Expert Systems With Applications 95 (2018) 43–56

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# A new image classification method using CNN transfer learning and web data augmentation

Dongmei Han[a,b], Qigang Liu[a,*], Weiguo Fan[c]

[a] School of Information Management & Engineering, ShangHai University of Finance and Economics, 777 Guoding Road, ShangHai City 200433, China
[b] Shanghai Key laboratory of Financial Information Technology, 777 Guoding Road, ShangHai City 200433, China
[c] Accounting and Information Systems Department, Virginia Tech, 3007 Pamplin Hall, Blacksburg, Virginia 24061, USA

A R T I C L E   I N F O

*Article history:*
Received 23 February 2017
Revised 26 October 2017
Accepted 11 November 2017
Available online 13 November 2017

*Keywords:*
Feature transferring
Data augmentation
Convolutional neural network
Feature representation
Parameter fine-tuning
Bayesian optimization

A B S T R A C T

Since Convolutional Neural Network (CNN) won the image classification competition 202 (ILSVRC12), a lot of attention has been paid to deep layer CNN study. The success of CNN is attributed to its superior multi-scale high-level image representations as opposed to hand-engineering low-level features. However, estimating millions of parameters of a deep CNN requires a large number of annotated samples, which currently prevents many superior deep CNNs (such as AlexNet, VGG, ResNet) being applied to problems with limited training data. To address this problem, a novel two-phase method combining CNN transfer learning and web data augmentation is proposed. With our method, the useful feature presentation of pre-trained network can be efficiently transferred to target task, and the original dataset can be augmented with the most valuable Internet images for classification. Our method not only greatly reduces the requirement of a large training data, but also effectively expand the training dataset. Both of method features contribute to the considerable over-fitting reduction of deep CNNs on small dataset. In addition, we successfully apply Bayesian optimization to solve the tuff problem, hyper-parameter tuning, in network fine-tuning. Our solution is applied to six public small datasets. Extensive experiments show that, comparing to traditional methods, our solution can assist the popular deep CNNs to achieve better performance. Particularly, ResNet can outperform all the state-of-the-art models on six small datasets. The experiment results prove that the proposed solution will be the great tool for dealing with practice problems which are related to use deep CNNs on small dataset.

## 1. Introduction

With the advent of large-scale category-level training data, such as ImageNet (Deng et al., 2009), and efficient overfitting preventing technique (“dropout”) (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), deep convolutional neural network (CNN) has outperformed all known methods in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)2012 (Russakovsky et al., 2015). The main advantage of CNN for images classification is that the entire system is trained end-to-end, from raw pixels to ultimate categories, which alleviates the requirement for manually designing a suitable feature extractor. While the main disadvantages of CNN are: (i) a large amount of labeled training samples is required for weight parameters learning; (ii) a powerful GPU is needed to accelerate the learning process. Sometimes, people who have no such computational power, time and large-scale training set unfortunately cannot take advantage of the powerful CNN. The popular deep CNN architecture (shown in Fig. 1) comes from (Krizhevsky, Sutskever, & Hinton, 2012) composed of five convolutional layers and three fully-connected layers with a final soft-max classifier, and contains more than 60 million parameters. Some deeper networks, such as 16 and 19 layers (Simonyan & Zisserman, 2014), 22 layers (Szegedy et al., 2015) which can achieve better performance contain even more parameters. Directly learning so many parameters from only thousands of training samples will result in serious over-fitting even though the over-fitting preventing technique is applied. Therefore, there is a challenge on how to make the deep CNN fit to small dataset while keep the similar performance as on large-scale dataset.

The common solution for using deep CNNs on small dataset is parameters transfer learning, which drop the classifier layer of a pre-trained CNN and fine-tune it on target dataset. Practically, this is a very effective method. While, when this method is applied to deal with a specific problem, there are still some questions bother us: how to set the hyper-parameters (e.g. learning rate) for network fine-tuning, how to decide the fine-tuning strat-

* Corresponding author.
*E-mail addresses:* handongmei_shufe@163.com (D. Han), ryan.liu@shu.edu.cn (Q. Liu), wfan@vt.edu (W. Fan).

https://doi.org/10.1016/j.eswa.2017.11.028
0957-4174/© 2017 Elsevier Ltd. All rights reserved.

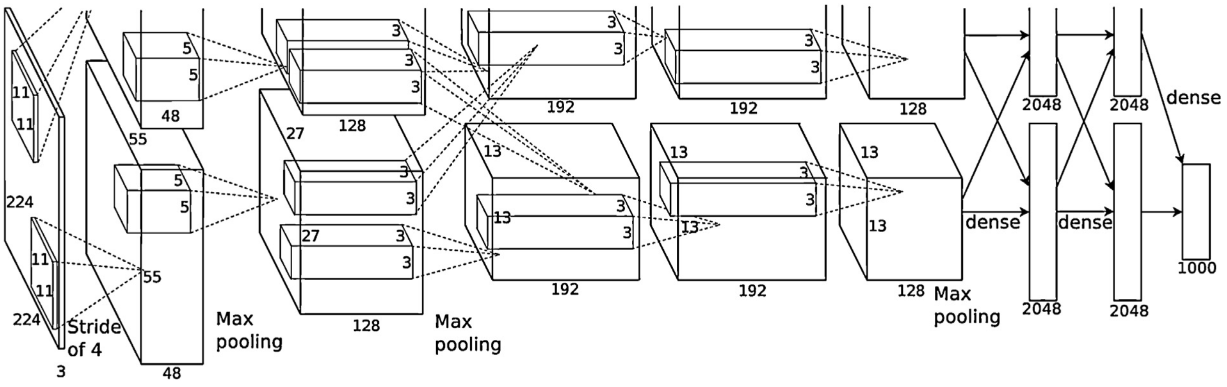

**Fig. 1.** This figure shows the detailed information about the architecture of AlexNet from (Krizhevsky et al., 2012).

egy, are there other methods to improve the network performance. In this work, we want to further expand the pre-training plus fine-tuning method. Apart from employing Bayesian optimization algorithm (BOA) for tuning the hyper-parameters for network fine-tuning, we develop a new web data augmentation method for augmenting the original small dataset. The web data augmentation is originated from traditional data augmentation techniques, such as rotation, translation, zoom, flips, shear, mirror and color perturbation (Flusser & Suk, 1993), which address the limited training data issue by enriching the training set with transformed original samples. For images with the small intra-class variability (such as changing the view point, flipping, cropping, making color changes or affine transformations) in digital number, plankton or face images, the traditional data augmentation techniques could improve the model performance. While, for complicated images (i.e. images about indoor scene or street scene), these techniques are not very effective as shown in our experiments.

Specifically, our method is composed of two phases. The phase one focuses on building a strong classifier by transfer learning for filtering massive images from the Internet in phase two. The phase two focuses on augmenting the training data with web images and the classifier learned in phase one. With this two-phase method, the serious over-fitting will be reduced, the classification accuracy will be improved. With BOA, the hyper-parameters tuning process will become sample. These are the strengths of our solution. The weakness is the operations of our method are a little complicated and time-consuming. With the rapid growth of the Internet images resource, the potential of our method will be further strengthened.

For a conclusion the contributions of this work are:

(i) we successively apply Bayesian optimization algorithm to hyper-parameters tuning for deep network fine-tuning.
(ii) we propose an effective augmentation method which can leverage the extensive Internet resources.
(iii) we propose an integration solution which integrates transfer learning, BOA based hyper-parameter tuning and web data augmentation into one workflow.

The rest of the paper is organized as follows. In Section 2, some closely related works about transfer learning and data augmentation are reviewed. Next, a method integrating CNN transfer learning with web images based data augmentation is presented in Section 3. The detailed experimental methodology and experimental results and analysis are shown in Section 4 followed by conclusions in Section 5.

## 2. Related Works

Our proposed method is directly related with transfer learning and data augmentation. The followings are the relative literature review in these two areas.

### 2.1. Transfer learning

Transfer learning which aims to learn from related tasks has attracted more and more attention recently. Research of this field begins with Thrun (1996) which is the first discussion about the role of previously learned knowledge played in generalization, particularly when training data is scarce. The later well-known work (Ando & Zhang, 2005) tries to discover a common structure of the hypothesis spaces shared by the multiple tasks with the purpose of transferring knowledge from supervised learning to unsupervised learning. Ling et al. (2008) propose an information bottleneck based approach to address this cross-language classification problem. Yang, Chen, Xue, Dai, and Yu. (2009) present a new learning scenario, heterogeneous transfer learning, which improves learning performance when the data can be in different feature. Xue, Dai, Yang, and Yu (2008) propose a novel cross-domain text classification algorithm which extends the traditional probabilistic latent semantic analysis (PLSA) algorithm to integrate labeled and unlabeled data, which come from different but related domains, into a unified probabilistic model. Tian, Tao and Rui (2012) try to implement transfer learning by mapping source domain and target domain into a new space to reduce the bias caused by cross-domain correspondence, which resorts to sparse coding method to find a proper mapping space. This approach is based on the strong hypothesis that the sample distributions of two domains are very similar. There are also some researches focusing on CNN based transfer learning. Yosinski, Clune, Bengio, and Lipson (2014) study extensively the transferability of features pre-learned from ImageNet dataset by employing different fine-tuning strategies on other datasets, and demonstrate that the transferability of features decreased when the distance between the base task and target task increases. Donahue et al. (2014) illustrate the transferability of each layer parameters of a pre-trained AlexNet by fine-tuning the network layer by layer.

Among these works about transfer learning, most of them deal with the text data from different language or different domain. The last two studies are related to transfer learning in image field. The pre-training plus fine-tuning method proposed by Yosinski et al. (2014) is very popular. This work can be seen as an extension of Yosinski et al. (2014). We particularly focus on dealing with the hyper-parameter tuning problem for fine-tuning a pre-trained network. For getting an overview of all methods, we list the technical features and suitable case of each method in Table 1.

