Pattern Recognition 60 (2016) 473–485

Contents lists available at ScienceDirect

Pattern Recognition

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

Unsupervised feature learning with C-SVDDNet

Dong Wang [a,b], Xiaoyang Tan [a,b,*]

[a] Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Yudao Street 29, Nanjing, 210016, China
[b] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing University of Aeronautics and Astronautics, Yudao Street 29, Nanjing, 210016, China

ARTICLE INFO

Article history:
Received 27 January 2016
Received in revised form
30 April 2016
Accepted 1 June 2016
Available online 4 June 2016

Keywords:
Unsupervised feature learning
K-means
Support Vector Data Description (SVDD)
Centering SVDD (C-SVDD)
C-SVDDNet

ABSTRACT

In this paper we present a novel unsupervised feature learning network named C-SVDDNet, a single-layer K-means-based network towards compact and robust feature representation. Our contributions are three folds: (1) we introduce C-SVDD encoding, a generalization of the K-means local encoding that adapts to the distribution information and improves the robustness against outliers; (2) we propose a method that effectively embeds the spatial information of 2D data into the final representation based on a modified SIFT descriptor; and (3) we extend our C-SVDDNet to exploit multi-scale information for better feature learning. Extensive experiments on several popular object recognition benchmarks, such as STL-10, MINST, Holiday and Copydays shows that the proposed method yields comparable or better performance than that of the previous state-of-the-art unsupervised feature learning methods.

1. Introduction

Learning good feature representation from unlabeled data is the key to making progress in recognition and classification tasks, and has attracted great attention and interest from both academia and industry recently. A representative method for this is the deep learning (DL) approach [1] with its goal to learn multiple layers of abstract representations from data. Among others, one typical DL method is the so called convolutional neural network (ConvNet), which consists of multiple trainable stages stacked on top of each other, followed by a supervised classifier [2,3]. Many variations of ConvNet network have been proposed as well for different vision tasks [4–8] with great success.

In these methods layers of representation are usually obtained by greedily training one layer at a time on the lower level [5,9,3], using an unsupervised learning algorithm. Hence the performance of single-layer learning has a big effect on the final representation. Neural network based single-layer methods, such as autoencoder [10] and RBM (Restricted Boltzmann Machine, [11]), are widely used for this but they usually have many parameters to adjust, which is very time-consuming in practice.

That motivates more simple and more efficient methods for single-layer feature learning. Among others K-means clustering algorithm is a commonly used unsupervised learning method, which maps the input data into a feature representation simply by associating each data point to its nearest cluster center. There is only one parameter involved in the K-means based method, i.e., the number of clusters, hence the model is very easy to use in practice. Coates et al. [12] shows that the K-means based feature learning network is capable to achieve superior performance compared to sparse autoencoder, sparse RBM and GMM (Gaussian Mixture Model). However, the K-means based feature representation may be too terse, and does not take the non-uniform distribution of cluster size into account - Intuitively, "bigger" clusters are likely to be part of the features with higher influential power, compared to the smaller ones.

In this paper we present a novel unsupervised feature learning network C-SVDDNet, which combines the strength of K-means encoding [12] and SVDD (Support Vector Data Description, [13–15]) towards compact and robust feature representation while being rich enough to express the knowledge needed in problem solving. Given the aforementioned weakness of traditional K-means based methods, we make three improvements: One is that we propose a more discriminative encoding method— SVDD encoding which takes the distribution information into consideration. We use SVDD encoding as the local encoding through which an image can be transformed into a set of feature maps. Our second contribution is that we propose an SIFT-based global representation which embeds spatial information of local codings into the final representation. This is different from both the Bag of Words model which simply histograms all local coding and the traditional K-means coding [12] which ignores spatial information in feature maps. Lastly we extend our C-SVDDNet to exploit multi-scale information for better feature learning. Coates et al.'s work

* Corresponding author.
E-mail addresses: dongwang@nuaa.edu.cn (D. Wang), x.tan@nuaa.edu.cn (X. Tan).

http://dx.doi.org/10.1016/j.patcog.2016.06.001
0031-3203/

[12] shows that a small receptive field is more appropriate in feature learning. But single scale receptive field can seldom characterize an object well, and information at different scales (from edges to poselets, objects) are complementary to each other to make a better feature representation.

