DEFEATnet—A Deep Conventional Image Representation for Image Classification

Shenghua Gao, Lixin Duan, and Ivor W. Tsang

***Abstract*— To study underlying possibilities for the successes of conventional image representation and deep neural networks (DNNs) in image representation, we propose a deep feature extraction, encoding, and pooling network (DEFEATnet) architecture, which is a marriage between conventional image representation approaches and DNNs. In particular, in DEFEATnet, each layer consists of three components: feature extraction, feature encoding, and pooling. The primary advantage of DEFEATnet is twofold. First, it consolidates the prior knowledge (e.g., translation invariance) from extracting, encoding, and pooling handcrafted features, as in the conventional feature representation approaches. Second, it represents the object parts at different granularities by gradually increasing the local receptive fields in different layers, as in DNNs. Moreover, DEFEATnet is a generalized framework that can readily incorporate all types of local features as well as all kinds of well-designed feature encoding and pooling methods. Since prior knowledge is preserved in DEFEATnet, it is especially useful for image representation on small/medium size data sets, where DNNs usually fail due to the lack of sufficient training data. Promising experimental results clearly show that DEFEATnets outperform shallow conventional image representation approaches by a large margin when the same type of features, feature encoding and pooling are used. The extensive experiments also demonstrate the effectiveness of the deep architecture of our DEFEATnet in improving the robustness for image presentation.**

***Index Terms*— Conventional image representation, deep architecture, feature encoding, local max pooling.**

I. Introduction

IMAGE representation is the foundation of good performance for image classification. To represent each image effectively, a plethora of image representation methods [1]–[4] have been proposed in both computer vision and machine learning communities, and these methods can be categorized into conventional image representation and deep neural network (DNN)-based image representation.

Feature extraction, encoding, and pooling-based image representation is the most classical image representation pipeline in image classification [1], [2], [5]. In feature extraction module, the local features such as SIFT [6], HOG [7], and SURF [8] are extracted to represent a local image region. These local features usually show their robustness to some kind of image variances, such as rotation or translation, because of the art of handcrafted design of features with prior knowledge. In feature encoding stage, all the extracted local features are transformed with some encoding functions that removes some noises in the extracted local features, resulting in a succinct image representation. Then, a pooling operation is employed to aggregate the transformed features as one feature vector for the final image representation. A proper feature pooling strategy is crucial as it can make the image representation robust to some local translation, meanwhile preserving the spatial layout of the object. Because lots of prior knowledge and tricks have been explored for designing a robust image representation, such conventional image representation strategy manifests the best performance of image classification for quite a long while. Even nowadays, it still achieves the best performance for image classification on some small/medium size data sets.

Recently, with the advancement of machine learning technologies, DNNs have demonstrated their successes in many applications, like speech recognition [9] and hand-written digits recognition [3], [10]. In computer vision area, rather than using the manually designed features for image classification, DNN methods automatically learn features from raw pixels directly. By gradually increasing the local receptive field in a layer-by-layer manner, DNNs are able to extract low-level features (like line and corner) in low-level layers and higher level features (some object parts, like a part of a car, or even the whole object, like cat and face) in higher level layers [4], [11]. In other words, DNNs represent an object in different granularities. It is worth noting that on the very challenging ImageNet classification task (ILSVRC-2012), convolutional neural net (CNN)-based DNN outperforms conventional image representation methods (SIFT + FVs [12]) by about 8% in terms of recognition accuracy [13], and achieves the best performance on this task.

As DNNs achieve quite encouraging classification performance by largely outperforming the conventional image representation, two questions are naturally raised.

1) Is DNN really a right way for image representation?
2) Will the elaborately designed image features become obsolete?

As far as we know, for the first question, there is still no immediate answer because the fundamental theory behind the successes of DNNs has not been fully understood yet. However, for the second question, we are aware of the fact that conventional image representation still manifests the best

Manuscript received July 21, 2014; revised October 2, 2014 and November 24, 2014; accepted December 25, 2014. Date of publication January 9, 2015; date of current version March 3, 2016. This work was supported in part by the Shanghai Pujiang Program under Grant 15PJ1405700 and in part by the National Science Foundation of China under Grant 61502304. This paper was recommended by Associate Editor Y. Fu.

S. Gao is with ShanghaiTech University, Shanghai 200050, China (e-mail: gaoshh@shanghaitech.edu.cn).

L. Duan is with Amazon, Seattle, WA 98144 USA.

I. W. Tsang is with University of Technology, Sydney, NSW 2007, Australia.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2389413

performance for some computer vision tasks on some data sets, where there are not many training samples [14]. While the success of deep learning for image classification relies on huge number of training data to train a robust network with millions of parameters, the performance of DNNs-based image representation is still worse than that based on conventional image representation on some small/medium size data sets, like the Caltech 101 data set, since the prior knowledge can be used in conventional image representation which makes the image representation more robust.

The importance of our study lies in three aspects.

1) There still exist some computer vision tasks, such as medical image classification, in which it is still time consuming and cost expensive to collect data, resulting in small databases for training. Therefore, it is still demanding for the design of proper image representation for these tasks.
2) Lin *et al.* [15] and Simonyan *et al.* [16] have shown that by combining the handcrafted features-based image representation and DNN-based image representation, the recognition accuracy can be boosted. In other words, handcrafted feature-based image representation is an important complementary for DNN-based image representation, therefore, it is still meaningful to study the handcrafted feature-based image representation.
3) In our study, besides the handcrafted features, we also use the raw pixels as the input in the first layer for image representation. In this way, we can quantitatively evaluate the importance of gradually increasing the local receptive fields in image representation, which is a common practice in DNN-based image representation. Therefore, our framework helps the understanding of the components in DNNs.

By understanding the advantages of deep architecture and conventional image representation, in this paper, we propose a deep feature extraction, encoding, and pooling network (DEFEATnet), in which each layer consists of three components: feature extraction, encoding, and pooling. Using conventional image representation in each layer, the prior knowledge (e.g., translation invariance) for image representation can be preserved. To build a deep architecture, the object to be recognized is represented in different granularities. Compared with existing DNNs [3], [4], [17]–[20], our proposed DEFEATnet has two advantages.

1) Unlike those existing DNNs, DEFEATnet does not need to learn tens of thousands of parameters in the deep network.
2) By preserving the prior knowledge for image representation in each layer, DEFEATnet excels at image classification on small/medium size data sets, where the number of training images is not sufficient for the learning of those existing DNNs.

In summary, our proposed DEFEATnet has the following contributions.

1) Our DEFEATnet is a general deep architecture for image representation, which readily incorporates all types of manually designed local features as well as all kinds of feature encoding and pooling methods. More importantly, DEFEATnet bridges both the DNNs and conventional image representation and inherits the advantages of both sides for image classification.
2) Using the raw pixels as image features, we qualitatively evaluate the effect of increasing local receptive field on the improvement of performance accuracy in deep architecture.

The rest of this paper is organized as follows. We briefly review the work related to conventional image representation in Section II. In Section III, we detail modules in building the DEFEATnet architecture as well as the properties of DEFEATnet. In Section IV, we instantiate DEFEATnet with different features and feature encoding methods. We experimentally evaluate the proposed image representation framework in Section V and conclude this paper in Section VI.