**Table 1**
The technical feature and suitable case of above transfer learning methods.

| Literature | Method type | Technical features | Suitable case |
|---|---|---|---|
| Ando and Zhang (2005) | Semi-supervised learning | Based on learning predictive functional structures and Good Hypothesis Spaces. | The auxiliary data and the source data share a common predictive structure, the auxiliary task data is unlabeled, the source data is labeled. The data is text data. |
| Dai e al. (2007) | Supervised learning | Based on imbedded feature. Employing Nearest neighborhood algorithm and feature translation technique. | The auxiliary data and source data could come from different domains and different feature space. The auxiliary data is text data, while the source data is image data. |
| Xue et al. (2008) | Unsupervised learning | Based on data feature. Generating the mutual representation by clustering on the mixture of auxiliary data and source data. | The auxiliary data and source data could come from different domains. The data is unlabeled. The auxiliary data is much more than source data. The data is text data. |
| Ling et al. (2008) | Supervised learning | Based on data feature. Using the co-clustering algorithm to generate the mutual representation for data. | The auxiliary data and source data could come from different domains, but they share the feature space. The data is text data. |
| Yang et al. (2009) | Supervised learning | Based on data instance. Filtering out the irrelevant auxiliary data. | The auxiliary data and source data come from the same domain and are very similar. Only part auxiliary data is helpful for boosting the model performance. The data is text data. |
| Tian, Tao, and Rui (2012) | Unsupervised learning | Based on feature mapping. Using sparse dimension reduction to find the proper mapping space. | The auxiliary data and source data share the same mapping space. The data is text data. |
| Donahue et al. (2014) | Supervised learning | Based on feature fine-tuning. Using clustering and visualization to study the transferability. | The auxiliary data and source data could come from different domain. The fine-tuning strategy is hard to decide. The data is image data. |
| Yosinski et al. (2014) | Supervised learning | Based on fine-tuning different level feature of pre-trained deep neural network on target domain. | The auxiliary data and source data could come from different domain, the auxiliary dataset is large and the source dataset is small. The data is image data. |

### 2.2. Image data augmentation

Data augmentation technique is used to expand the existing data set, and it can be implemented in many ways, such as linear or non-linear transformation, adding auxiliary variable, simulation based on dynamic system or evolutionary system, data generation based on generative model. In image processing field, the simplest approach for data augmentation are adding noise and applying affine transformations (translation, zoom, flips, shear, mirror and color perturbation) (Flusser & Suk, 1993) on existing data. These basic forms of data augmentation have been widely used on small datasets for combatting over-fitting. In the work (Nair & Hinton, 2010), two forms of data augmentation are applied to training data. One is random crop and horizontal reflections, while the other is altering the intensities of the RGB channels by applying PCA on color space. With the two forms of data augmentation, AlexNet successfully avoids substantial over-fitting and gains accuracy improvement. Yosinski, Clune, Nguyen, Fuchs, and Lipson (2015) report that, for MNIST (a dataset containing handwritten digit images), augmenting the training set with randomly (by at most 5%) translated images greatly decreases the error from 28% to 20%. In the work (Simonyan & Zisserman, 2014), elastic deformation with random interpolations, a new form of data augmentation, is proposed and helps model achieve optimal performance on MNIST. Goodfellow et al. (2014) proposed an interesting Generative Adversarial Networks (GANs) which is composed of two competing neural network models. The purpose of the competition is to drive the generated artificial data to be indistinguishable from real data. The experiments show that GANs trained on MNIST can generate high quality artificial digit images. Ratner, Ehrenberg, Hussain, Dunnmon, and Ré, (2017) propose a method for automating the data augmentation process by learning a generative sequence model over user-specified transformation functions using a generative adversarial approach. Their method can make use of arbitrary, non-deterministic transformation functions, is robust to miss specified user input, and is trained on unlabeled data. DeVries and Taylor (2017) adopt a simpler, domain-agnostic approach to dataset augmentation. They start with existing data points and apply simple transformations such as adding noise, interpolating, or extrapolating between them. Their main insight is to perform the transformation not in input space, but in a learned feature space.

All these researches show that the transformation based data augmentation helps improve the final results. However, these experiments are almost all based on simple images (i.e. number, face or animal). When it comes to complicated images (i.e. outdoor or indoor scenes, images with complicated background), the effects of these forms of data augmentation are not obvious, which is proved in our experiments. Although GANs related methods can generate valuable artificial images, the generated images are only of small sized, $28 \times 28$ or $64 \times 64$. So far, GANs is incapable of generating high resolution image, such as $256 \times 256$. The limitation of these augmentation methods motivates us to develop a more effective data augmentation method. For getting an overview of all methods, we c list the technical features and suitable case of each method in Table 2.

### 2.3. CNN overview

In this section we will explain mathematically the training process of CNN in which number of parameters should be learned. The explanation is based on the knowledge of a standard ANN. For more details please refer to the work (Mitchell, 1997).

#### 2.3.1. Forward propagation

Different from Multi-layer perceptron neural networks (MLPNN) whose layers are all fully-connected layers, CNN is composed of convolutional layers, pooling layers and fully-connected layers. The convolutional layer, followed by a pooling layer (shown as in Fig. 1), composes of several convolution kernels which are used to compute different feature maps. The $j$th feature map of $l$th convo-

**Table 2**
The technical feature and suitable case of above data augmentation methods.

| Literature | Technical features | Suitable case |
|---|---|---|
| Flusser and Suk (1993) | Affine transformations (translation, zoom, flips, shear, mirror). | All images contain similar objects with some view angle variations. The images are sample images. |
| Nair and Hinton (2010) | Color jittering, PCA Jittering. | Images suffer many color or content variations. The images are complicated images. |
| Simonyan and Zisserman (2014) | Elastic deformation with random interpolations. | All images contain similar object with many object shape variations. |
| Goodfellow et al. (2014) | Generating Artificial images by generative and adversarial networks. | Images suffer all types of variations. Dataset size is large. |
| DeVries and Taylor (2017) | Doing transformation in feature space of deep layer rather than input space. | Images suffer feature space variations. Dataset size is large. |
| Ratner et al. (2017) | learning a generative sequence model over user-specified transformation functions using a generative adversarial approach. | Images suffer all types of variations. Dataset size is large. |

lutional layer, $x_j^l$, is calculated by:

$$a_j^l = \sum_{i=1}^{N^{l-1}} K_{ij}^l * x_i^{l-1} + b_j^l, \quad x_j^l = f(a_j^l) \tag{1}$$

where $x_i^{l-1}$ is the $i$th feature map of ($l$-1)th layer, $N^{l-1}$is the number of feature maps of this layer, $K_{ij}^l$ is the convolution kernel corresponding to $j$th map between $l$th layer and $i$th map in ($l$-1)th layer, $b_j^l$ is the bias term of the above kernel, $f(\cdot)$ denotes the element-wise non-linear activation function which introduces non-linearity into the multi-layer networks. The typical activation functions are sigmoid, tanh and ReLU (Glorot, Bordes, & Bengio, 2011).

After a convolutional layer, there is the pooling layer which aims to achieve integrating features, reducing parameters and shift-invariance by the reduction of the resolution of the feature maps. Denoting the pooling function as *downsample*$(\cdot)$, for each feature map $x_j^l$ we have:

$$s_j^l = downsample(x_j^l) \tag{2}$$

The typical pooling operations are average pooling and max pooling (Boureau, Ponce, & LeCun, 2010). Pooling operation simply takes some $k \times k$ region and outputs a single value, which is the mean or maximum in that region.

After the convolutional layer and pooling layer, in some cases (e.g. AlexNet, LeNet, VGG), there are several fully-connected layers which aim to learn mid-level feature maps, large number of weight parameters is required for implementing full connection. The feed forward process is identical with the standard ANN, which is formulated as:

$$a_j^l = \sum_{i=1}^{N^{l-1}} x_i^{l-1} W_{ij}^l + b_j^l, \quad x_j^l = f(a_j^l) \tag{3}$$

where $W_{ij}^l$ and $b_j^l$ are the weight vector and bias term of the $i$-th filter of the $l$-th layer respectively. The Softmax activation is normally applied to the very last layer in a neural net, which converts the output of the last layer into an essential probability distribution. It is responsible for predicting the class label for input data. let $o_i$ and $y_i$ denote the predicted label and ground-truth label for input sample separately. The loss function is usually formulated as:

$$E = \frac{1}{2} \sum_{i=1}^{N^L} \|y_i - o_i\|^2, \quad o_i = x_i^L \tag{4}$$

where $N^L$ is the number of nodes of $L$th layer, the last output layer, and $E$ represents the total classification error of all output nodes for a single sample. Loss function in the form of Eq. (4) is called Euclidean Loss. There are other alternatives for it, such as Contrastive loss, Hinge Loss, InformationGain Loss, Sigmoid Cross Entropy Loss, Softmax With Loss. Refer to the work (Lowe, 1999) for details.

