-4	Last, $p = 2$	512	**68.60**	**67.40**	**51.30**	**82.56**	**92.00**	**59.25**
conv-5-3	Orig., AVG	512	77.40	74.66	59.47	88.36	94.03	55.44
conv-5-3	Orig., MAX	512	75.93	71.38	57.03	87.10	91.30	55.19
conv-5-3	Next, $p = 1$	512	80.31	**74.80**	59.63	90.29	94.84	67.64
conv-5-3	Next, $p = 2$	512	80.73	74.52	**59.74**	**91.56**	95.16	**73.14**
conv-5-3	Last, $p = 1$	512	80.77	73.68	59.10	90.73	95.40	69.32
conv-5-3	Last, $p = 2$	512	**80.84**	73.58	58.96	91.19	**95.70**	69.75
pool-5	Orig., AVG	512	81.40	**74.93**	**55.22**	91.78	94.70	69.72
pool-5	Orig., MAX	512	79.61	71.88	54.04	89.43	90.01	68.52
pool-5	Next, $p = 1$	512	81.50	72.70	53.83	92.01	95.41	71.96
pool-5	Next, $p = 2$	512	81.58	72.63	53.57	**92.30**	95.40	**73.21**
pool-5	Last, $p = 1$	512	81.60	72.58	53.93	92.20	**95.43**	72.47
pool-5	Last, $p = 2$	512	**81.68**	72.68	53.79	92.18	95.41	72.51
fc-6	Orig., AVG	4096	83.51	**75.52**	**61.30**	93.08	**93.54**	**71.69**
fc-6	Orig., MAX	4096	83.59	74.47	59.39	93.07	93.20	71.03
fc-6	Last, $p = 1$	4096	83.44	75.48	61.28	92.84	93.40	70.26
fc-6	Last, $p = 2$	4096	**83.61**	75.50	61.19	**93.10**	93.45	71.60

Table 1. Classification accuracy (%) comparison among different configurations. **Bold** numbers indicate the best performance in each group (*i.e.*, same dataset, same layer). For *fc-6*, the *next* and *last* layers are the same (see the texts in Section 3.4 for details).

Model	**Caltech256**	**Indoor-67**	**SUN-397**	**Pet-37**	**Flower-102**	**Bird-200**
Murray *et.al.* [32]	–	–	–	56.8	84.6	33.3
Kobayashi *et.al.* [21]	58.3	64.8	–	–	–	30.0
Liu *et.al.* [28]	75.47	59.12	–	–	–	–
Xie *et.al.* [51]	60.25	64.93	50.12	63.49	86.45	50.81
Chatfield *et.al* [4]	77.61	–	–	–	–	–
Donahue *et.al* [9]	–	–	40.94	–	–	64.96
Razavian *et.al.* [38]	–	69.0	–	–	86.8	61.8
Zeiler *et.al* [54]	74.2	–	–	–	–	–
Zhou *et.al.* [57]	–	69.0	54.3	–	–	–
Qian *et.al.* [36]	–	–	–	81.18	89.45	67.86
Xie *et.al.* [48]	–	70.13	54.87	90.03	86.82	62.02
Ours (Orig., AVG)	84.02	78.02	62.30	93.02	95.70	73.35
Ours (Orig., MAX)	84.38	77.32	61.87	93.20	95.98	74.76
Ours (Next, $p = 1$)	84.43	78.01	62.26	92.91	96.02	74.37
Ours (Next, $p = 2$)	84.64	78.23	62.50	93.22	96.26	74.61
Ours (Last, $p = 1$)	84.94	78.40	62.69	93.40	96.35	75.47
Ours (Last, $p = 2$)	**85.06**	**78.65**	**62.97**	**93.45**	**96.40**	**75.62**

Table 2. Accuracy (%) comparison with recent works (published after 2014) without (above) and with (middle) using deep features. We use the concatenated feature vectors from all the 9 layers used in Table 1. For the **Bird-200** dataset, most competitors use extra information (bounding boxes and/or detected parts) but we do not. With bounding boxes, we achieve higher accuracy: **77.53**%. See texts for details.

the improvement of InterActive is not so small as it seems. On the other hand, recognition rates are consistently boosted with InterActive, without requiring extra information, which demonstrates that deep features can be intrinsically improved when neuron activeness is considered.