II. Related Work

A. Conventional Image Representation

Bag-of-words (BoW) model [1] has been a pioneering work in feature extraction, encoding, and pooling-based conventional image representation. In the BoW model, after representing each local region as a feature vector, k-means clustering is used for generating the dictionary, and each feature is only approximated by its nearest codeword/atom in the dictionary in feature encoding module. Then, an average pooling is used, where the frequency histogram over the whole image is used to represent each image. Though the BoW representation is compact, the average pooling over the whole image causes the loss of spatial information, which is important for object and scene recognition. Moreover, the hard assignment feature quantization-based feature encoding also leads to information loss, especially for those features close to several codewords. Therefore, many more advanced feature encoding and pooling strategies have been proposed.

To preserve such spatial information, spatial pyramid matching (SPM) [2] is proposed, where each image is partitioned into increasingly finer subregions, and a histogram is calculated in each subregion and weighted. Then, all the weighted histograms are concatenated together to represent the whole image. To avoid the information loss, kernel codebook method [21] is proposed, in which each feature contributes different number to its several nearest codewords, which are determined by its similarity to the codewords. However, the codebook is still generated by k-means and one still needs to determine the number of codewords each feature contributes to. As the emergence of sparse coding technique, Yang *et al.* [5] proposed a sparse coding-based SPM (ScSPM) that uses sparse coding for dictionary learning and feature encoding. As a result, the dictionary learning and feature encoding are concurrently conducted, and each feature is sparsely represented by only a few codewords. After that, the maximum pooling is used, in which the pooling area is represented by the largest response to each codeword. Such maximum pooling [22] is experimentally proven more suitable for sparse coding-based feature encoding. Moreover,

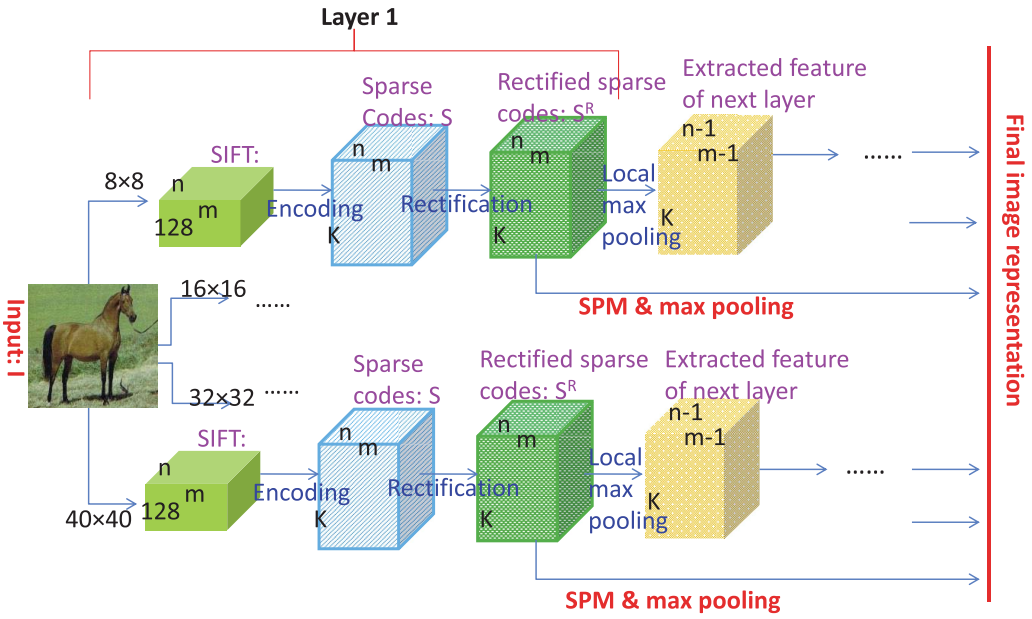


Fig. 1. Pipeline of DEFEATnet for image representation.

to encode more information among the local features, more advanced feature encoding techniques are proposed. For example, locality-constrained linear coding (LLC) [23] and Laplacian sparse coding [24] are proposed to preserve the locality information among the local features; kernel sparse representation [25] is proposed to deal with nonlinear feature reconstruction problem; local similarity global coding [26] uses a nonlinear global similarity to measure the similarity between local features and bases; Boureau *et al.* [27] propose to partition the whole feature space into different clusters with k-means first, then the local feature pooling is performed within each cluster;[1] moreover, based on SIFT feature, a middle-level feature, namely, macrofeature [28], is proposed, which actually is concatenation of spatially neighboring SIFT features. Such macrofeature covers larger object region, therefore, it is more semantically meaningful for object recognition.

B. Deep Neural Network-Based Image Representation

DNNs have shown good performance for image representation in many computer vision tasks. Restricted Boltzmann machine [19], autoencoder (AE), and convolutional neural nets (CNN) [3] are three typical building blocks used for building DNNs. Based on these building blocks, some task-specific DNN architectures are also proposed, like convolutional deep belief networks [11], stacked denoising AE [29], contractive AE [10], reconstruction-independent component analysis [20], Deconvolutional Networks [4], and so on. These techniques further improve the performance of many computer vision tasks, like hand-written digit recognition on the MNIST data set and its variants (like MNIST-rot and MNIST-bg), object recognition on the NORB, CIFAR, and ImageNet data sets. The intuitive observation from these work is that different layers of the DNNs extract different features in different scales, and these features range from the low-level features (edge and corner) to higher level features (for example, semantically meaningful object parts) [4], [11], [17]. However, the training of these DNNs usually is very time consuming, and there are many tricks for the network training. In additional to the computation/optimization issue, lots of training samples are usually the premise for the good performance of DNNs, and it is also validated that more training samples improves the recognition performance of the trained network. However, sometimes we do not have many training samples or auxiliary data sets at hand to train the network. Such scenarios restrict the performance and applicability of the DNNs in computer vision tasks.

[1]In addition to the above mentioned feature encoding methods, there are many other encoding methods. Because of the space constraint, they are not listed here.

III. Building DEFEATnet With Conventional Image Representation

A. Overview of the DEFEATnet Architecture

We show the pipeline of DEFEATnet in Fig. 1. Specifically, our DEFEATnet includes four channels, which corresponds to features extracted from patches with different sizes. For each layer in each channel, we do the feature extraction first, then feature encoding, and rectification sequentially. After that, the output of previous processing goes into two branches. For one branch, we do the max pooling over different regions generated by SPM partition. Then, the max pooling results from all the SPM pooling regions are concatenated as the output of that layer. For the other branch, we do the local max pooling, which serves as the feature extraction for the subsequent layer. Then, we concatenate the outputs of all layers in all channels as the image representation. In the following sections, we will explain each module in detail.

B. Modules in Each Layer

1) Feature Extraction in Layer One: First, by extracting the features from patches with different sizes, we can represent object parts at different scales. Features extracted from larger patches correspond to larger object parts, therefore, they are probably more discriminative than the features extracted from smaller local patches. However, since the object size is unknown beforehand, the proper patch size for extracting the feature, which can represent the object parts effectively is hard to determine. As a compromise, extracting features from patches with different sizes is a good solution.