A naive implementation of the above idea, however, shows that the centroid of an SVDD ball tends to be unreliable, because its position is mainly determined by the support vectors on the boundary and the noise in the data may deviate the center far from the mode (c.f., Fig. 4 (left)). This makes the resulting feature representation inconsistent with the data's distribution. Hence we add a new constraint to the original SVDD objective function to make the model align better with the data. We show that our modified SVDD can be solved very efficiently as a linear programming problem, instead of as a quadratic one. Usually we need to compute hundreds of clusters, and a linear programming solution can thus save us large amounts of time.

A preliminary version appeared in [16], and this work is a significant extension to the previous one with more implementation details added. We also present extensive experimental results on several popular object recognition and image retrieval benchmarks with competitive performance. The remaining parts of this paper are organized as follows: In Section 2, preliminaries are provided regarding unsupervised feature learning representation, then we detail our improved feature learning method in Section 3. In Section 4, we investigate the performance of our method empirically over several popular datasets. We conclude this paper in Section 5.

2. Unsupervised feature learning

The goal of unsupervised feature learning (UFL) is to automatically discover useful hidden patterns/features in large datasets without relying on a supervisory signal, and those learnt patterns can be utilized to create representations that facilitate subsequent supervised learning (e.g., object classification). Compared to supervised learning, unsupervised learning has its unique characteristics and advantages. Among others, one of the major advantages of UFL is that it allows to learn consistent patterns from cheap and abundant unlabelled data, without the need to manually annotate them. Such patterns distinguish from noise since by definition noise can be thought of as random variations presented in the data. This implies many potential applications of unsupervised learning, e.g., to transfer knowledge from one domain to another related domain, to regularize the behavior of a supervised algorithm, and to represent the data in a compact but effective manner. Due to these reasons, unsupervised learning is regarded as the future of deep learning [17].

There are many kinds of unsupervised learning methods in computer vision, such as Bag of Words (BoW) [18], Vector of Linearly Aggregated Descriptors (VLAD) [19], Fisher vector (FV) [20], and so on. A typical pipeline for unsupervised feature learning includes three steps. The first step is to train a set of local filters from the unlabeled training data. This is usually done by running K-means (for BoW, VLAD) or GMM (for FV) on lots of local patches sampled from the dataset and then using the centers of clusters as filter bank. The second step is to partition a given image into patches and encode them into a set of feature vectors using the learnt filter bank. These feature vectors are finally combined and normalized as the feature representation for the input image. In what follows we give a brief review on these methods.

2.1. Bag of words and its variants

The simple and basic unsupervised feature learning method is the BoW model. In this model local filters are usually the centers of clusters from K-means. These filters are looked as bins, which serves to pool the local patches nearest to them. This can be regarded as a "hard voting" method:

$$f_k(x) = \begin{cases} 1 & \text{if } k = \underset{j}{\operatorname{argmin}} \parallel c_j - x \parallel_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $f_k(x)$ is the value that a patch x was encoded as with the k-th filter c_k. In BoW, we simply count the number of patches in each bin to get a histogram representation. Thus it is a very coarse way to encode the information of an input image.

Alternatively, VLAD [19] and FV [20] encode each data point x with a vector instead of a simple count number as in BoW, which effectively improve the richness and robustness of the feature representation. Particularly, FV captures the first and second order difference between an input x and the centres of a GMM, denoted as c_k,

$$\begin{aligned} f_{uk}(x) &= \frac{1}{N\sqrt{\pi_k}} q_{ik} \Sigma_k^{-\frac{1}{2}} (x - c_k) \\ f_{vk}(x) &= \frac{1}{N\sqrt{2\pi_k}} q_{ik} [(x - c_k)\Sigma_k^{-\frac{1}{2}} (x - c_k) - 1] \end{aligned} \tag{2}$$

Then the FV coding for an local patch x is a vector of $[f_{u1}^T(x), f_{v1}^T(x), f_{u2}^T(x), f_{v2}^T(x), \ldots, f_{uK}^T(x), f_{vK}^T(x)]^T$. VLAD [19] is a simplified version of FV, with the difference signal between a patch x and a filter c_k defined as $f_k(x) = x - c_k$. As in FV, these difference signals are concatenated into a K-dim vector for feature representation. Obviously, both FV and VLAD encode much richer information than that of BoW, hence being more discriminative in subsequent tasks such as object classification.