### 2.3.2. Backward propagation

In the backward propagation, errors originated from label prediction in the output layer will be propagated to the input layer. According to these errors, the weight vectors and bias term can be updated layer by layer. The update for these parameters is formulated as:

$$W_{ij}^l = W_{ij}^{l-1} + \eta \frac{\partial E}{\partial W_{ij}^{l-1}}, \quad b_i^l = b_i^{l-1} + \eta \frac{\partial E}{\partial b_i^{l-1}} \tag{5}$$

where $\eta$ is the learning rate, $\partial E / \partial W_{ij}^l$ and $\partial E / \partial b_i^l$ are the partial derivatives of the loss function w.r.t $W_{ij}^l$ and $b_i^l$ separately which can be expanded to the format as:

$$\frac{\partial E}{\partial W_{ij}^l} = \frac{\partial E}{\partial a_i^l} \frac{\partial a_i^l}{\partial W_{ij}^l}, \quad \frac{\partial E}{\partial b_i^l} = \frac{\partial E}{\partial a_i^l} \frac{\partial a_i^l}{\partial b_i^l} \tag{6}$$

Let $\delta_i^l$ denotes the first part of right hand side of Eq. (6) as the error term on $l$-th layer, and combines it with the result of second part, Eq. (6) can be reformulated as:

$$\frac{\partial E}{\partial W_{ij}^l} = \delta_i^{l+1} f'(a_i^l) x_i^l, \quad \frac{\partial E}{\partial b_i^l} = \delta_i^{l+1} f'(a_i^l) \tag{7}$$

If the $l$th layer is fully-connected layer and layer $l+1$ is the output layer, the error term $\delta_i^l$ is computed as:

$$\delta_i^l = \frac{\partial}{\partial a_i^{l-1}} \frac{1}{2} \sum_{i=1}^{N^{l+1}} \|y_i - o_i\|^2 = -(y_i - x_i^l) f'(a_i^{l-1}) \tag{8}$$

where $f'(x_i^l)$ is the derivative of the activation function of layer $l$. If layer $l$ and $l+1$ are all convolutional layers, following the chain rule, the error term $\delta_i^l$ is computed as:

$$\delta_i^l = \left( \sum_{j=1}^{N^{l+1}} W_{ji}^l \delta_j^{l+1} \right) f'(a_i^{l-1}) \tag{9}$$

If the $l$-th layer is a pooling layer and layer $l+1$ is a convolutional layer, the error $\delta_i^l$ is computed as:

$$\delta_i^l = \left( \sum_{j=1}^{N^{l+1}} K_{ji} * \delta_i^{l+1} \right) f'(x_i^l) \tag{10}$$

where $f'(x_i^l)$ is the derivative of pooling function $f(x_i^l)$ which is a linear function (mean or max function). So, the derivative term

**Table 3**
The details of parameters amount and distribution among each layer.

| Layer | Kernel size | Number of kernels | Number of bias | Total |
|---|---|---|---|---|
| Conv1 | $11 \times 11 \times 3$ | 96 | 96 | 34,944 |
| Conv2 | $5 \times 5 \times 48$ | 256 | 256 | 307,456 |
| Conv3 | $3 \times 3 \times 256$ | 384 | 384 | 885,120 |
| Conv4 | $3 \times 3 \times 192$ | 384 | 384 | 663,936 |
| Conv5 | $3 \times 3 \times 192$ | 256 | 256 | 442,624 |
| FC1 | $6 \times 6 \times 256 \times 4096$ | 1 | 4096 | 37,752,832 |
| FC2 | $4096 \times 4096$ | 1 | 4096 | 16,781,312 |
| FC3 | $4096 \times 1000$ | 1 | 1000 | 4,097,000 |
| Total | | | | **60,965,224** |

$f'(x_i^l)$ to 1, then the last term of Eq. (10) will disappear. If the $l$-th layer is convolutional layer and layer $l$+1 is a pooling layer, the error $\delta_i^l$ is computed as:

$$\delta_i^l = upsample(\delta_i^{l+1}) f'(x_j^l) \tag{11}$$

where *upsample*() represents upsampling operation. If the pooling layer adopts mean pooling, then upsampling uniformly distributes the error among the units which feed into it in the previous layer. In max pooling the unit which is chosen as the max receives all the error, since very small changes in input would perturb the result only through that unit. Following the up-down direction, the weight vector and bias term can be updated based on previous update.

#### 2.3.3. Parameters counting with AlexNet as an example

For a deep CNN usually a large set parameters need to be learned. In this section, a detailed explanation on the parameter amount and distribution among each layer will be offered, with AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) as an example. The architecture of AlexNet is illustrated in Fig. 1, which is composed of five convolutional layers and three fully-connected layers. The kernel size and kernel number of each convolutional layer and the size of weight matrix of each fully-connected layer are listed in the following Table 3:

The total number of each line is calculated as: S(size of kernel) × N(number of kernel)+B(number of bias), the last line of Table 1 shows that there are more than 60 million parameters to be learned from training data, which is the reason why deep CNN get over-fitted easily on the small or medium size training data.

## 3. Our novel two-phase classification method

The method we proposed is composed of two phases, phase one focuses on building a strong classifier with current training data. Phase two focuses on augmenting dataset with help of classifier obtained in first phase. Details for each phase will be offered in following two subsections.

### 3.1. Phase one: implementation of CNN parameter transferring

#### 3.1.1. Parameter transferring roadmap

As mentioned in Section 1, images from different datasets share common low-level features which mainly come from convolutional layers. If there are not enough training data, there is no need to learn the weight parameters for these layers again. The learning process for the new dataset can be started based on the parameters learned from the other dataset. What we need to do on the new dataset is to fine-tune the pre-learned parameters using the limited training data. The above-mentioned process is the roadmap that implements parameter transferring from the source dataset to the target dataset.

#### 3.1.2. Obtain a pre-trained network

To obtain the pre-trained weight parameters, we build an AlexNet based on Caffe (Jia et al., 2014), an efficient framework which is implemented in C++ and takes advantage of powerful computation of GPU. We train the AlexNet for two days, 3 million iterations, on ILSVRC2012 dataset which contains 1.4 million labeled data. Finally, the classification top-1 error rate is 41.6% very close to 40.7% in (Krizhevsky, Sutskever, & Hinton, 2012). Then we build a new AlexNet and initialize it with the learned parameters to implement parameters transferring. The advantage of this pre-trained network is that it inherits the superior capacity of fully trained AlexNet, while the disadvantage is that we cannot change the architecture when applying it to other datasets.

#### 3.1.3. Fine-tune the pre-trained network

To make the pre-trained network adapt well to the target dataset, the pre-trained network will be fine-tuned on the target dataset. The fine-tune process is similar to the training process, but the strategies for hyper-parameter setting and searching are different: (i) in the fine-tune process, the learning rate which be chosen by Bayesian optimization in the next subsection should be much smaller than 0.1 which is a common learning rate for learning from scratch (Krizhevsky, Sutskever, & Hinton, 2012); (ii) not all weight parameters should be involved in the fine-tune process since the training data is limited. The effective way for fine-tuning the pre-trained parameters is to make the learning process happen only in the fully-connected layers which are closely related to specific image category, and keep parameters of convolutional layers frozen, which could be implemented by setting a positive value to the learning rate of fully-connected layers and 0 to the learning rate of convolutional layers.

#### 3.1.4. Hyper-parameter tuning during fine-tuning

Hyper-parameter search is usually critical for achieving the best performance of algorithms and is a time-consuming process. In deep learning, some superior optimizers that can tune the parameter automatically during training have been proposed, such as Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), RMSprop (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2014). The recommended learning rate for these methods according Keras documentation are 0.01, 1, 0.001 and 0.001 separately. Actually, these methods are designed for a deep CNN training from scratch. However, when a pre-trained CNN fine-tunes on a small size dataset, the situation is very different. In addition to the limited training data, only part of the model needs to be updated, which may lead to the training process only involving thousands or tens of thousands of iterations (as is case with our experiments). Therefore, the recommended learning rate is unsuitable for fine-tuning process. As we kown, in fine-tuning process, the learning rate should be a lower value, but it is hard to choose a proper initial learning rate to get the CNN model well trained. To address this problem, Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1999) is adopted to facilitate the learning rate search because of its high efficiency compared to grid search and random search.

In grid search and random search, the next point $x^*$ to evaluate is selected by certain forward step or by random. In BOA, the next point is decided by the output $y^*$ (such as loss) of objective function. The output is not generated by running classification model, instead, it is just an evaluation based on the history outputs with Gaussian distribution model. The superiority of $y^*$ is evaluated by an acquisition function which is the linear function of its mean and variance, the point corresponding to the maximum of acquisition function will be chosen as the next query point (the minimal problem needs to be converted to a maximal problem), which is formulated as:

$$x^* = \arg\max_{x \in D}(\mu(y^*_x) + \beta_t^{1/2}\sigma(y^*_x)) \tag{12}$$

the weight $\beta_t^{1/2}$ for variance is a trade of parameter, which is set to 1 in our experiment for simplicity. The next problem is how to get the mean and variance of $y^*$. In BOA, the joint distribution of $y^*$ and history outputs $y$ is assumed to subject to Gaussian distribution, which is formulated as:

$$\begin{bmatrix} y \\ y^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) & K(X,x^*) \\ K(x^*,X) & k(x^*,x^*) \end{bmatrix}\right) \tag{13}$$

$X$ is the set of evaluated points, $K(X, X)$ is the covariance matrix of these points, $K(X, x^*)$ and $K(x^*, X)$ is the covariance matrix of evaluated points and new point, and $k(x^*, x^*)$ is the covariance of new point itself. According to the Gaussian process regression model in Eq. (13), the predicted output $y^*$ can be computed as the posterior probability of $X, y, x^*$:

$$y^*|X, y, x^* \sim N(\mu(y^*), \sigma(y^*)) \tag{14}$$

where $\mu(y^*)$ and $\sigma(y^*)$ are the mean and variance in Eq. (12) respectively. They can be easily obtained by computing following expressions:

$$\begin{aligned} \mu(y^*) &= K(x^*,X)[K(X,X)]^{-1}y, \\ \sigma(y^*) &= k(x^*,x^*) - K(x^*,X)[K(X,X)]^{-1}K(X,x^*) \end{aligned} \tag{15}$$

By optimizing Eq. (12), the candidate selected by BOA is much superior to the one selected by grid search or random search. Therefore, BOA is much efficient and is well suited for functions that are very expensive to evaluate. The whole learning rate tuning procedure is summarized in Algorithm 1.

## 3.2. Phase two: augment training data with web data under the help of the fine-tuned CNN

Based on the limited training data, one important means to improve the classification accuracy is augmenting the training data with augmentation techniques. Traditional augmentation techniques focus on the affine transformation of original data, which can exert very subtle influence on the classification results when it comes to complicated images. In this section, we introduce our innovative data augmentation method, which leverages the huge volume of Internet images.