Given an image I, after densely extracting local features from certain patch size, we get a feature map $X \in \mathbb{R}^{m\times n\times d}$, which is organized based on the spatial locations of all local features. Here, $m \times n$ corresponds to the total feature number, and d is the dimensionality of local features. Here, X can be any kind of local features, like SIFT [6], HOG [7], SURF [8], and so on.

2) Feature Encoding: Suppose that the dictionary used in feature encoding is $D \in \mathbb{R}^{d\times K}$. After feature encoding, we would get a new feature map S whose size is $m \times n \times K$. The main purposes of feature encoding are discovering the structure of the local features and removing the noises of the features to some extent. Mathematically, the feature encoding function can be written as

$$S = f(X; D). \tag{1}$$

Usually, sparse coding-related feature encoding methods are used, therefore, we simply name S as sparse codes for the ease of explanation in the following sections. Note that f can be any feature encoding function, like sparse coding [5], LLC [23], kernel sparse coding [25], soft thresholding [30], and so on.

3) Rectification: Rectification is commonly used in conventional image representation [5], [25]. In rectification module, the absolute value operation is applied on the sparse codes S. In addition to the absolute value operation, some work also try to use other rectification functions, like using the positive parts only, or decomposing S into the positive and negative parts and then applying the absolute value operation on the negative parts. As claimed in [31], these operations have similar performance. Therefore, we just use the absolute value operation for simplification. The size of the output is still the same with that of input ($m \times n \times K$) in rectification. We denote the output in this layer as S^r, then

$$S^r = \mathbf{abs}(S) \tag{2}$$

where **abs()** is the element-wise absolute value operation. We then name S^r as rectified sparse codes.

4) Max Pooling Over SPM Regions—Image Representation for the lth Layer: To represent image for the lth ($l = 1, 2, \ldots, N$. N is the depth of the network) layer, we use the commonly used SPM strategy to generate the pooling region, and use the max pooling to aggregate the sparse codes within the same SPM pooling region. One the one hand, such SPM strategy preserves the spatial layout of the object/scene in the image and allows the translation within the SPM region. On the other hand, the max pooling demonstrates its robustness to noises [22]. It is worth noting that other more advanced feature pooling methods can also be used for representing the output of each layer.

Following the commonly used SPM partition strategy [2], [5], [23], we divide each image into 1×1, 2×2, and 4×4 subregions. Consequently, each image contains 21 ($1 + 4 + 16$) pooling regions. We denote the rectified sparse codes within the pooling region R_j as $S^r_{R_j}$,[2] which is a matrix whose column number equals to the number of local features within this area, N_{R_j}. That is, $S^r_{R_j} \in \mathbb{R}^{K\times N_{R_j}}$. The output of max pooling $y_{R_j} \in \mathbb{R}^K$ is calculated as

$$y_{R_j}(i) = \max\left[S^r_{R_j}(1,i), S^r_{R_j}(2,i), \ldots, S^r_{R_j}(N_{R_j},i)\right]. \tag{3}$$

Here, $y_{R_j}(i)$ is ith entry of y_{R_j}, and it corresponds to the largest response (in terms of the absolute value) to the ith atom in codebook (D) for all the features in the pooling region R_j. Then, the output of the lth layer, which is denoted as z, is the concatenation of y_{R_j} over all the SPM pooling regions. That is

$$z = [y_{R_1}; y_{R_2}; \ldots; y_{R_{21}}]. \tag{4}$$

5) Local Max Pooling—Feature Extraction for the Subsequent Layer: Feature extraction aims to extract local features that is resilient to the variances. However, the extracted features are still not robust enough. Moreover, local features only cover a small region which is not discriminative enough to describe the image contents for image classification. To get more discriminative features to characterize more discriminative object parts, macrofeature [28] is proposed, which concatenates the four neighboring local feature together. Jia *et al.* [32] propose to apply the concept of local reception fields in neuroscience for object representation, which conducts feature pooling over a smaller region than that in SPM. Similar operation is also used in convolutional neural network that also conducts pooling over a small region to capture the higher level object structure or image contents.

Motivated by these endeavors, we propose to conduct the local max pooling as the feature extraction for the input of the subsequent layer in our DEFEATnet. The local max pooling operation is the same as the max pooling in SPM except for the pooling region is much smaller. In our implementation, we just do the local max pooling for the rectified sparse codes of 2×2 neighboring features.

On the one hand, by pooling from the smaller regions to larger regions, we can represent the object parts in different granularities, and such operation coincides with the observation that DNNs represent object parts with different granularities at different layers. On the other hand, such local max pooling does the information aggregation and makes the representation more robust to local variances, like translation. Moreover, compared with directly concatenating sparse codes within certain area, the local max pooling not only makes the representation more robust to local variances but also reduces the dimensionality of the features in the subsequent layer.

[2]Here, we reshape the 3-D sparse codes submatrix to a 2-D matrix, in which each column corresponds to the sparse codes of one feature.

Consequently, the computational cost will be significantly reduced. After the local max pooling, we perform the ℓ_2 normalization to each output.

C. Discussions

1) DEFEATnet Versus Conventional Image Representation: In conventional image representation, features extracted from different patches are equally treated, i.e., they are encoded using the same dictionary, and the rectified sparse codes of all features are pooled together to present images. Different from conventional image representation, we encode features in different channels using different dictionaries, and pool the rectified sparse codes in different channels separately. Finally, the outputs of all channels are concatenated for image representation.

2) DEFEATnet Versus DNNs: In the first layer, by reducing the distance between two neighboring features, this extraction strategy is very similar to feature extraction in the CNNs. For example, when the distance between two neighboring features is one pixel, we also extract the feature in some convolutional way, but as the feature is manually designed, therefore, it is some implicated feature extraction function. In contrast, in CNNs, the feature extraction function is learnt from the data, therefore, it is an explicit function and it is also adapted to the data. When there are not many data, the manually designed feature extraction may achieve better performance because it encodes the prior knowledge about the data. On the other hand, when the data are large, probably the learnt feature extractors are better because it is adapted from data and captures more information about the data. It is interesting that in the second layer and onward, we also extract the feature by conducting the max pooling over spatially neighboring patches, and the distance between the new features is one. In this way, it is analogous to extracting feature in a convolutional way. In this sense, by stacking the conventional image representation multiple layers, our DEFEATnet bridges the conventional image representation and DNNs.

3) Advantages of Deep Architecture: The second and third layer of the DEFEATnet work as denoising and data structure discovery at different levels. (Here, noises correspond to the real noises in the image as well as the variances of the data.) In the first layer, since the noise level is unknown, the feature encoding only partially removes some noises, and detects the data structure to some extent. In the second layer and layers onward, the local max pooling captures larger object parts in the image by gradually increasing the local receptive field. Meanwhile, feature encoding for features in higher layer further reduces noise level and discovers the structure of data corresponding to larger image regions from the relatively more clean data outputted by previous layer.

4) Performance: When only one layer is used, our DEFEATnet degenerates to the standard image representation, therefore, it can be guaranteed that the performance of proposed DEFEATnet is at least comparable with that of conventional image representation.