2.2. Coates et al.'s method

To the best of our knowledge, the work of [12] is the first "deep" unsupervised learning method that is based on the K-means method, hence having close connection with the aforementioned BoW, VLAD and FV methods. Particularly, after learning a filter bank, instead of using it as basin of attraction like in BoW or as references for calculating difference vectors, it is utilized to generate a series of feature maps, one for each filter. This has at least two potential advantages: (1) compared to VLAD and FV, the encoded information is even more rich; (2) the feature maps preserve the spatial information well and hence the whole procedure could be repeated, leading to a deep unsupervised learning architecture.

Furthermore, to deal with the problem of "hard coding" in K-means, the following "triangle" encoding is proposed [12]:

$$f_k(x) = \max\{0, \mu(z) - z_k(x)\} \tag{3}$$

where $z_k(x) = \parallel x - c_k \parallel_2$, and $\mu(z)$ is the mean of the elements of z. This activation function only suppresses the response for such feature f_k with its distance to the centroid c_k above the average, hence leading to a less sparse representation (roughly half of the features could be set to be 0). Interestingly, this "triangle" encoding strategy can be also looked at as a ReLU (rectified linear unit) activation function [21] shifted by $\mu(z)$. This builds the connection with other modern deep learning literature. Last but not least, this strategy allows us to learn a distributed representation using the simple K-means method instead of using more complicated network-based methods (e.g., autoencoder and RBM).

However, this method does not take the characteristics of each cluster into consideration. Actually, the number of data point in each cluster is usually different, so is the distribution of data points

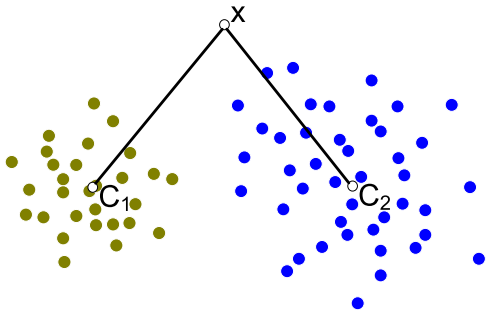

Fig. 1. Illustration of the unequal cluster effect, where the distances from a test point x to two cluster centers C_1 and C_2 are equal but the size of two clusters are different.

in each cluster. We believe that these differences would make a difference in feature representation as well. Unfortunately the aforementioned K-means feature mapping scheme completely ignores these and only uses the position of center for feature encoding. As shown in Fig. 1, although the data point x has the same distance to the centers C_1 and C_2 of two clusters, it should be assigned a different score on C_1 than on C_2 since the former cluster C_1 is much bigger than the latter. In practice such unequal clusters are not uncommon and the K-means method by itself cannot reliably grasp the size of its clusters due to the existence of outliers. To this end, we propose an SVDD based method to describe the density and distribution of each cluster and use this for more robust feature representation.

3. The proposed method

In this section, after presenting an overview of the proposed method, we give the details of our Centered-SVDD method for feature encoding, and compare it with the K-means "triangle" encoding method. Then we describe our SIFT-based post-pooling layer and discuss how to extend the method to extract multi-scale information.

3.1. Overview

A typical single-layer network contains several components: an input image is first mapped into a set of feature maps using filter banks (or dictionary), which are then subjected to a pooling/subsampling operation to condense the information contained in the feature maps. Finally, the pooled feature maps are concatenated to a feature vector, which serves as the representation for the subsequent classification/cluster tasks. There are several design options in this procedure, where the size of filter bank and that of the pooling grids are the major tradeoff one has to make.

Generally speaking, bigger filter banks help each sample find its nearby representative points more accurately but at the cost of yielding a high-dimensional representation, hence a crude pooling/subsampling is needed to reduce the dimensionality. Overall this type of architecture emphasizes more on the global aspects of the samples than on the local ones (e.g., local texture, local shape, etc.). Actually, Coates et al. show that this kind of network is able to yield state-of-the-art results on several challenging datasets [12]. On the other hand, other works use smaller filter banks but highlight the importance of detailed local information in constructing the representation, usually based on some complicated feature encoding strategy, as done in PCANet [22] or Fisher Vector [23].