### 3.2.1. The workflow of web data augmentation

Our web data augmentation includes three steps, and each step contains an algorithm which is explained in next subsection. The steps are as followings:

1) designing a sorting method to arrange all images of the same class into a list. The order of the list item reflects the representativeness of this item to the class. The most representative item will be stored in the top, while the least will be stored in bottom. selecting the image (called the seed) from the top one by one and uploading it to Google search engine can be followed.
2) downloading the search results automatically, computing the similarity of each downloaded image to its corresponding class with our special similarity function. The image meeting the lowest requirement will be added into the candidate set. If the size of candidate set reaches the pre-set value, the loop will be terminated.
3) selecting top K images with highest similarity within each class, adding these images into the training set, and fine-tuning the network on the augmented dataset according to the similarity of each image.

### 3.2.2. Seed image selection method

In step 1, each image needs to be assigned a value representing its closeness to all images within the same class. The smaller the summed distance of one image is, the closer the image is to be the center of the class distribution. The images in distribution center will be the highest representative to its class. Here, the distance between images can be considered as the similarity between these images. Methods based on many features are proposed for images similarity measuring, such as image color histogram, pHash feature, HOG feature, SIFT feature. The mutual drawback of these methods is that they can hardly compare images on semantic level. For sports, outdoor or indoor scene images, the diversity of images within the same class is very big. Therefore, we apply a pre-trained CNN (AlexNet) to extract the semantic level feature on which the similarity is computed. Specifically, we remove the last layer that corresponding to the classifier of the pre-trained AlexNet, then the dimension of the output vector of the network is 4096, the distance between images is measured by cosine similarity. Based on this method, we compute the similarity between each pair of images within one class. The summed similarity of one image to others is considered as the degree that this image represents its corresponding class. The representation degree of image $x_j$, $degree_{rep}(x_j)$, is formulated as:

$$degree_{rep}(x_j) = \sum_{i=1\ (i \neq j)}^{n} sim(x_i, x_j) \tag{16}$$

Finally, all images are sorted in descending order according to their representative degree. The images will be selected in a top to bottom order to be uploaded to Google engine.

### 3.2.3. Search results filtering method

In step 2, the downloaded images need to be validated if it belongs to the corresponding class, since the search results contain much diversity. The method proposed to measure the membership degree of one image to its corresponding class is defined as:

1) calculating one image's probability distribution among all categories with the fine-tuned CNN built Section 3.1.3.
2) setting up rules to decide which kind of images can be added to the candidate set. To make sure the images added into training set can make contribution to the classification results, we must set up rules for filtering these images. The first rule is that the checking image's distribution probability to its corresponding class must be the highest one among all classes. This rule makes sure that the checking image belongs to the corresponding class. The second rule is that the highest distribution probability must be 1.5 times bigger than the second one. This rule makes sure that the checking image is of an obvious distance with other classes. Combining the above two rules, we get a method to compute a membership index for images meeting requirement:

$$membership_{index} = \lambda * \frac{pro_{predicted}}{pro_{random\_choice}} + (1-\lambda) * \frac{pro_{first\_highest}}{pro_{second\_highest}} \quad \lambda \in (0,1) \tag{17}$$

Where $pro_{predicted}$ denotes the predicted distribution probability to the checking image's corresponding class, $pro_{random\_choice}$ denotes the average distribution probability, $pro_{first\_highest}$ and $pro_{second\_highest}$ denotes the first highest and second highest distribution probability separately. $pro_{predicted}/pro_{random\_choice}$ denotes the distance between the checking image and the images within its corresponding class, and $pro_{first\_highest}/pro_{second\_highest}$ denotes the distance between the checking image and the images of other classes, $\lambda$ and $1-\lambda$ is the weight of two distance. Since we want to treat two distances equally, $\lambda$ is set to 0.5.

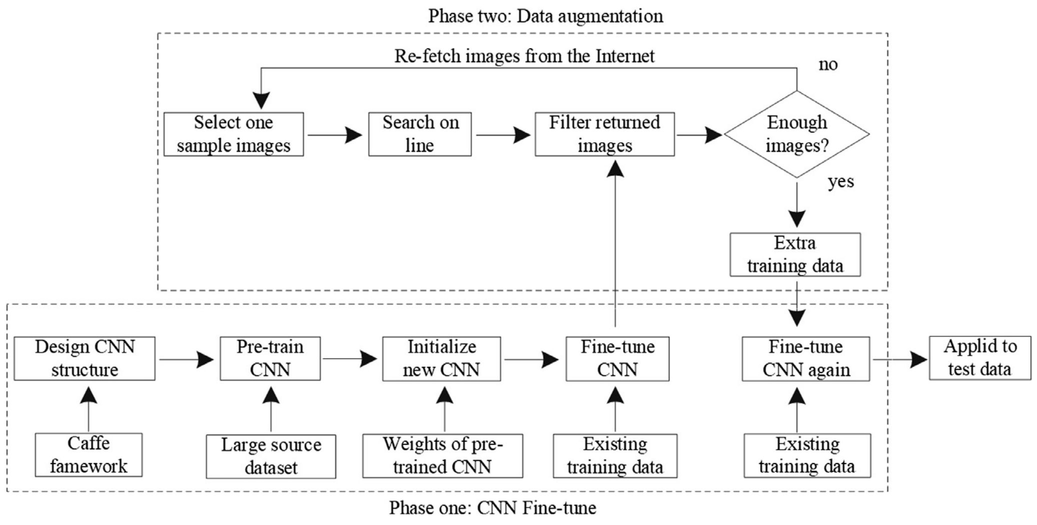

**Fig. 2.** The workflow of our web data augmentation method.

With above method, we can decide if one image can be added into the candidate set, and the position it locates in the candidate list.

In step 3, the size of candidate set can be set to several times of expected argumentation target size. In our approach, it is set to 1.2 times of target size. When the candidate set is full, we sort these images by their membership index in descending order, and choose the images from top to bottom until the amount meets the argumentation requirement.

In a word, our method to fetch argumentation images from Google is different from the work (Fergus, Fei-Fei, Perona, & Zisserman, 2010, Hauagge & Snavely, 2012), in which authors got the argumentation images by searching the class name in Google, and authors did not design any method to validate the search results. Contrasting to their approach, our content-based search method can result in high-quality images, which is the exact reason why our web augmentation method can outperform traditional augmentation methods in experimental section.

### 3.3. Overview of our two-phase classification algorithm

When going through the second phase, we obtain a bigger training set for further fine-tuning the pre-trained CNN. To get an overall idea for the workflow of our two-phase augmentation method, all steps are drawn in a diagram in Fig. 2. To show the details of our method, we summarize whole process in Algorithm 2.

## 4. Empirical evaluation

In this section, we will conduct extensive experiments to validate the effectiveness of our proposed method. The hypotheses we want to test in experiments include: (1) the fine-tuned CNN are more effective than commonly used classification models on various datasets; (2) our data augmentation technique is more effective than traditional data augmentation techniques; (3) BOA can be applied to hyper-parameters tuning in network fine-tuning.

### 4.1. Data description

The datasets to carry out experiments are listed in Table 4. As the table shown, the datasets chosen are all of small size (tens or hundreds of samples per category). For illustration, 8 images from 8 different categories of each dataset are displayed in Fig. 3. As the figure shown, these datasets contain different styles of images: the

**Table 4**
Detailed statistics about seven datasets ([a]= (Nilsback & Zisserman, 2008), [b]= (Khosla et al., 2011), [c]= (Fei-Fei, Fergus, & Perona, 2007), [d]= (Li & Fei-Fei, 2007), [e]= (Oliva & Torralba, 2001), [f]= (Quattoni & Torralba, 2009)).

| Dataset | Total images | Num. of classes | Size of classes |
|---|---|---|---|
| Flowers 102[a] | 8189 | 102 | 40–250 |
| Dogs[b] | 20,580 | 120 | 148–252 |
| Caltech-101[c] | 9146 | 101 | 40–800 |
| Event-8[d] | 1579 | 8 | 137–250 |
| 15 Scene[e] | 4485 | 15 | 210–410 |
| 67 Indoor Scene[f] | 15,620 | 67 | 101–734 |

**Table 5**
The brief information about each framework.

| Network name | Top-5 error on ImageNet | Num of layers | Num of parameters |
|---|---|---|---|
| AlexNet | 16.4% on ILSVRC-2012 | 8 | 60 M |
| VGG-16 | 7.4% on ILSVRC-2014 | 16 | 138 M |
| ResNet-152 | 3.57% on ILSVRC-2015 | 152 | 60 M |

images structure of Flowers 102 and Dogs is simpler than that of 15 Scene and 67 Indoor Scene. These datasets in different levels of classification difficulty are chosen to validate the efficiency and robustness of proposed method. More detailed statistics of each dataset is displayed in Table 2.

As the Fig. 3 shown, some overlap between the validation datasets and ImageNet can be observed. For example, there are16 categories related to dog, 5 categories are related to flower, and many categories are related to other animals which are the contents of STL-10 and Caltech-101. This overlap will impact the classification performance of pre-trained CNN models on different validation datasets, which will be explained in subsection 4.5.1.

### 4.2. CNNs description

To comprehensively evaluate the effect of features transfer learning and web data augmentation for image classification on small dataset, three well-known deep CNN models (AlexNet, VGG-16, ResNet-152) are employed to implement evaluations. Some brief information about each network is summarized in Table 5.