5) Expansibility: Our DEFEATnet is a very general architecture for image representation and it can be readily incorporated with all types of local features, and all kinds of features encoding and pooling methods. In real applications, we can adopt different local features in the feature extraction layer based on the different data properties, e.g., if texture is important for recognition, we can use the SIFT or HOG feature; if both color and texture are important, we can use color SIFT feature [33]. We can also adopt different feature encoding functions based on the computational efficiency and good performance.

IV. Instantiation of DEFEATnet

A. DEFEATnet Using SIFT and LLC

SIFT feature [6] is commonly used in conventional image representation and it has demonstrated state-of-the-art performance for many classification tasks. Therefore, we can take the SIFT feature as a showcase for DEFEATnet. Moreover, we employ LLC as our feature encoding function, because of its good performance and efficiency over the standard sparse coding-based feature encoding methods. Specifically, a variant of LLC called approximated LLC [23] is used in our network to encode a given feature x_i via

$$\min_{s_i} \|x_i - Ds_i\|_2, \quad \text{s.t. } \mathbf{1}^T s_i = 1 \tag{5}$$

where $\mathbf{1}$ is a vector whose all entries equal to 1, and D is the dictionary. In approximated LLC, D is simply learnt from the k-means with some randomly sampled features. D_{x_i} is corresponding to dictionary by setting all the columns who are not in the top k nearest neighbors (k-NN) of x_i to be zero vectors. For the sake of simplicity, we fix $k = 5$ in all layers by following the setting of LLC. s_i corresponds to the representation under current dictionary, and it is used for the image representation.

B. DEFEATnet Using SIFT and K-SVD

In addition to the LLC, we also instantiate SIFT feature-based DEFEATnet with K-SVD [34] because of its advantage in computational efficiency. The objective of K-SVD can be formulated as

$$\min_{D,S} \|X - DS\|_F, \quad \text{s.t. } \|s_i\| \le T \quad \forall i \tag{6}$$

where X is a collection of features, and K-SVD guarantees that at most T atoms are activated in the reconstruction of each feature. In our experiments, we first random sample some features (around 50K) to alternatively learn the dictionary D and sparse reconstruction coefficients matrix S. Then, we fix the dictionary. When a new feature x_i comes in, we use the following objective function to do the feature encoding:

$$\min_{s_i} \|x_i - Ds_i\|_2, \quad \text{s.t. } \|s_i\| \le T. \tag{7}$$

For simplification, we set T to be 5 in all the layers on all the data sets. Fine-tuned T may improve the recognition accuracy.

C. DEFEATnet Using Raw Pixels and K-SVD

To better understand the proposed DEFEATnet, in addition to the SIFT feature, we also propose to concatenate the intensity of the pixels within each patch as the input features for image representation. Specifically, following the common

practice in DNNs [20], we first whiten each feature by subtracting the mean of each feature. Then, we normalize each features to make its ℓ_2 norm equals to 1. As for the feature encoding, we also use the K-SVD-based feature encoding under such setting.

V. Experiments

We evaluate the performance of our proposed DEFEATnet on four standard image data sets: Corel [35], Event [36], Caltech 101, and Caltech 256. In the experiments, we compare our DEFEATnet with some existing state-of-the-art methods, such as ScSPM [5], LLC [23], and KSRSPM [25].

A. Data Set Description

The Corel-10 data set [35] is a simple data set. It contains 10 classes and each class contains 100 images. An interesting thing is that this data set contains the sport images (skiing), scene images (beach and buildings), and the object images (tigers, owls, elephants, flowers, horses, mountains, and food). To keep consistent with [35], we fix the number of the training and test images to be 50 and 50, respectively.

The Event data set[3] [36] contains 1792 images which correspond to eight different sports, including badminton, bocce, croquet, polo, rock climbing, rowing, sailing, and snowboarding. The image number in each class ranges from 137 to 250. Following the commonly used setting [5], we randomly select 70 images from each class to train the classifier and randomly select 60 images from the remaining data as the test data for each class.

The Caltech 101 data set[4] contains 101 object classes and one background class. The number of images in each class ranges from 40 to 800, and the total image number is 9144. This data set contains both classes corresponding to rigid object (like bikes and cars) and classes corresponding to nonrigid object (like animals and flowers). Therefore, the shape variance is significant. Following the commonly used protocol, we conduct the experiments by selecting 15 and 30 training images, respectively.

The Caltech 256 data set[5] contains 256 classes and 29 780 images besides the background class which has no overlap with these 256 categories. Compared with the Caltech 101 data set, this data set is more challenging in terms of the scale of the data set as well as the intra-class/inter-class variance. For example, objects are often in the center of images in Caltech 101, but the object location varies quite a lot in Caltech 256. We evaluate our method under four different settings: selecting 15, 30, 45, and 60 training images from each class and use remaining images as test data.

B. Experimental Setup

In our experiments, except for the Event data set, only gray-scale images are used, and all the images are resized to make its maximum side (width or height) to be 300 pixels without changing the aspect ratio. Following the commonly used setting [25], [37], the maximum side of images in the Event data set is resize to be 400 pixels due to the high resolution of original images. In our implementation, all the features are extracted from 8×8, 16×16, 32×32, and 40×40 patches. The step size, which is the distance between two neighboring regions for extracting features, is fixed to be four pixels, therefore, all the patches are overlapping. All the features are normalized with ℓ_2 normalization for the input of each layer. For the dictionary size in LLC and K-SVD, we fix it to be 1024, 2048, and 4096 in the layer 1, layer 2, and layer 3, respectively. (K-SVD-based DEFEATnets achieve the best performance when only two layers are used on Caltech 101, so the results reported in this paper on Caltech 101 is based on two-layer architecture for K-SVD-based DEFEATnets.) Linear SVM classifier is commonly used for sparse coding related image representation because of its efficiency and effectiveness [5], [23]. For fair comparison, we also used linear SVM classifiers with one-versus-all strategy based on the LibSVM [38] implementation. We fix the parameter $C = 10$ in the SVM formulation. On all the data sets, we conduct the experiments for 10 times by randomly generating training/test split. We use the average classification accuracy over all the classes as performance evaluation metric.

TABLE I
Performance Comparison on Corel-10 and Event Data Sets (%)

Method	Corel-10	Event
ScSPM [5]	86.2±1.01	82.74±1.46
KSRSPM[25]	90.50±1.07	86.85±0.45
LLC[23]	88.57±1.27	85.36±1.02
deep SIFT+LLC	**91.74±1.12**	**88.24±1.12**
deep SIFT+K-SVD	90.38±0.75	86.98±0.84
deep Raw Pixels+K-SVD	90.38±0.75	82.74±1.33

TABLE II
Performance Comparison on Caltech 101 Data Set (%)

ImgNo	15	30
CDBN[39]	57.7±1.5	65.4±0.5
ConvNet[40]	57.6±0.4	66.3±1.5
DeconvNet[4]	58.6±0.7	66.9±1.1
MacroFeature[28]	NA	75.7±1.1
SPM[41]	56.4±1.1	64.6±0.7
KC[21]	NA	64.14±1.18
ScSPM[5]	67.00±0.45	73.20±0.54
LLC[23]	65.43	73.44
deep SIFT+LLC	70.80±0.58	**77.49±0.97**
deep SIFT+K-SVD	**71.28±0.61**	**77.60±0.96**
deep Raw Pixel+K-SVD	69.21±0.58	75.25±1.25

C. Performance Comparison

The main purpose of this section is to evaluate the effectiveness of building deep architecture with conventional image representation, therefore, the comparison should be based on the same features, the same feature encoding method, and the same classification method. For fair comparison, we implement LLC with SIFT feature using the codes provided in [23] and follow exactly the same experimental setup with us. In addition to LLC, we also compare our work with ScSPM [5], which is based on SIFT feature and sparse

[3]http://vision.stanford.edu/lijiali/event_dataset/.
[4]www.vision.caltech.edu/Image_Datasets/Caltech101/.
[5]www.vision.caltech.edu/Image_Datasets/Caltech256/.