In this work, we follow the second design choice, based on the consideration that the learned representation should preserve enough local spatial information for the subsequent processing. Fig. 2 gives the architecture of our single layer network. Compared to [12], we use an improved feature encoding method named C-SVDD (detailed in the next section) and adopt the architecture of relatively small dictionary. Different to [22] or [23], we learn filter banks for feature encoding but add an SIFT-based post-pooling processing procedure onto the network, which essentially projects the responses of a pooling operation into a more compact and robust representation space.

3.2. Using SVDD ball to cover unequal clusters

Traditional K-means based method only uses the centroid to represent the data of each cluster, while ignoring the distribution and density information, which is important for feature encoding. In this section we describe a method which incorporates such information into the representation model based on the SVDD (Support Vector Data Description, [13]). Formally, assume that a dataset contains N data objects, $\{x_i\}$, $i = 1, \ldots, n$. The goal of SVDD is to find a closed spherical boundary around the given data points. In this work, such a closed spherical boundary is called an SVDD ball and is described by its center a and the radius R. In order to avoid the influence of outliers, SVDD actually faces the tradeoff between two conflicting goals, i.e., minimizing the radius while covering as many data points as possible.

This can be formulated as the following objective,

$$\begin{aligned} \min_{a,R,\xi_i} \quad & R^2 + \lambda \sum_{i=1}^{N} \xi_i \\ \text{s. t.} \quad & \| x_i - a \|^2 \le R^2 + \xi_i \\ & \xi_i \ge 0, \end{aligned} \tag{4}$$

where the slack variable ξ represents the penalty related with the deviation of the i-th training data point outside the ball, and λ is a user defined parameter controlling the degree of regularization imposed on the objective. With the KKT conditions, we have $a = \sum_{i=1}^{N} x_i$, i.e., the center a of the ball is a linear combination of the data x_i. The dual function of Eq. (4) is

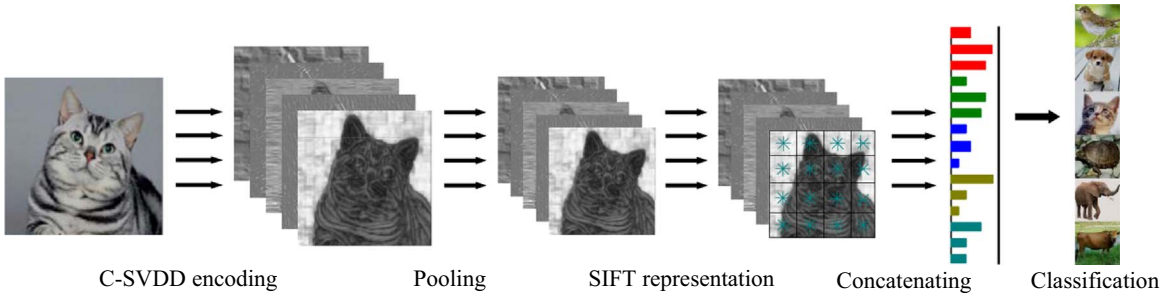

Fig. 2. The architecture of the proposed method (see text for details).

$$\begin{aligned} \max \quad & \sum_i \alpha_i \langle x_i, x_i \rangle - \sum_i \sum_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ \text{s. t.} \quad & \sum_i \alpha_i = 1, \quad \alpha_i \in [0, \lambda], \quad i = 1, \ldots, N, \end{aligned} \tag{5}$$

where α_i and α_j are Lagrangian multipliers. By solving the quadratic programming problem we can get the center a and the radius R.

The SVDD method can be understood as a type of one-class SVM and its boundary is solely determined by support vectors points. SVDD allows us to summarize a group of data points in a nice and robust way. Hence it is natural to use SVDD ball to model each cluster from K-means, thereby combining the strength of both models. In particular, for a given data point we first compute its distance h_k to the surface of each SVDD ball C_k, and then use the following modified "triangle" encoding method for feature representation (c.f., Eq. (3)),

$$f_k(x) = \max\{0, g(h) - h_k(x)\}, \tag{6}$$

where $h_k(x) = \| x - R_k \|_2$ is the distance from the point x to the surface of the k-th SVDD ball, while $g(h)$ is the average of the values h_k.