As the Table shown, the top-5 classification error of these models becomes lower and lower, and the architecture of these networks becomes deeper and deeper. All these networks have been implemented by the most popular deep learning frameworks, such

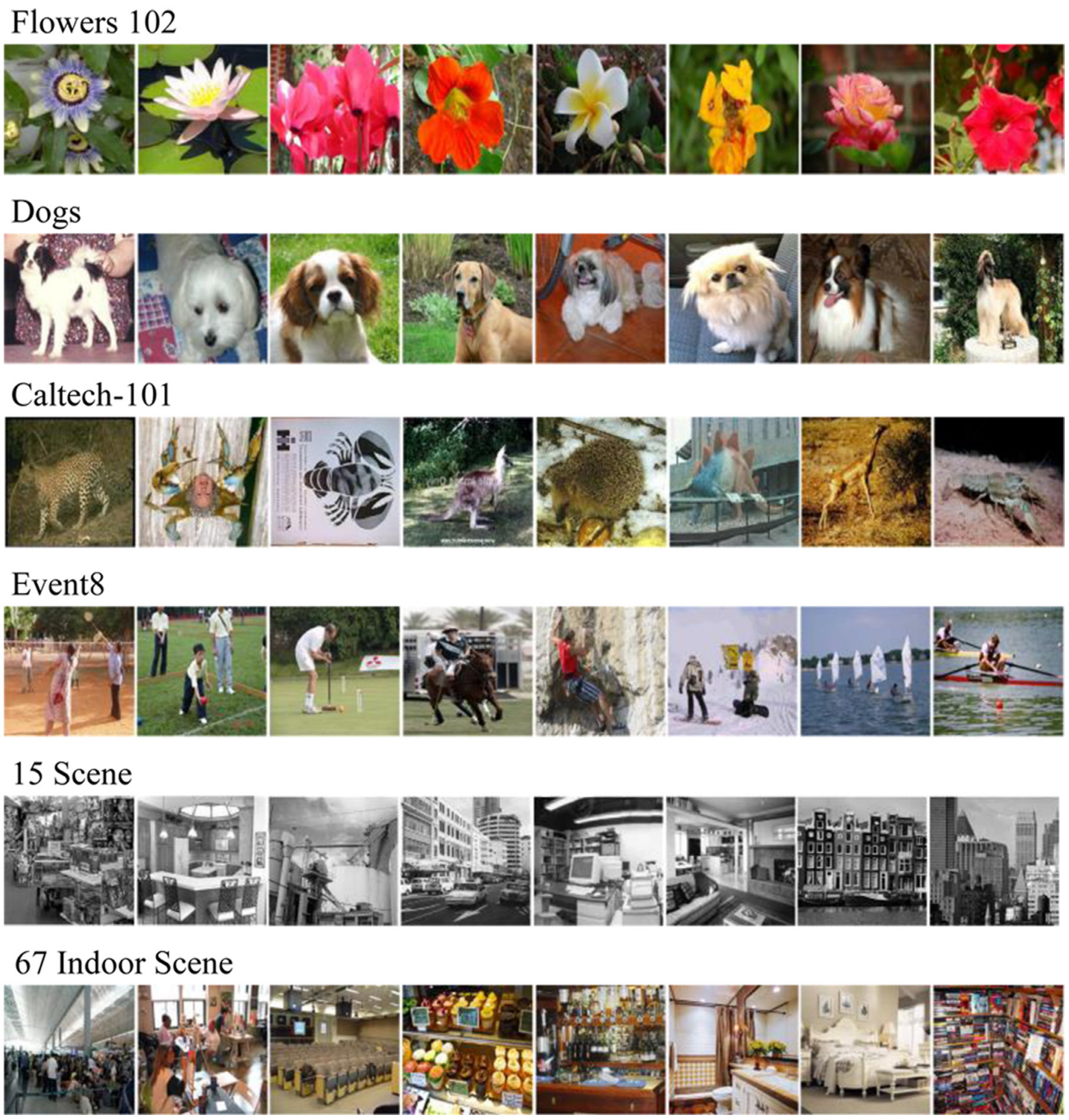

**Fig. 3.** Eight images picked up from 8 different categories of each dataset, which provide an intuitive sense of what these datasets look like.

as Caffe, Theano, Tensorflow, PyTorch, Keras. The network weights pre-trained on ImageNet can be downloaded from each framework's website. In this paper, we use Caffe to implement these networks because of its high efficiency and flexible configuration. The pre-trained weights of these networks are available from Caffe Model Zoo[1].

### *4.3. Fine-tune the pre-trained networks on existing training data*

#### *4.3.1. Fine-tuning strategies*

The architectures of these networks are very different, the fine-tuning strategy for each network also should be different too. The fine-tuning strategies for each network we adopted are listed below:

- For AlexNet, freezing the weights of the all convolutional layers, fine-tuning the weights of the last three fully-connected layers.
- For VGG-16, the fine-tune strategy is same with AlexNet.
- For GooLeNet, freezing the weights of the first 15 layers, fine-tuning the weights of the rest 7 layers.
- Fort ResNet-152, freezing the weights of the first 100 layers, fine-tuning the weights of the rest 52 layers.
- For all networks, truncating the last layer (softmax layer) of the pre-trained network and replacing it with a new softmax layer that is relevant to current problem. For example, if our task is a classification on 10 categories, the new softmax layer of the network will be of 10 categories instead of 1000 categories which is for ImageNet.

#### *4.3.2. Hyper-parameter setting with BOA*

In this section, we will first explain how to use BOA to choose a proper learning rate, then the effectiveness of this method will be validated with comparison experiment. Finally, by comparing to grid search, the advantage of BOA will be illustrated with experiments.

[1] https://github.com/BVLC/caffe/wiki/Model-Zoo.

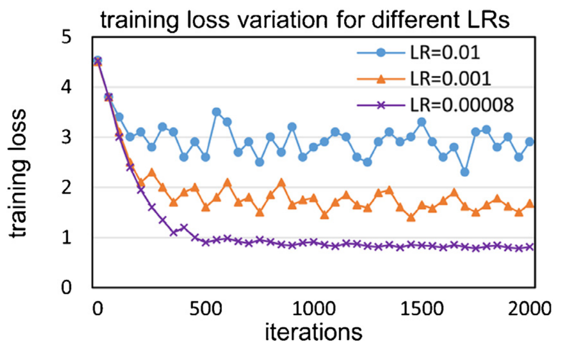

**Fig. 4.** Training loss comparing for understanding the impact of different learning rates on training loss.

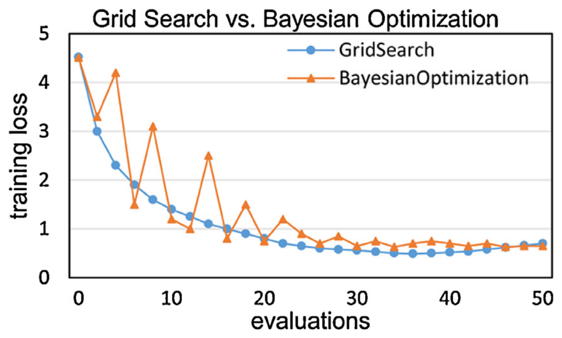

**Fig. 5.** Training loss comparison for two hyper-parameter tuning method.

**Table 6**
The proper learning rate found by BOA for each network and dataset.

| Network | Dataset | Learning rate |
|---|---|---|
| AlexNet | Flowers 102 | 0.000018 |
| | Dogs | 0.000002 |
| | Caltech-101 | 0.000014 |
| | Event8 | 0.000036 |
| | 15 Scene | 0.000068 |
| | 67 Indoor Scene | 0.000084 |
| VGG-16 | Flowers 102 | 0.000044 |
| | Dogs | 0.000024 |
| | Caltech-101 | 0.000072 |
| | Event8 | 0.000108 |
| | 15 Scene | 0.000096 |
| | 67 Indoor Scene | 0.000122 |
| ResNet-152 | Flowers 102 | 0.000004 |
| | Dogs | 0.000002 |
| | Caltech-101 | 0.000022 |
| | Event8 | 0.000042 |
| | 15 Scene | 0.000066 |
| | 67 Indoor Scene | 0.000094 |

Theoretically, BOA can be used to tune all types of hyper-parameters of deep neural network, such as learning rate, momentum, optimizer, learning rate policy. But if we tune all these hyper-parameters together, it will definitely result in high computation cost. To validate the effectiveness of BOA for hyper-parameter tuning in deep CNN fine-tuning, we try to tune the learning rate with BOA, and fix other hyper-parameters. The common setting for momentum, weight decay, optimizer and learning rate policy are: 0.9, 0.0005, Adam and fixed separately. In Caffe, the learning rate adjustment policy should be set to “fixed” if optimizer other than SDG is used. According to the theory mentioned in Section 3.1, when BOA is used to tune hyper-parameter, a set of candidate value should be defined first. Then, BOA uses exploitation and exploration method to evaluate these candidates efficiently. In literatures, the bigger learning rate and smaller learning rate seen for fine-tuning a pre-trained network are 0.001 and 0.000001 separately. We use these two values as the start point and end point of the candidate value range. To avoid huge computation cost, the whole candidate value range is only divided into 50 parts. The 50 segment points form the candidate value set: (0.0001, 0.00098, ..., 0. 000001). For finding the proper learning rate, BOA evaluates the candidate learning rate with network fine-tuning. As for the settings of other parameters, the batch size is set to 100, and the max iteration is set to 2000. After almost 30 evaluations in terms of our 50 candidates, BOA can return the proper learning rate for fine-tuning. To illustrate the superiority of the learning rate proposed by BOA, we compare it with other two common learning rates (0.01 and 0.001) for fine-tuning, taking ResNet-152 model and Caltech-101 dataset as an example. The fine-tuning strategy for ResNet-152 mentioned in last subsection is applied. With 28 evaluations BOA finds the proper learning rate 0.00008 for this dataset. The three learning rates (0.01,0.001 and 0.00008) are used to fine-tune ResNet-152 on Caltech-101. The training loss of every 50 iterations is drawn in Fig. 4.