TABLE III
PERFORMANCE COMPARISON ON THE CALTECH 256 DATA SET (%)

	15	30	45	60
SPM	NA	34.10	NA	NA
KC	NA	27.17±0.46	NA	NA
ScSPM	27.73±0.51	34.02±0.35	37.46±0.55	40.14±0.91
KSRSPM	33.61±0.34	40.63±0.22	44.41±0.12	47.03±0.35
LLC	32.68±0.27	39.62±0.15	43.85±0.23	46.11±0.27
deep SIFT+LLC	34.58±0.40	41.57±0.27	45.62±0.29	48.21±0.24
deep SIFT+K-SVD	**35.07±0.38**	**42.06±0.25**	**45.98±0.26**	**48.52±0.32**
deep Raw Pixels+K-SVD	30.57±0.41	36.75±0.18	40.17 ±0.33	42.55±0.32

Fig. 2. Classification results of some images using different methods. The red font means the label is not correctly predicted.

coding-based feature encoding, as well as KSRSPM [25], which uses SIFT feature and kernel sparse coding-based feature encoding. We also compare our method with some existing DNNs on Caltech 101. As the performance of these DNNs on other data sets are not reported, therefore, we do not include them in this paper. We list the performance of different methods on Corel-10, Event, Caltech 101, and Caltech 256 in Tables I–III, respectively. From these results, we have the following observations.

1) The DEFEATnet based on SIFT+LLC outperforms LLC on all the data sets. Especially on Corel-10, Event, and Caltech 101, the improvement is usually more than 3%. On Caltech 256, the improvement is around 2%. The improvement of our method over LLC validates the effectiveness of DEFEATnet for image representation. We also show some images and their labels predicted by different methods on Caltech 101 in Fig. 2. We can see that the deep architecture helps the label prediction.
2) On some simple data sets (Corel-10 and Caltech 101. The background is simple, and the objects are usually at the center of the image.) the DEFEATnet based on SIFT + K-SVD achieves similar accuracy with the DEFEATnet based on raw pixels + K-SVD (different is less than 2%). On some challenging data sets with more significant variances in translation, backgrounds, and rotation (Event and Caltech 256), the DEFEATnet based on SIFT + K-SVD outperforms the DEFEATnet based on raw pixels + K-SVD by more than 5%, which validates the benefits of handcrafted features for object recognition in the DEFEATnet for the very challenging data sets.
3) For the same type of feature, K-SVD-based DEFEATnet achieves comparable or even better performance than LLC-based DEFEATnet does. However, as shown later, K-SVD-based DEFEATnet is faster than LLC. Therefore, K-SVD-based feature encoding is a good choice in DEFEATnet.
4) Similar to conventional image representation, more training samples boosts recognition accuracy for DEFEATnet-based image representation.
5) Results on Caltech 101 demonstrate that with the help of prior knowledge in extracting features, feature encoding and pooling, DEFEATnet achieves better performance for image classification on small/medium-size data sets than many existing DNNs. Even for the raw pixels feature-based DEFEATnet, the performance is still better than that of many existing DNNs. The reason may be that the performance of DNNs relies on the robustness of network for image representation, and such a robust network is with huge number of parameters which should be learnt with sufficient training samples. But on Caltech 101, the training samples are limited. Therefore, the performance of existing DNNs on Caltech 101 is not satisfactory. In contrast, our DEFEATnet preserves the prior knowledge about the data in image representation which helps the image classification. For example, SPM pooling captures the spatial layout of the object, and it also allows translation within the pooling region.
6) As KSRSPM [25], which uses nonlinear kernel sparse coding for feature encoding achieves better performance than linear feature encoding (ScSPM and LLC), we believe using nonlinear feature encoding function, the performance of DEFEATnet can be further boosted. Actually, in DNNs, nonlinear activation functions, like sigmoid, hyperbolic tangent, and sparse rectifier, are commonly used. Therefore, integrating the nonlinear feature encoding into our DEFEATnet is a promising research direction.
7) As shown in [42], when using the CNN trained on ImageNet to extract the features, the recognition accuracy is 86.5%, 74.2%, on Caltech 101 (30 training images) and Caltech 256 (60 training images). However, such baseline is an adaptive learning which requires additional data to train the network, and it is not fair to compare our method with this baseline. Moreover, with hierarchical matching pursuit [43], the recognition accuracy on these two data sets also reaches up to 82.5% and 58%. However, it uses different feature encoding strategy. The study of this paper aims at showing that deep architecture helps improve the recognition accuracy. As any feature encoding method can be easily incorporated in DEFEATnet, with more advanced feature

TABLE IV
CLASSIFICATION ACCURACY WITH DIFFERENT NUMBER OF LAYERS (%)

	deep SIFT+LLC					
Layer Number	Caltech 256				Caltech 101	
	15	30	45	60	15	30
1	33.17±0.38	40.01±0.26	43.87±0.21	46.46±0.25	69.83±0.48	76.47±0.93
2	34.22±0.38	41.26±0.30	45.20±0.18	47.75±0.26	70.50±0.66	77.18±1.04
3	**34.58±0.40**	**41.57±0.27**	**45.62±0.29**	**48.21±0.24**	**70.80±0.58**	**77.49±0.97**
	deep SIFT+K-SVD					
Layer Number	Caltech 256				Caltech 101	
	15	30	45	60	15	30
1	34.32±0.44	41.08±0.31	44.89±0.19	47.37±0.29	71.04±0.57	77.14±0.98
2	34.84±0.42	41.77±0.25	45.65±0.20	48.13±0.27	**71.28±0.61**	**77.60±0.96**
3	**35.07±0.38**	**42.06±0.25**	**45.98±0.26**	**48.52±0.32**	71.11±0.55	77.49±1.03
	deep Raw Pixels+K-SVD					
Layer Number	Caltech 256				Caltech 101	
	15	30	45	60	15	30
1	28.89±0.37	34.82±0.23	38.00±0.28	40.17±0.29	28.86±2.57	29.71±1.41
2	30.40±0.41	36.56±0.22	40.05±0.25	42.29±0.29	**69.21±0.58**	**75.25±1.25**
3	**30.57±0.41**	**36.75±0.18**	**40.17 ±0.33**	**42.55±0.32**	62.93±0.61	68.41±0.93

encoding techniques [26], [43], [44] and more advanced feature pooling techniques [27], [45], the performance of our method on different data sets can be easily boosted.