Shown in Fig. 3 for a data point x, C_i, $i = 1, 2$ respectively are the centroids of two SVDD balls with R_i, $i = 1, 2$ being the radius. Since the distances from x to C_1 and C_2 are equal, x will be assigned the same scores on the two ball with the K-means scheme (c.f., Eq. (3)). However, if we take the density and size of the clusters into accounts, the score from C_2 should be higher in our method. Hence our method potentially provides more detailed information to the subsequent processing components compared to the traditional K-means encoding method, e.g., [12].

3.3. The C-SVDD model

Although SVDD ball provides a robust way to describe the cluster of data, one unwelcome property of the ball is that it may not align well with the distribution of data points in that cluster. As illustrated in Fig. 4 (left), although the SVDD ball covers the cluster C_1 well, its center is biased to the region with low density. This should be avoided since it actually gives suboptimal estimates on the distribution of the cluster of data.

To address this issue, inspired by the observation that the centers of K-means are always located at the corresponding mode of their local density, we propose to shift the SVDD ball to the centroid of the data such that it may fit better with the distribution of the data in a cluster. Our new objective function is then formulated as,[1]

$$\begin{aligned} \min_{R,\xi_i} \quad & R^2 + \lambda \sum_{i=1}^N \xi_i \\ \text{s. t.} \quad & \| x_i - a \|^2 \le R^2 + \xi_i \\ & a = \frac{1}{N} \sum_{i=1}^N x_i \\ & \xi_i \ge 0, \end{aligned} \tag{7}$$

and its Lagrange function is as follows,

$$\begin{aligned} L(R, \xi, \alpha, \beta) = R^2 + \lambda \sum_{i=1}^N \xi_i + \sum_{i=1}^N \alpha_i \{ \| x_i - a \|^2 - R^2 - \xi_i \} \\ - \sum_{i=1}^N \beta_i \xi_i, \end{aligned} \tag{8}$$

where $\alpha_i \ge 0$ and $\beta_i \ge 0$ are the corresponding Lagrange multipliers. According to KKT Conditions, we have,

$$\frac{\partial L}{\partial R} = 2R - 2R \sum_{i=1}^N \alpha_i = 0, \quad \sum_{i=1}^N \alpha_i = 1 \tag{9}$$

$$\frac{\partial L}{\partial \xi_i} = \lambda - \alpha_i - \beta_i = 0 \tag{10}$$

Taking Eqs. (9) and (10) into the Lagrange function (8) we get that

$$L(R, \xi, \alpha, \beta) = \sum_{i=1}^N \alpha_i \{ \| x_i - a \|^2 \}.$$

Recalling that $a = \frac{1}{N} \sum_{i=1}^N x_i$, one has the following dual function,

$$\begin{aligned} \max \quad & \sum_i \alpha_i \langle x_i, x_i \rangle - \frac{2}{N} \sum_i \sum_j \alpha_i \langle x_i, x_j \rangle \\ \text{s. t.} \quad & \sum_i \alpha_i = 1, \quad \alpha_i \in [0, \lambda], \quad i = 1, \ldots, N. \end{aligned} \tag{11}$$

This can be reformulated as

$$\begin{aligned} \min \quad & \frac{2}{N} \alpha^T H e - \alpha^T F \\ \text{s. t.} \quad & \alpha^T e = 1, \quad \alpha_i \in [0, \lambda], \quad i = 1, \ldots, N, \end{aligned} \tag{12}$$

where $H = (\langle x_i, x_j \rangle)_{N \times N}$, $F = (\langle x_i, x_i \rangle)_{N \times 1}$, $e = (1, 1, \ldots, 1)^T$. This objective function is linear to α, and thus can be solved efficiently with a linear programming algorithm.

Since the model is centered towards the mode of the distribution of the data points in a cluster, we named our method as C-SVDD (centered-SVDD). Fig. 4 shows the difference between SVDD and C-SVDD, where the left is from SVDD and the right from C-SVDD. We can see that our new model aligns better with the density of the data points, as expected. It is also worth mentioning that the normalization parameter λ plays an important role in our model-a larger λ value would allow more noise to enter the ball, while $\lambda = 0$, the C-SVDD model actually reduces to the naive single-cluster K-means. More discussions on setting this value empirically will be given in Section 4.