As the figure shown, the learning rate will seriously impact the convergence of training loss. Comparing to 0.00008, the learning rate 0.01 and 0.001 seem to be too big for fine-tuning ResNet-152 on Caltech102. As the training loss shown, BOA is capable for finding a proper learning rate for deep neural network fine-tuning. The reason why the very small learning rate is more suitable for current dataset is that most images features captured by pre-trained network are shared by current dataset and ImageNet, only a little adjustment to weights is needed to make the network fit to current dataset.

In our experiments, a GTX TITAN Z graphic card is used for training, it takes 3 to 5 min to complete one evaluation for our datasets.

*4.3.2.1. Compare BOA with grid search.* We also conduct experiment to compare BOA with the simple hyper-parameters tuning method, grid search. In this experiment, the candidate learning rate set, network model and validation dataset are the same with last experiment. The training loss is considered as the metric for evaluating the performance of two methods. The mean of training loss of the last 100 iterations is considered as the training loss of the current evaluation result. The 50 evaluation results of each method are drawn in Fig. 5.

As the figure shown, comparing to grid search, BOA can find a proper learning rate with less evaluations. This feature is very valuable for deep network fine-tuning, which saves a lot of computation time. On the other hand, we find that the training loss does not decrease all the time along with the learning rate decreasing, and BOA does not necessary converge to the optimal learning rate.

With BOA we found the proper learning rates (see Table 6) for each network and each dataset. With these proper learning rates, we fine-tune each network on the original datasets.

### 4.4. Augment training data and fine-tune networks again

In this subsection, the traditional augmentation methods and our proposed method will be applied to augment the original training data separately, generating another four times artificial training data for each dataset. Finally, the artificial data will be mixed with the original data to form two types of augmented training sets for each dataset. Three pre-trained CNNs will be employed to fine-tune on these augmented datasets with the aim of comparing the effectiveness of our methods. To avoid unbalance distribution, the smallest class size of each dataset is taken as the standard size of that dataset, based on which the same amount of augmentation data is generated for each class. The number of

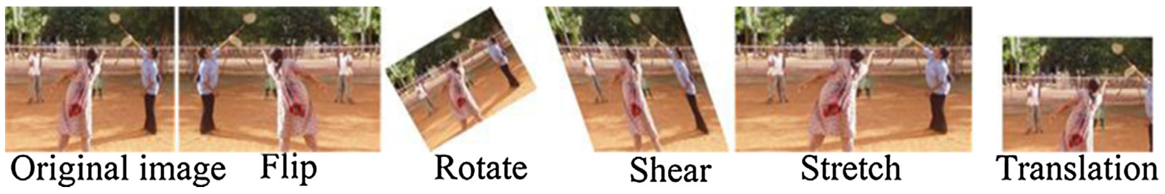

**Fig. 6.** The direct view of the different types of transformation.

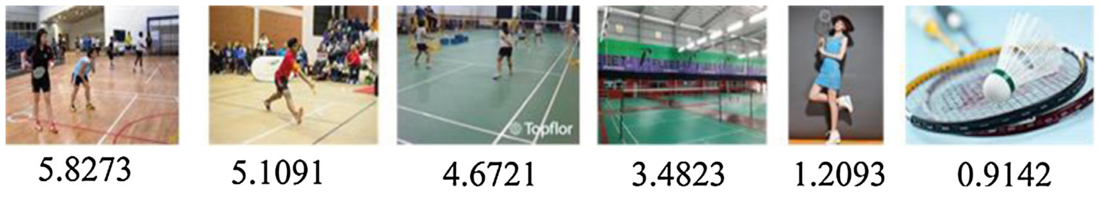

**Fig. 7.** The image on the far left is the original image from badminton class of Event8 dataset, others are gathered from the Internet with our augmentation method.

**Table 7**
The original number and augmentation target number of each class in each dataset ("4× " denotes four times).

| Dataset | Original class size | 4× |
|---|---|---|
| Flowers 102 | 40 | 160 |
| Dogs | 148 | 592 |
| Caltech-101 | 40 | 160 |
| Event8 | 137 | 528 |
| 15 Scene | 210 | 840 |
| 67 Indoor scene | 101 | 404 |

**Table 8**
Parameter setting for traditional data augmentation method.

| Method | Setting |
|---|---|
| Rotation | random with angle in [0°, 360°] |
| Translation | random with shift in [−10, 10] pixels |
| Rescaling | random with scale factor [1/1.6, 1.6] |
| Flipping | left to right |
| Shearing | random with angle [−20°, 20°] |
| Stretching | random with stretch factor [1/1.3, 1.3] |

original training data and target number of augmentation for each class in each dataset is listed in Table 7.

### 4.4.1. Traditional data argumentation

The traditional data augmentation methods include several types of transformation: rotation, translation, rescaling, flipping, shearing and stretching. The settings for these transformations are presented in Table 8. To guarantee each type of transformation exerting same effect when generating augmentation data, one type of transformation is only applied to half of original training images. After the required amount of augmentation images are gathered, three pre-trained CNNs are fine-tuned on each augmented dataset.

To get a clear idea for different types of transformation, the transformed images for the same target image with different methods are shown in Fig. 6.

### 4.4.2. Web data argumentation

To get the similar images from Google search engine, we first use our seeds selection method to select the representative images (seed image) to upload to Google search engine. Then we download images from Google engine. At last, we use the image filtering method to obtain the requirement meeting images.

To avoid causing training data unbalance problem, we choose no more than 10 samples meeting requirement for one seed image. The searching process will continue until enough images are collected. The time needed to gather all required samples depends on the availability of similar images from the Internet and the coefficient setting in the similarity function. The whole process will be time-consuming for complicated images, such as images in 67 Indoor Scene. The example image gathered from Internet of badminton class are presented in Fig. 7.

The numbers shown below the images are the values calculated with the similarity function. The numbers imply the extent that each image belongs to its corresponding class. It could be noticed that the images with high value is closely related to corresponding class. When two data augmentation processes finish, each pretrained CNNs will be fine-tuned on each augmented dataset.

## 4.5. Comparison study

In this section, we will examine the performance of three networks when they are trained from scratch on small datasets, fine-tuned on original datasets and on augmented datasets. To achieve this purpose, we apply all fine-tuned networks to the validation data of each dataset to get the classification accuracy. The accuracy of each network on each dataset for each strategy (training from scratch or fine-tuning) are listed in Table 9. To facilitate the comparison, we draw these accuracy in bar chart in Fig. 8.

### 4.5.1. Results and analysis

By comparing the classification accuracy of each network on each dataset, we can draw following conclusions:

1) In terms of small datasets, traditional shallow classification methods are much better than deep neural networks. As the accuracy of all deep CNNs trained from scratch shown, deep CNNs get very low classification accuracy, they have not been fully trained because of the large number of parameters and limited training data. To get an intuitive idea of the difference between the networks trained on small dataset and large dataset, we visualize the convolution filters of the first layer of three network in Fig. 9.
2) Pre-training combined with fine-tuning is a very effective transfer learning method for images classification. As the accuracy of all networks fine-tuned on original dataset shown, the fine-tuned networks can easily achieve the state-of-the-art accuracy. If a fine-grained fine-tuning strategy is adopted, the accuracy can be improved a little further.
3) Comparing to traditional augmentation techniques, our web data augmentation can make all CNNs achieve better performance. As the Table 9 shown, with web augmentation, all CNNs can be observed more than two percent improvement in accuracy comparing to one or half percent improvement

**Table 9**
Comparison between fine-tuned CNN and other two state-of-the-art methods on six datasets ([a]= (Chai, Lempitsky, & Zisserman, 2011), [b]= (Wang et al., 2010), [c]= (Zhou et al., 2014), [d]= (Hayat et al., 2015), [e]= (Lazebnik, Schmid, & Ponce, 2006), [f]= (Gong et al., 2014)). Here, TFS, ODS, ADSTM, ADSWDM, FT stands for trained from scratch, original dataset, augmented dataset with traditional methods, augmented dataset with web data method, pre-trained, fine-tune separately.