D. Performance Analysis of DEFEATnet

To further demonstrate the effectiveness of our deep architecture, we show the performance of DEFEATnet with respect to different layers in Table IV on Caltech 101 and Caltech 256.

1) We can see that two-layers DEFEATnets always outperform one-layer DEFEATnets, which proves the effectiveness of the deep architecture in DEFEATnet.
2) For different data sets and different feature encoding methods, the optimal layer which corresponds to the best performance is different. The possible reason may be that the optimal layer in DEFEATnet depends on the size of the objects in the image as well as the information loss in the feature encoding/pooling procedure. On the one hand, deeper architecture can capture objects in different scales by gradually increasing the local receptive fields. On the other hand, the local max pooling and feature encoding also causes the information loss in higher layer which makes deeper architecture may reduce the accuracy. Therefore, the optimal layer depends on the tradeoff of these two factors. Compared deep SIFT + K-SVD with deep SIFT + LLC on Caltech 101, the locality information is lost in the reconstruction in K-SVD. More information loss in lower layers makes the useful information kept in higher layer is less, which possibly makes the three-layer DEFEATnet based on K-SVD is inferior to the two-layer DEFEATnet based on K-SVD. Compared deep SIFT + K-SVD on Caltech 101 with that on Caltech 256, the layer corresponds to the best performance are different. The reason for this may be that the objects in Caltech 256 are at different locations of the image and with different sizes. In contrast, the objects are almost with the same size and at the image center in Caltech 101. Therefore, more layers can better characterize the objects in the images, and help improve the recognition accuracy on Caltech 256.
3) For the cases, where the three-layer DEFEATnets outperform the two-layer DEFEATnet, an interesting phenomenon can be observed that as we use more layers, the increase of accuracy becomes slow. Taking the SIFT+LLC-based DEFEATnet on Caltech 256 as an example, the accuracy of DEFEATnet with two layers outperforms that with one layer by 1.1%–1.3%. In contrast, the accuracy of DEFEATnet with three layers only outperforms LLC-based DEFEATnet with two layers by 0.3%–0.5%. The possible reason is that we set the number of k in k-NN to be 5 in LLC,[6] which means we only use five atoms to represent the feature in feature encoding process, which leads to information loss, especially coupled with spatial information loss in the max pooling. As the layer increases, more information is lost, which makes the performance improvement saturate. Using more atoms for image representation in LLC and preserving the spatial structure in the local max pooling in higher layer may be a way out for this phenomenon.
4) The improvement of the two-layer DEFEATnet over the one-layer DEFEATnet is more significant for raw-pixels feature than that for SIFT feature. Especially on Caltech 101, the improvement is around 40%. The possible reason for this phenomenon is that SIFT feature is robust to some local variance already, but raw-pixel-based feature is not robust to these variances. With the feature encoding and local max pooling, the robustness of the features in the second layer are greatly enhanced for raw-pixel-based input,[7] therefore, the improvement is more significant for deeper structure for raw pixels than that for SIFT features.

[6]Following the experimental setup of LLC, we empirically set $k = 5$ in all the layers. We also fix $k = 3$ or 8 in all the layers, but the performance is similar. Please note that this paper aims at proposing the DEFEATnet framework and how to set k in a more smart way is beyond the study scope of this paper.

[7]Here, robustness means the invariance of image representation or feature to the variances in scale, illumination, rotation, occlusion, and translation.

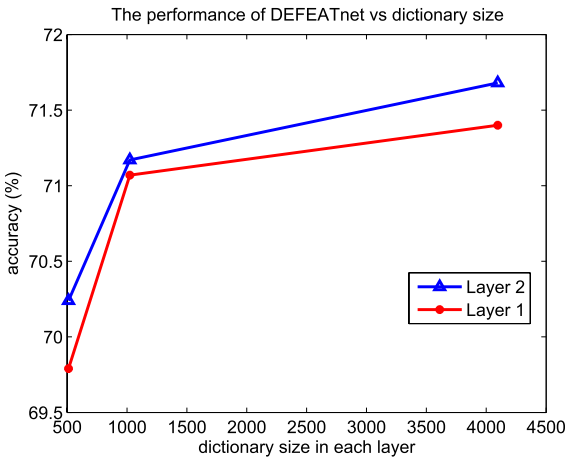


Fig. 3. Effect of dictionary size in each layer on the Caltech 101 data set (deep SIFT+K-SVD, 15 training images). As on Caltech 101, the two-layer DEFEATnet achieves the best performance under the SIFT+K-SVD setting. Here, we only show the performance corresponding to the first two layers.

E. Analysis of Different Dictionary Sizes Used in Different Layers

We increase the dictionary size of LLC with the increase of layer because of the following two reasons.

1) As the network goes deeper, the dimensionality of input feature increases because its dimensionality is the same with the dictionary size of LLC in previous layer. Following the work of commonly used criteria in feature coding [5], [23], which suggests to use an overcomplete dictionary for feature encoding, we also increase the dictionary size with the layer accordingly.
2) In DNNs, usually the number of hidden states is larger than the input features because more hidden states can disentangle the variances in features [46]. Inspired by practices in DNNs, we gradually increase the size of the dictionary in higher layer.

In addition to the dictionary size we used in the above experiments (1024–2048–4096), we also evaluate the performance of DEFEATnet with different layers by fixing the size of the dictionary in all layers to be the same. Fig. 3 shows that as we increase the dictionary size in each layer, the recognition accuracy of each layer also increases, which agrees with the observations for both the conventional image representation [47] as well as that in DNN-based image representation [29].

F. Computational Time

In our DEFEATnet, the number of parameters in each layer is the size of dictionary to be learnt, therefore, the unknown parameters in our model is much less than that in the commonly used DNNs architectures, like CNNs [13]. Moreover, the subproblems to be solved in our DEFEATnet are all linear. In contrast, the problems in DNNs are all nonlinear. Therefore, DEFEATnet can be trained more easily and efficiently than DNNs.

Specifically, when the SIFT features are used, the average time cost using unoptimized MATLAB code for LLC, K-SVD-based DEFEATnet, and LLC-based DEFEATnet