After the model is trained, we use the modified "triangle" encoding (Eq. (6)) for feature encoding, with almost the same computational complexity with its K-means counterpart.

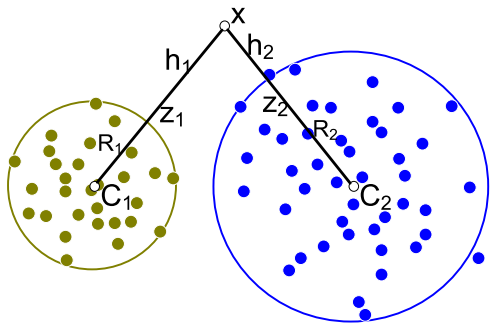

Fig. 3. Using the SVDD ball to cover the clusters of K-means, where two SVDD balls cover two clusters with different sizes, respectively. For a test point x, we encode its feature using its distance h to the surface of an SVDD ball. This can be calculated by subtracting the length R of the radius of the ball from the distance z between x to the ball center C. Hence for two SVDD balls with different sizes, the encoded features for the same point x would be different.

[1] We choose the squared L2 norm distance as a convenient for optimization. There are also other robust distance such as non-squared L2 norm distance [24].

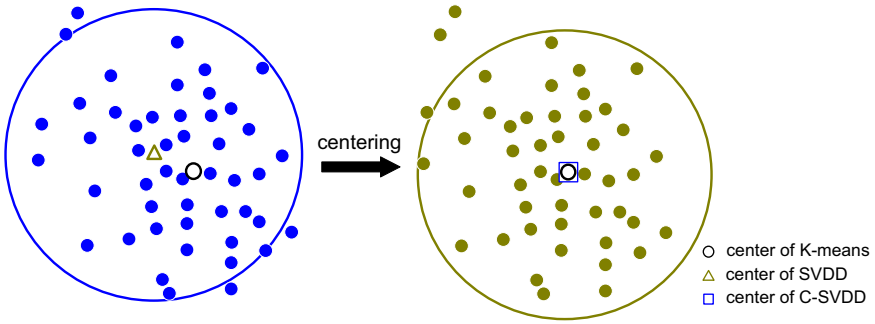

Fig. 4. Illustration of the difference between SVDD and C-SVDD. Note that after centering the SVDD ball (left), the center of C-SVDD ball (right) aligns better with the high density region of the data points.

3.4. K-means encoding versus C-SVDD encoding

In the previous sections, we present the C-SVDD encoding methods and show that the coding is more discriminative than K-means encoding in distinguishing between different inputs. In this section we will show another advantage of the C-SVDD encoding, that is, it tends to suppress the response values of "useless" feature atoms, which helps to improve the robustness of feature presentation.

It will be useful to first take a brief discussion on the difference of two kinds of feature maps, i.e., K-means-based "triangle" encoding (Eq. (3)) and our C-SVDD-based one.[2] For this a pilot experiment is conducted. Particularly, we learn a very small dictionary containing only five atoms using five face images, by clustering ZCA-whitened patches randomly sampled from the faces, and then take these for feature encoding. Fig. 5 illustrates the face images used for dictionary learning (top) and the five learnt atoms (leftmost). The feature maps of face images encoded by the K-means encoding method and those by the C-SVDD encoding method are respectively shown in Fig. 5(a) and (b), where each row is corresponding to one dictionary atom next to it and each column corresponding to one face.

By comparing the feature maps shown in Fig. 5(a) and (b), one can see that the C-SVDD-based ones contain more detailed information than the K-means feature maps for the first three atoms, while the responses of the last two atoms are largely suppressed by our method (c.f., last two rows of Fig. 5(b)). To further understand this phenomenon, we plot the entropy of each atom (by treating them as a small image patch) in Fig. 6(c). The figure shows that the entropy of the last two atoms is much smaller than that of the first three ones, which indicates that the local appearance patterns captured by these last two atoms are much simpler than those by the first three. Hence these two atoms will tend to be widely used by many faces, resulting in reduced discriminative capability in distinguishing different subjects. In this sense, it will be useful to suppress their responses (c.f., the last two rows of Fig. 5(b)).