| Dataset | Model types | Methods | Acc (%) | Dataset | Model types | Methods | Acc (%) |
|---|---|---|---|---|---|---|---|
| Flowers 102 | Traditional models | MSDS[a] | 73.4 | Event8 | Traditional models | Hybrid-CNN[c] | 94.2 |
| | | Kanan and Cottrell, (2010) | 75.2 | | | $S^2$ICA[d] | 95.8 |
| | AlexNet | Trained on ODS | 23.8 | | AlexNet | Trained on ODS | 35.6 |
| | | FT on ODS | 87.4 | | | FT on ODS | 91.5 |
| | | FT on ADSTM | 87.9 | | | FT on ADSTM | 92.2 |
| | | FT on ADSWDM | 89.1 | | | FT on ADSWDM | 94.8 |
| | VGG-16 | Trained on ODS | 12.4 | | VGG-16 | Trained on ODS | 17.6 |
| | | FT on ODS | 80.4 | | | FT on ODS | 85.0 |
| | | FT on ADSTM | 82.9 | | | FT on ADSTM | 85.9 |
| | | FT on ADSWDM | 83.6 | | | FT on ADSWDM | 87.3 |
| | ResNet | Trained on ODS | 28.9 | | ResNet | Trained on ODS | 39.2 |
| | | FT on ODS | 88.2 | | | FT on ODS | 91.6 |
| | | FT on ADSTM | 90.1 | | | FT on ADSTM | 93.5 |
| | | FT on ADSWDM | 92.5 | | | FT on ADSWDM | 95.1 |
| Dogs | Traditional models | Simon and Rodner (2015) | 68.1 | 15 Scene | Traditional models | SPM [e] | 81.4 |
| | | Sermanet et al. (2014) | 70.8 | | | Xiao et al. (2010) | 88.1 |
| | AlexNet | Trained on ODS | 13.4 | | AlexNet | Trained on ODS | 21.8 |
| | | FT on ODS | 75.1 | | | FT on ODS | 87.0 |
| | | FT on ADSTM | 76.2 | | | FT on ADSTM | 87.8 |
| | | FT on ADSWDM | 78.5 | | | FT on ADSWDM | 89.1 |
| | VGG-16 | Trained on ODS | 8.8 | | VGG-16 | Trained on ODS | 18.4 |
| | | FT on ODS | 72.4 | | | FT on ODS | 83.2 |
| | | FT on ADSTM | 73.2 | | | FT on ADSTM | 83.9 |
| | | FT on ADSWDM | 75.6 | | | FT on ADSWDM | 85.2 |
| | ResNet | Trained on ODS | 15.2 | | ResNet | Trained on ODS | 25.3 |
| | | FT on ODS | 76.9 | | | FT on ODS | 88.3 |
| | | FT on ADSTM | 77.5 | | | FT on ADSTM | 89.2 |
| | | FT on ADSWDM | 79.8 | | | FT on ADSWDM | 90.6 |
| Caltech 101 | Traditional models | LLC[b] | 73.4 | 67 Indoor Scene | Traditional models | CNN-MOP[f] | 68.9 |
| | | Hybrid-CNN[c] | 84.8 | | | $S^2$ICA[d] | 71.2 |
| | AlexNet | Trained on ODS | 15.8 | | AlexNet | Trained on ODS | 20.3 |
| | | FT on ODS | 90.2 | | | FT on ODS | 70.3 |
| | | FT on ADSTM | 90.8 | | | FT on ADSTM | 71.4 |
| | | FT on ADSWDM | 92.3 | | | FT on ADSWDM | 75.1 |
| | VGG-16 | Trained on ODS | 10.2 | | VGG-16 | Trained on ODS | 15.9 |
| | | FT on ODS | 80.6 | | | FT on ODS | 67.7 |
| | | FT on ADSTM | 81.4 | | | FT on ADSTM | 68.5 |
| | | FT on ADSWDM | 83.2 | | | FT on ADSWDM | 70.5 |
| | ResNet | Trained on ODS | 19.5 | | ResNet | Trained on ODS | 25.9 |
| | | FT on ODS | 88.1 | | | FT on ODS | 71.6 |
| | | FT on ADSTM | 89.3 | | | FT on ADSTM | 72.1 |
| | | FT on ADSWDM | 93.8 | | | FT on ADSWDM | 74.5 |

**Algorithm 1** Learning rate tuning

```
1:  Randomly select a learning rate $x$ from a predefined range
2:  Calculate the output $y$ of objective function given $x$
3:  **Repeat**
4:  Compute the mean and variance of all candidate query points by Eq. (15)
5:    Select next query point $x^*$ by optimizing Eq. (12)
6:    Compute the output $y^*$ of the objective function
7:    Add the evaluated point and output into history set: D={D, ($x^*$, $y^*$)}
8:    iteration ++;
9:  **Until** iteration>limit or $y^*$<loss threshold
10:  **Output:** the optimal learning rate
```

for traditional augmentation techniques. For a clear understanding of the influence of web augmentation on the classification results, we draw the training and validation accuracy curves when the network is trained from scratch, fine-tuned on original dataset and web augmented dataset, taking ResNet-152 model and 67-indoor-scene dataset as an example shown in Fig. 10.

As the figure shown, the network undertakes serious over-fitting when it is trained on a small dataset, and the contribution of pre-training and web augmentation is alleviating the over-fitting and making the classifier robust.

**Algorithm 2** Two-phase classification algorithm

```
1:  **Input:** existing training data, web data, AlexNet setting
2:  Design CNN structure
3:  Pre-train CNN on ILSVRC2012 dataset
4:  Initialize new CNN with pre-trained parameters
5:  Fin-tune CNN on existing training data
6:  $n$←calculate the number of images to be retrieved from the web
7:  $t$←set the similarity threshold
8:  **For** $c$ in categories **do**
9:    $i$=0
10:   **Repeat**
11:       Select an existing sample data in $c$ with seed image selection method
      to upload
12:       $S$←download the search results
13:       **For** $s$ in $S$ **do**
14:         **If** $s$ meets requirements **then** Compute membership index of $s$ with
      Eq. (17)
15:         $S_c$←$s$
16:       **End if**
17:       **End for**
18        $i$++
19    **Until** $i$>$n$
20    Sort all items in $S_c$ in descending order
21:    Select top $K$ (augmentation target size) from $S_c$ insert into straining set
22:  **End for**
23:  Fine-tune CNN on augmented training set
24:  **Output:** new CNN classifier
```

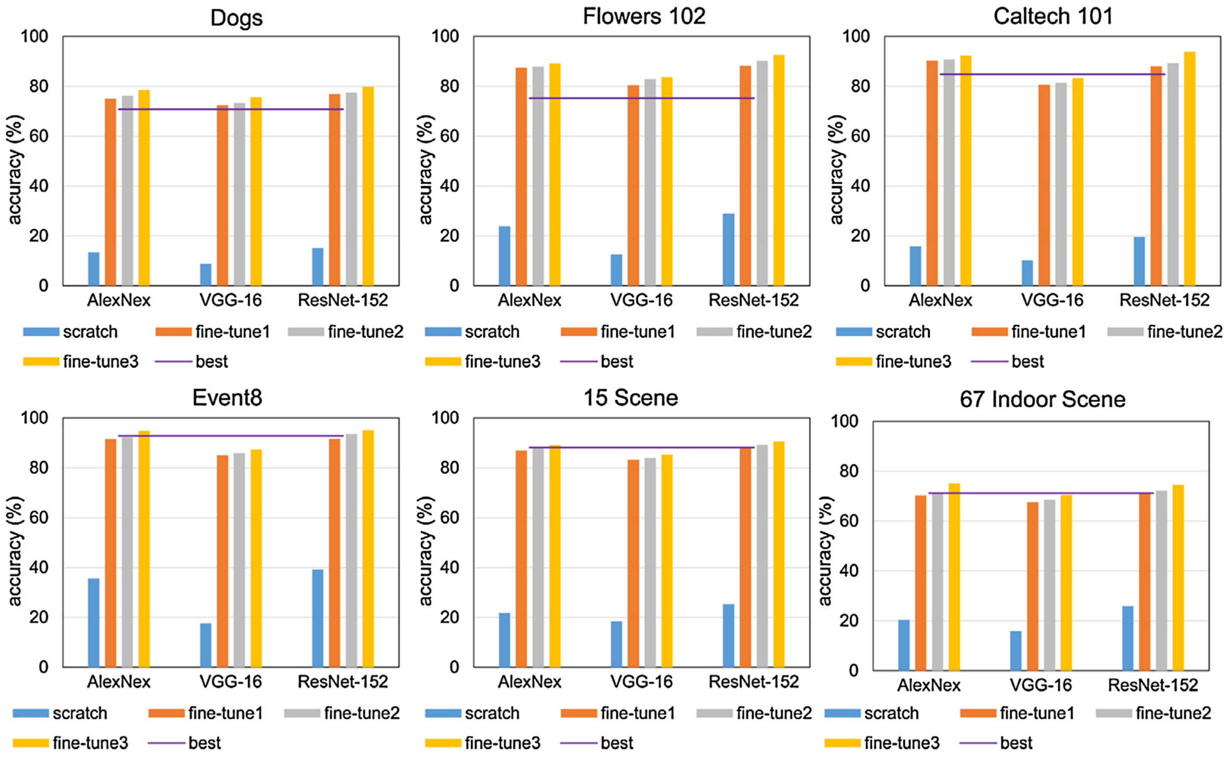

**Fig. 8.** Visualization of the classification accuracy of each network on each dataset. The horizontal line represents the state-of-the-art of accuracy.

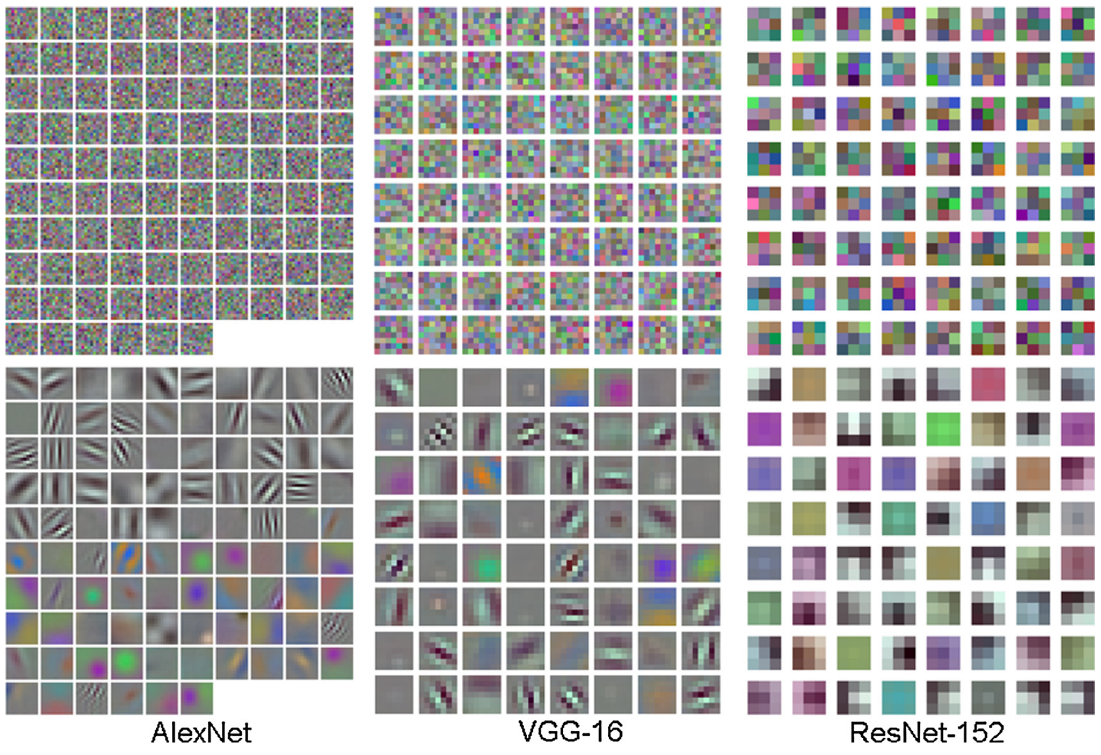

**Fig. 9.** Visualization of convolution filters of first layer of three CNNs trained on small dataset (Event8, top line) and large dataset (ImageNet, bottom line). As the figure shown, the network trained on large dataset can capture the subtle features of images, such as corners, edges, while the network trained on small dataset cannot.