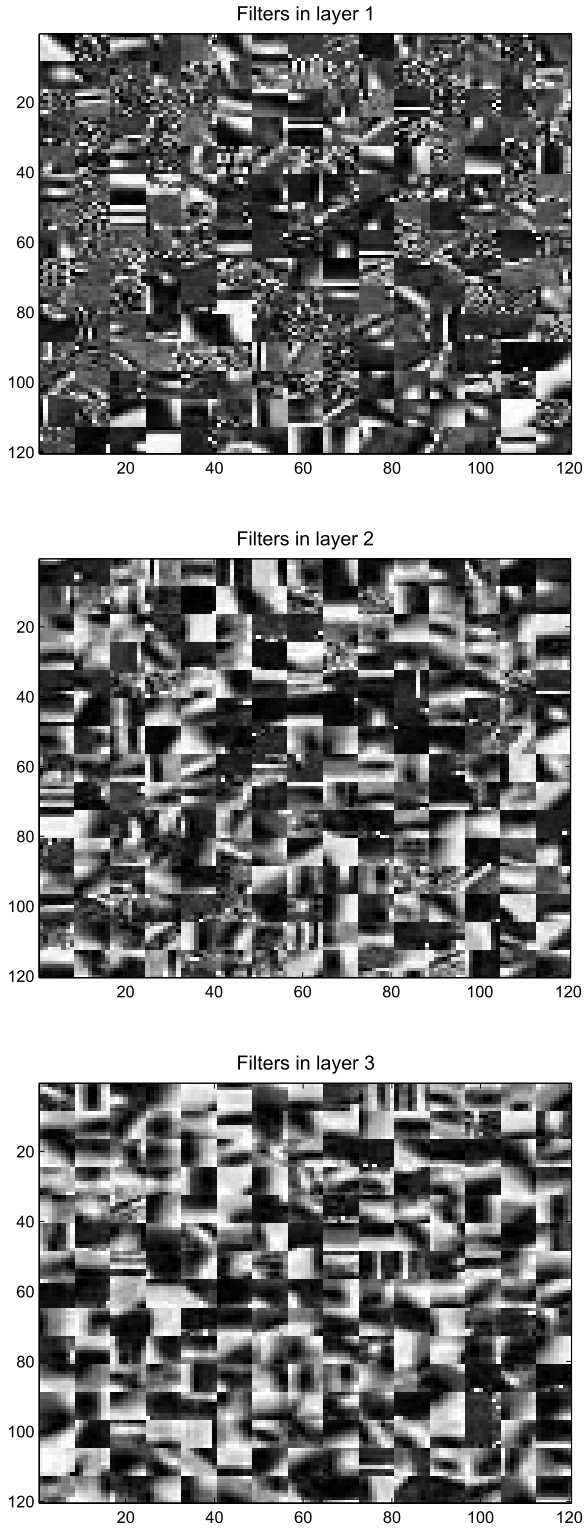


Fig. 4. Filters learnt in different layers. It is clear that the filters in the first layer contain less structured patterns than the filters learnt in the higher layer. Thus, filters in the higher layer can capture more useful structure in the image and helps image classification.

is 5.2, 8.5, and 22.5 s for representing one image on the Caltech 101 data set,[8] respectively. The reason for the slowness of LLC-based DEFEATnet is that it involves the k-NN search

[8]The average time costs for sparse coding-based feature encoding and kernel sparse coding-based feature encoding are about 8.0 and 9.2 s, respectively.

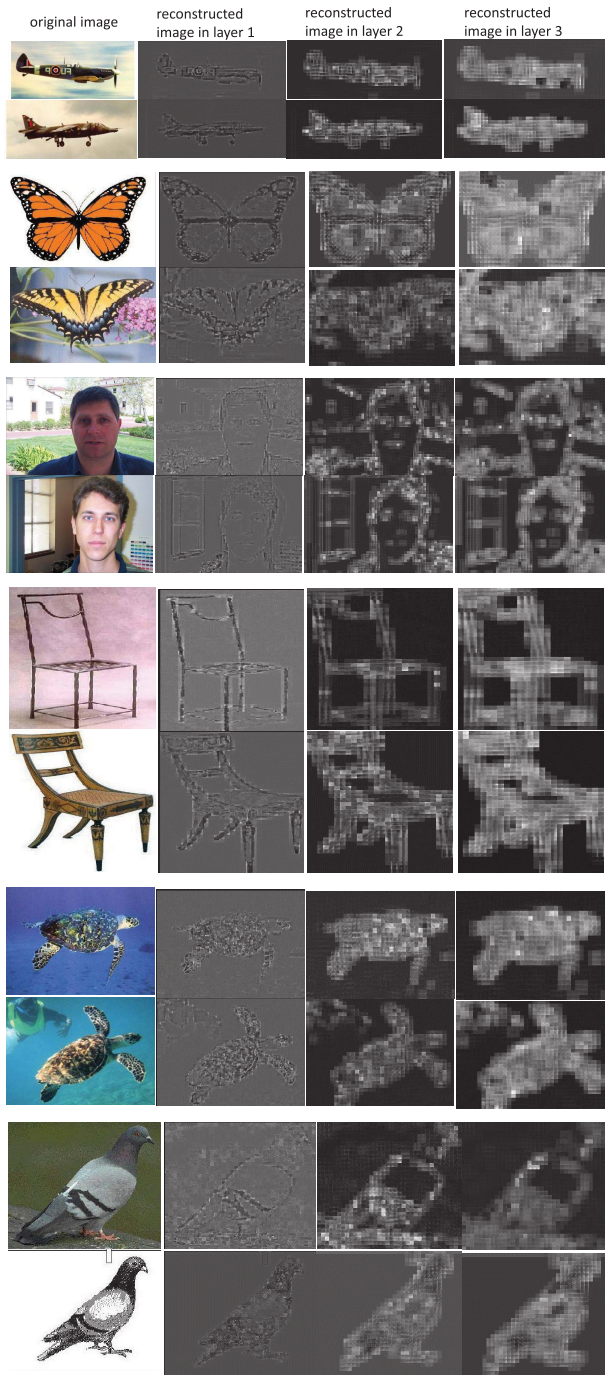


Fig. 5. Reconstructed images in different layers. We can see that the reconstructed images in the second layer are much similar for images in the same class, which visually explains the reason for higher accuracy in higher layer.

in the feature encoding step, which is usually very time consuming, especially for the features in higher layers where the dimensionality of the feature is high. It is worth noting that the optimization of DEFEATnet can be paralleled which would further boost the speed of image representation.

G. Visualization of DEFEATnet

We also visualize the learnt filters and the reconstructed images at different layers in Figs. 4 and 5, respectively, using the raw pixels and K-SVD-based feature encoding. Specifically, we set the patch size to be eight, and all patches are nonoverlapped. The dictionary size in each layer is 1024.

For the filters in the first layer, as the patch size is small, therefore, the structure is not evident. Only some filters contain structure information. In the second layer, we can see that the structure information is strong, which means each filter corresponds to edge or corner. The reasons behind this are twofolds: 1) each feature covers larger region using local max pooling over 2×2 regions (16×16 pixels) and 2) some noises are also removed in the first layer via feature encoding. Similarly, the filters in the third layer contain more complex structures because of the same reasons as that in the second layer.

We also show the reconstructed images with the sparse coefficients in different layers in Fig. 5. Compared with raw images, some noises/details are removed by feature encoding in the first layer. In the second layer, as we use the local max pooling, more detail information about the object is lost, but the structure information of the object is evident. Therefore, images within the same class are more intra-class similar than that in layer one. This also explains the reason that the two-layer DEFEATnet outperforms the one-layer DEFEATnet for image classification task. As for the reconstructed image in the third layer, as each feature corresponds to an even larger patch (32×32 pixels), and more details are lost in this layer compared with that in the second layer. Moreover, as the location information is lost in the local max pooling, the third layer actually helps improve the classification a little, or even reduces the performance. It is whether this layer helps improve the performance or not dependents on the size of the object and the patch size. If the size of the object is large while the size of patch is small, larger receptive field may still helps to enhance the robustness of image representation.