It is also useful to inspect the distribution of local facial patches attracted by these atoms. Fig. 6(a) gives the results. It can be seen that this distribution is not uniform and the number of local patches attracted by the fourth atom is significantly larger than those by other atoms. As a result, for K-means encoding method, the feature maps yielded by this atom show much more rich details than others (see the fourth row of Fig. 5(a)), potentially indicating that it could play more important roles than others in the subsequent classification task. However, as explained above, since this atom actually contains much less information than the first three atoms (low entropy and being a "common word"), it is really not good to over-emphasize its importance in feature encoding.

This drawback of K-means feature mapping is largely bypassed by our C-SVDD-based scheme. As shown Fig. 6(b), the fourth atom actually represents a very small cluster. In fact, the radius of C-SVDD ball corresponding to the more informative atom tends to be large, and one major advantage of our C-SVDD-based strategy is that it is capable to exploit this characteristic of dictionary atoms for more effective feature encoding, as shown in the first three rows of Fig. 5(b). This partially explains the superior performance of the proposed C-SVDD method compared to its K-means counterpart (c.f., experimental results in Section 4).

3.5. Encoding feature maps with SIFT representation

In this section we describe the final component of the proposed method, i.e., constructing a global output representation for a given input image. Ideally, a good global feature should encode information rich enough for the subsequent processing and be compact enough for efficient representation. To enrich the representation, we embed the spatial relationship of local codings in feature maps into the output, rather than simply histogramming over them as in the Bag of Words (BoW) model [18]. However, the requirements of richness and compactness in feature representation are somewhat conflicting to each other, since preserving spatial information in feature representation would lead to huge dimension of the output features.

Suppose that the size of a receptive field is $r \times r$, and the size of an input image is $D \times D$. After densely extracting patches and encoding them, we would obtain K feature maps, one for each filter, with each of size $S \times S$ ($S = D - r + 1$). Particularly, for small images with $D=96$, and a small dictionary with size $K=256$ and with size of its filter $r=5$, the resulting dimension of K feature maps is nearly $2M$, which is too large for many applications. One can use such methods as average pooling or max pooling to reduce the size of feature maps. For $p \times p$ sized pooling blocks, the size of a feature map can be reduced to $\lceil \frac{S}{p} \rceil \times \lceil \frac{S}{p} \rceil$. In the above example if we set $p=5$, the dimension of each map becomes $19 \times 19 = 361$, which is still too big when concatenating K maps. However, if we choose a bigger pooling window, the more spatial information will be lost.

In this paper we proposed a variant of SIFT-representation to address the above issues. SIFT is a widely used descriptor in computer vision and is helpful to suppress the noise and improve the invariant properties of the final feature representation. However the traditional SIFT representation which extracts a 128-bit feature-descriptor could lead to a very high dimensionality of representation. For example, if we extract 128 dimensional SIFT-descriptors densely in 256 feature maps with the size of 16×16 in pixel, the dimension of the obtained representation vector will be

[2] Hereinafter we will call them respectively "K-means encoding" and "C-SVDD encoding" for short without confusion.