4) The deeper network can achieve better performance in terms of the same number of weights. The number of weights of AlexNet and ResNest-152 are similar, the ResNet-152 can achieve higher accuracy on all datasets. In addition, more weights the network containing, more data it needs for training. VGG-16, containing 138 million weights, can achieve much lower top-5 classification error than AlexNet (7.4% vs. 16.4%) on ImageNet. However, for these small datasets, VGG-16 cannot get higher accuracy than AlexNet on any dataset.

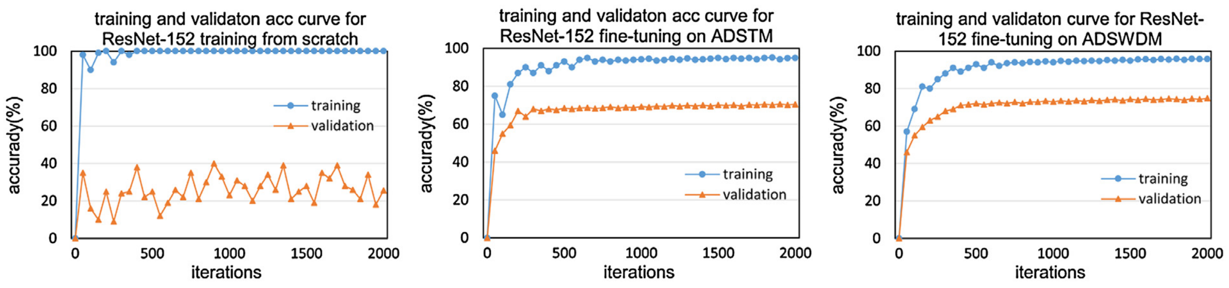

**Fig. 10.** Visualization of training and validation accuracy as the iteration goes on. The left figure is for training from scratch, the middle one is for fine-tuning on original dataset, and right one is for fine-tuning on web augmented dataset.

## 5. Conclusions and future work

In this paper, a novel two-phase classification method is proposed, which takes advantages of transfer learning and web data augmentation technique. By transfer learning, the requirement for a large number of training data is alleviated and a robust and powerful classifier is constructed; by web data augmentation, much diversity is added into the training set, which strengthens the generalization ability of the fine-tuned network and further alleviates the over-fitting. The strength of our method is to fully take advantage of existing resources (the pre-trained CNN and rich Internet images). Our method is very helpful when the training data is limited; In addition, we successfully apply Bayesian optimization algorithm to tune the hyper-parameters for network fine-tuning. The weakness of our method is that the augmentation process is time-consuming. While, with the fast-growing Internet resources, the similar candidates will be easily collected and the time needed will be shortened.

This work is an attempt to leverage the Internet images to assist the classification on local small dataset, and lots of work can be done in the future to improve the performance of deep neural network on small dataset. The main directions worth further researching in future include:

1) In Section 3.2.2, we use a pre-trained CNN without the softmax layer to extract the high-level features on which the images similarity is computed. Since the pre-trained CNN is designed for classification, the extracted feature is not very ideal for computing image similarity in the same class. The better way is to build a deep belief network (DBN) with restricted Boltzmann machines (RBM), because the stacked DBN with RBM is more good at dimensionality reduction which is the basis for image similarity computing. So far, there is no research related to using a stacked DBN with RBM on a small dataset to do dimensionality reduction has been seen. Pre-training a stacked DBN with RBM on a large dataset and reusing it on the small dataset maybe is a feasible solution, which is worth studying in the future.
2) GANs is a very effective method for generating small size and high-quality artificial data. So far, however, the biggest size of images that GANs can generate with the acceptable quality is 128*128 in the work (Salimans et al., 2016), and getting a deep GANs well trained also requires large training set. If we want to apply GANs to augment small dataset, there is lot of work to do. Pre-training plus fine-tuning may be a feasible solution worth trying in the future. Specifically, transferring the pre-trained weights of front layers to GANs, training GANs on the big size of image set and fine-tuning the pre-trained GANs on a small dataset may be a possible solution to generate bigger size and high quality artificial images for a small dataset.
3) fine-tuning strategy that is to leverage the power of pre-trained network and the domain knowledge of target dataset is very important for boosting final result. The more effective and fine-grained fine-tuning strategies are worth researching. In the next work, we will try other fine-tuning strategies, such as setting different learning rate to different layers with BOA instead of freezing the front layers and training the back layers, replace the big fully-connected layers with small fully-connected layers as that of LeNet.

## Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

## References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research, 6*(November), 1817–1853.
Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 111–118).
Chai, Y., Lempitsky, V. S., & Zisserman, A. (November 2011). BiCoS: A Bi-level co-segmentation method for image classification. In *ICCV* (pp. 2579–2586).
Dai, W., Yang, Q., Xue, G. R., & Yu, Y. (June 2007). Boosting for transfer learning. In *Proceedings of the 24th international conference on machine learning* (pp. 193–200). ACM.
Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (June 2009). Imagenet: A large-scale hierarchical image database. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on* (pp. 248–255). IEEE.
DeVries, T., & Taylor, G. W. (2017). Dataset Augmentation in Feature Space. *arXiv preprint* arXiv:1702.05538.
Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., et al. (June 2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML* (pp. 647–655).
Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12*, 2121–2159 (July).
Fei-Fei, L., Fergus, R., & Perona, P. (2007). 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.
Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2010). Learning object categories from internet image searches. *Proceedings of the IEEE, 98*(8), 1453–1466.
Flusser, J., & Suk, T. (1993). Pattern recognition by affine moment invariants. *Pattern Recognition, 26*(1), 167–174.
Glorot, X., Bordes, A., & Bengio, Y. (April 2011). Deep Sparse Rectifier Neural Networks. In *Aistats: Vol. 15* (p. 275).
Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (September 2014). Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision* (pp. 392–407). Springer International Publishing.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).
Hauagge, D. C., & Snavely, N. (June 2012). Image matching using local symmetry features. In *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on* (pp. 206–213). IEEE.
Hayat, M., Khan, S. H., Bennamoun, M., & An, S. (2015). A spatial layout and scale invariant feature representation for indoor scene classification. *arXiv preprint* arXiv:1506.05532.
Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (November 2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM international conference on multimedia* (pp. 675–678). ACM.
Kanan, C., & Cottrell, G. (June 2010). Robust classification of objects, faces, and flowers using natural image statistics. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on* (pp. 2472–2479). IEEE.

Khosla, A., Jayadevaprakash, N., Yao, B., & Li, F. F. (June 2011). Novel dataset for fine–grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine–grained visual categorization (FGVC): Vol. 2*.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06): Vol. 2* (pp. 2169–2178). IEEE.

Li, L. J., & Fei-Fei, L. (October 2007). What, where and who? Classifying events by scene and object recognition. *2007 IEEE 11th international conference on computer vision*. IEEE 1-8.

Ling, X., Xue, G. R., Dai, W., Jiang, Y., Yang, Q., & Yu, Y. (April 2008). Can chinese web pages be classified with english data source? In *Proceedings of the 17th international conference on World Wide Web* (pp. 969–978). ACM.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on: 2* (pp. 1150–1157). IEEE.

Mitchell, T. M. (1997). Artificial neural networks. *Machine learning*, 81–127.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).

Nilsback, M. E., & Zisserman, A. (December 2008). Automated flower classification over a large number of classes. In *Computer vision, graphics & image processing, 2008. ICVGIP'08. sixth Indian conference on* (pp. 722–729). IEEE.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision, 42*(3), 145–175.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (July 1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the 1st annual conference on genetic and evolutionary computation: Vol 1* (pp. 525–532). Morgan Kaufmann Publishers Inc.

Quattoni, A., & Torralba, A. (June 2009). Recognizing indoor scenes. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on* (pp. 413–420). IEEE.

Ratner, A. J., Ehrenberg, H. R., Hussain, Z., Dunnmon, J., & Ré, C. (2017). Learning to compose domain-specific transformations for data augmentation. *arXiv preprint*arXiv:1709.01643.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, 115*(3), 211–252.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., et al. (2016). Improved techniques for training gans. *Advances in Neural Information Processing Systems*, 2234–2242.

Sermanet, P., Frome, A., & Real, E. (2014). Attention for fine-grained categorization. *arXiv preprint* arXiv:1412.7054.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556.

Simon, M., & Rodner, E. (2015). Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1143–1151).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems* (pp. 640–646).

Tian, X., Tao, D., & Rui, Y. (2012). Sparse transfer learning for interactive video search reranking. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 8*(3), 26.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning, 4*(2), 26–31.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (June 2010). Locality-constrained linear coding for image classification. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on* (pp. 3360–3367). IEEE.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (June 2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on* (pp. 3485–3492). IEEE.

Xue, G. R., Dai, W., Yang, Q., & Yu, Y. (July 2008). Topic-bridged PLSA for cross-domain text classification. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 627–634). ACM.

Yang, Q., Chen, Y., Xue, G. R., Dai, W., & Yu, Y. (August 2009). Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP* (pp. 1–9). Association for Computational Linguistics. *Volume 1-Volume 1*.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 3320–3328.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. *arXiv preprint* arXiv:1506.06579.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint* arXiv:1212.5701.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 487–495.