On the **Bird-200** dataset, it is very important to detect the position and/or compositional parts of the objects [3][13][55][47], otherwise heavy computation is required to achieve good performance [48]. InterActive implicitly finds the semantic object regions, leading to competitive 75.62% accuracy. If the bounding box of each object is provided (as in [55] and [26]), the original and InterActive features produce 76.95% and 77.53% accuracy, respectively. Using bounding boxes provides 3.60% and 1.91% accuracy gain on original and InterActive features, respectively. InterActive significantly reduces the gap with implicit object detection. 77.53% is lower than 80.26% in [26] and 82.8% in [22], both of which require fine-tuning the network and R-CNN part detection [15] while InterActive does not. We believe that InterActive can cooperate with these strategies.

4.4. ImageNet Experiments

We report results on **ILSVRC2012**, a subset of **ImageNet** which contains 1000 categories. We use the pre-trained **VGGNet** models and the same image cropping techniques as in [40]. The baseline validation error rates on the 16-layer model, the 19-layer model and the combined model are 7.1%, 7.0% and 6.7%, respectively (slightly better than [40]). We apply InterActive to update the neuron responses on the second-to-last layer (*fc-7*) and forward-propagate them to re-compute the classification scores (*fc-8*). The error rates are decreased to 6.8%, 6.7% and 6.5%, respectively. The improvement is significant given that the baseline is already high and our method is very simple.

In the future, we will explore the use of InterActive on some challenging datasets, such as the **PASCAL-VOC** dataset and the **Microsoft COCO** dataset [27]. We thank the anonymous reviewers for this valuable suggestion.

5. Conclusions

In this paper, we present InterActive, a novel algorithm for deep feature extraction. We define a probabilistic distribution function on the high-level neuron responses, and back-propagate the score function through the network to compute the *activeness* of each network connection and each neuron. We reveal that high-level visual context carries rich information to enhance low-level and mid-level feature representation. The output of our algorithm is the activeness of each neuron, or a weighted version of the original neuron response. InterActive improves visual feature representation, and achieves the state-of-the-art performance on several popular image classification benchmarks.

InterActive can be applied to many more vision tasks. On the one hand, with the *last* configuration, neuron activeness provides strong clues for saliency detection. On the other hand, with the *next* configuration on a low-level layer, neuron activeness can be used to detect local high-contrast regions, which may correspond to edges or boundaries. All these possibilities are left for future research.

References

[1] O. Boiman, E. Shechtman, and M. Irani. In Defense of Nearest-Neighbor Based Image Classification. *Computer Vision and Pattern Recognition*, 2008.

[2] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, and W. Xu. Look and Think Twice: Capturing Top-Down Visual Attention with Feedback Convolutional Neural Networks. *International Conference on Computer Vision*, 2015.

[3] Y. Chai, V. Lempitsky, and A. Zisserman. Symbiotic Segmentation and Part Localization for Fine-Grained Categorization. *International Conference on Computer Vision*, 2013.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *British Machine Vision Conference*, 2014.

[5] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-Based Models for Speech Recognition. *Advances in Neural Information Processing Systems*, 2015.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *Workshop on Statistical Learning in Computer Vision, European Conference on Computer Vision*, 1(22):1–2, 2004.

[7] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[8] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *Computer Vision and Pattern Recognition*, 2009.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *International Conference on Machine Learning*, 2014.

[10] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[11] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[12] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-norm Feature Pooling for Image Classification. *Computer Vision and Pattern Recognition*, 2011.

[13] E. Gavves, B. Fernando, C. Snoek, A. Smeulders, and T. Tuytelaars. Fine-Grained Categorization by Alignments. *International Conference on Computer Vision*, 2013.

[14] R. Girshick. Fast R-CNN. *International Conference on Computer Vision*, 2015.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Computer Vision and Pattern Recognition*, 2014.

[16] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. *Technical Report: CNS-TR-2007-001, Caltech*, 2007.

[17] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv preprint, arXiv: 1207.0580*, 2012.

[18] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, 2015.

[19] T. Jaakkola, D. Haussler, et al. Exploiting Generative Models in Discriminative Classifiers. *Advances in Neural Information Processing Systems*, pages 487–493, 1999.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. CAFFE: Convolutional Architecture for Fast Feature Embedding. *ACM International Conference on Multimedia*, 2014.