VI. Conclusion

Inspired by the advantages of conventional image representation and deep architecture used in DNNs, in this paper, we propose the novel DEFEATnet image representation by recursively stacking the handcrafted feature extraction, feature encoding, and pooling modules. Compared with DNNs, which need lots of training data to train a network with tons of parameters, our DEFEATnet can be easily trained on small or medium data sets where DNNs usually fail. Moreover, we simply instantiate the DEFEATnet with commonly used features and feature encoding methods. Experimental results on four benchmark data sets show that DEFEATnet outperforms the shallow conventional image representation methods, which validates the effectiveness of our DEFEATnet.

Currently, we fix pooling region (2×2) in local max pooling modules because we experimentally find that the performance based on 2×2 is better than that using other local pooling region (3×3 and 5×5). In future, we will explore different pooling regions to generate high-level features in multiple scales. We conjecture such a strategy can capture object parts in even finer granularities.

REFERENCES

[1] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th Int. Conf. Comput. Vis.*, Oct. 2003, pp. 1470–1477.

[2] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. 2006, pp. 2169–2178.

[3] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[4] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2528–2535.

[5] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1794–1801.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2005, pp. 886–893.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[9] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[10] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 833–840.

[11] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22*. Red Hook, NY, USA: Curran Associates, 2009, pp. 1096–1104.

[12] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1665–1672.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[14] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. (2014). "CNN features off-the-shelf: An astounding baseline for recognition." [Online]. Available: http://arxiv.org/abs/1403.6382

[15] M. Lin, Q. Chen, J. Dong, J. Huang, W. Xia, and S. Yan, "Adaptive non-parametric rectification of shallow and deep experts," Learn. Vis. Group, Nat. Univ. Singapore, Singapore, Tech. Rep., 2013.

[16] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Fisher networks for large-scale image classification," in *Advances in Neural Information Processing Systems 26*. Red Hook, NY, USA: Curran Associates, 2013, pp. 163–171.

[17] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: http://arxiv.org/abs/1312.4400

[18] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan, "DL-SFA: Deeply-learned slow feature analysis for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2625–2632.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[20] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "ICA with reconstruction cost for efficient overcomplete feature learning," in *Advances in Neural Information Processing Systems 24*. Red Hook, NY, USA: Curran Associates, 2011, pp. 1017–1025.

[21] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in *Proc. 10th Eur. Conf. Comput. Vis.*, 2008, pp. 696–709.

[22] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 111–118.

[23] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3360–3367.

[24] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, Jan. 2013.

[25] S. Gao, I. W. Tsang, and L.-T. Chia, "Sparse representation with kernels," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 423–434, Feb. 2013.

[26] A. Shaban, H. R. Rabiee, M. Farajtabar, and M. Ghazvninejad, "From local similarity to global coding: An application to image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2794–2801.

[27] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun, "Ask the locals: Multi-way local pooling for image recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2651–2658.

[28] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2559–2566.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 5, pp. 3371–3408, 2010.

[30] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 921–928.

[31] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 2146–2153.

[32] Y. Jia, C. Huang, and T. Darrell, "Beyond spatial pyramids: Receptive field learning for pooled image features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jun. 2012, pp. 3370–3377.

[33] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Dec. 2006, pp. 1978–1983.

[34] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[35] Z. Lu and H. H. S. Ip, "Image categorization with spatial mismatch kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 397–404.

[36] L.-J. Li and L. Fei-Fei, "What, where and who? Classifying events by scene and object recognition," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[37] J. Wu and J. M. Rehg, "Beyond the Euclidean distance: Creating effective visual codebooks using the histogram intersection kernel," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 630–637.

[38] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011, Art. ID 27.

[39] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 609–616.

[40] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Advances in Neural Information Processing Systems 23*. Red Hook, NY, USA: Curran Associates, 2010, pp. 1090–1098.

[41] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2007-001, 2007.

[42] M. D. Zeiler and R. Fergus. (2013). "Visualizing and understanding convolutional networks." [Online]. Available: http://arxiv.org/abs/1311.2901

[43] L. Bo, X. Ren, and D. Fox, "Multipath sparse coding using hierarchical matching pursuit," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 660–667.

[44] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja, "Low-rank sparse coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 281–288.

[45] J. Feng, B. Ni, Q. Tian, and S. Yan, "Geometric ℓ_p-norm feature pooling for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2609–2704.

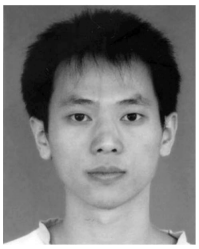
[46] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[47] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: An evaluation of recent feature encoding methods," in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 76.1–76.12.

Shenghua Gao received the B.E. (Hons.) degree from University of Science and Technology of China, Hefei, China, in 2008 and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012.

He was a Post-Doctoral Fellow with Advanced Digital Sciences Center, Singapore, from 2012 to 2014. He is currently an Assistant Professor with ShanghaiTech University, Shanghai, China. His research interests include computer vision and machine learning.

Dr. Gao received the Microsoft Research Fellowship in 2010.



Lixin Duan received the B.E. degree from University of Science and Technology of China, Hefei, China, in 2008 and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012.

He is a Machine Learning Scientist with Amazon, Seattle, WA, USA. His research interests include transfer learning, multiple instance learning, and their applications in computer vision.

Dr. Duan received the Microsoft Research Asia Fellowship in 2009 and the best student paper award at the IEEE Conference on Computer Vision and Pattern Recognition in 2010.

Ivor W. Tsang received the Ph.D. degree in computer science from Hong Kong University of Science and Technology, Hong Kong, in 2007.

He was the Deputy Director of the Centre for Computational Intelligence with Nanyang Technological University, Singapore. He is currently an Australian Future Fellow and Associate Professor with the Centre for Quantum Computation and Intelligent Systems, University of Technology, Sydney, NSW, Australia. He has authored/co-authored over 100 research papers in refereed international journals and conference proceedings, including *Journal of Machine Learning Research*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS/IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the Neural Information Processing Systems Conference, the International Conference on Machine Learning, the Conference on Uncertainty in Artificial Intelligence, the Special Interest Group on Knowledge Discovery and Data Mining Conference, the International Joint Conference on Artificial Intelligence, the Association for the Advancement of Artificial Intelligence, the Association for Computational Linguistics, the International Conference on Computer Vision, and the Conference on Computer Vision and Pattern Recognition (CVPR).

Dr. Tsang received the 2008 Natural Science Award (Class II) by the Ministry of Education, China, in 2009, which recognized his contributions to kernel methods. He received the prestigious Australian Research Council Future Fellowship for his research regarding Machine Learning on Big Data in 2013, the prestigious IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006, IEEE TRANSACTIONS ON MULTIMEDIA Prized Paper Award in 2014, and a number of best paper awards and honors from reputable international conferences, including the best student paper award at CVPR 2010, the best paper award at the International Conference on Tools with Artificial Intelligence in 2011, and the Best Poster Award Honorable Mention at the Asian Conference on Machine Learning in 2012. He also received the Microsoft Fellowship in 2005 and the European Conference on Computer Vision 2012 Outstanding Reviewer Award.