[21] T. Kobayashi. Three Viewpoints Toward Exemplar SVM. *Computer Vision and Pattern Recognition*, 2015.

[22] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-Grained Recognition without Part Annotations. *Computer Vision and Pattern Recognition*, 2015.

[23] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 2012.

[24] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *Computer Vision and Pattern Recognition*, 2006.

[25] Y. LeCun, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 1990.

[26] D. Lin, X. Shen, C. Lu, and J. Jia. Deep LAC: Deep Localization, Alignment and Classification for Fine-grained Recognition. *Computer Vision and Pattern Recognition*, 2015.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision*, 2014.

[28] Q. Liu and C. Liu. A Novel Locally Linear KNN Model for Visual Recognition. *Computer Vision and Pattern Recognition*, 2015.

[29] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *Computer Vision and Pattern Recognition*, 2015.

[30] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal on Computer Vision*, 60(2):91–110, 2004.

[31] V. Mnih, N. Heess, and A. Graves. Recurrent Models of Visual Attention. *Advances in Neural Information Processing Systems*, 2014.

[32] N. Murray and F. Perronnin. Generalized Max Pooling. *Computer Vision and Pattern Recognition*, 2014.

[33] M. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[34] O. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and Dogs. *Computer Vision and Pattern Recognition*, 2012.

[35] F. Perronnin, J. Sanchez, and T. Mensink. Improving the Fisher Kernel for Large-scale Image Classification. *European Conference on Computer Vision*, 2010.

[36] Q. Qian, R. Jin, S. Zhu, and Y. Lin. Fine-Grained Visual Categorization via Multi-stage Metric Learning. *Computer Vision and Pattern Recognition*, 2015.

[37] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. *Computer Vision and Pattern Recognition*, 2009.

[38] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *Computer Vision and Pattern Recognition*, 2014.

[39] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *Workshop of International Conference on Learning Representations*, 2014.

[40] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2015.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *Computer Vision and Pattern Recognition*, 2015.

[42] A. Vedaldi and K. Lenc. MatConvNet-Convolutional Neural Networks for MATLAB. *ACM International Conference on Multimedia*, 2015.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report: CNS-TR-2011-001, Caltech*, 2011.

[44] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

[45] F. Xia, J. Zhu, P. Wang, and A. Yuille. Pose-Guided Human Parsing by an AND/OR Graph Using Pose-Context Features. *AAAI Conference on Artificial Intelligence*, 2016.

[46] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN Database: Large-Scale Scene Recognition from Abbey to Zoo. *Computer Vision and Pattern Recognition*, 2010.

[47] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The Application of Two-Level Attention Models in Deep Convolutional Neural Network for Fine-Grained Image Classification. *Computer Vision and Pattern Recognition*, 2015.

[48] L. Xie, R. Hong, B. Zhang, and Q. Tian. Image Classification and Retrieval are ONE. *International Conference on Multimedia Retrieval*, 2015.

[49] L. Xie, Q. Tian, M. Wang, and B. Zhang. Spatial Pooling of Heterogeneous Features for Image Classification. *IEEE Transactions on Image Processing*, 23(5):1994–2008, 2014.

[50] L. Xie, Q. Tian, and B. Zhang. Simple Techniques Make Sense: Feature Pooling and Normalization for Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015.

[51] L. Xie, J. Wang, W. Lin, B. Zhang, and Q. Tian. RIDE: Reversal Invariant Descriptor Enhancement. *International Conference on Computer Vision*, 2015.

[52] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian. DisturbLabel: Regularizing CNN on the Loss Layer. *Computer Vision and Pattern Recognition*, 2016.

[53] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. *Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.

[54] M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *European Conference on Computer Vision*, 2014.

[55] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based R-CNNs for Fine-Grained Category Detection. *European Conference on Computer Vision*, 2014.

[56] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object Detectors Emerge in Deep Scene CNNs. *International Conference on Learning Representations*, 2015.

[57] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition Using Places Database. *Advances in Neural Information Processing Systems*, 2014.

[58] J. Zhu, W. Zou, X. Yang, R. Zhang, Q. Zhou, and W. Zhang. Image Classification by Hierarchical Spatial Pooling with Partial Least Squares Analysis. *British Machine Vision Conference*, 2012.