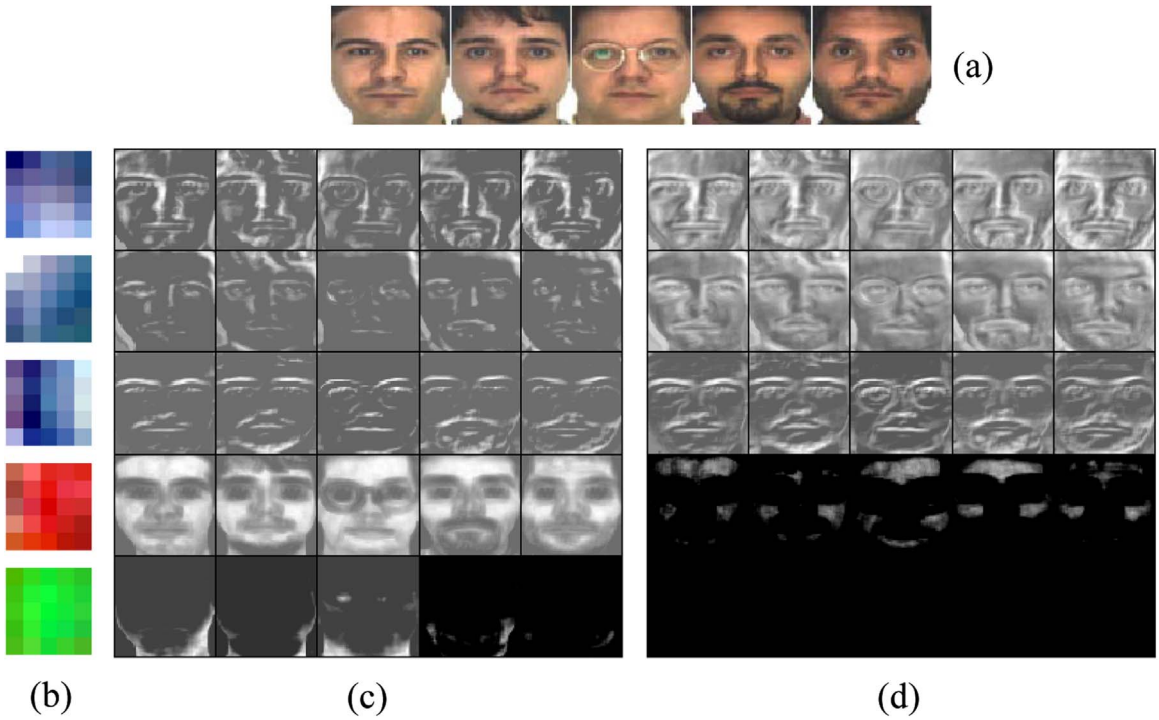

Fig. 5. Illustration of feature maps of five face images (a) using K-means (c) and C-SVDD (d) respectively, based on five local dictionary atoms (b), where maps in each row are corresponding to one atom next to it while each column corresponding to one face. For the response values in a feature map, the darker the lower.

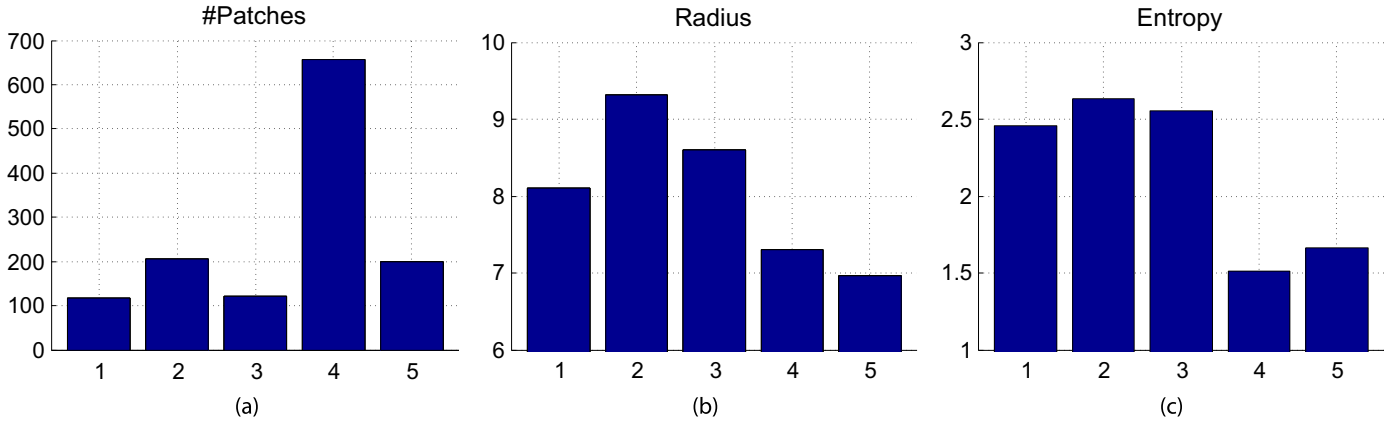

Fig. 6. Distribution of the number of patches attracted by each atom (a), the radius of the corresponding SVDD ball (b), and the entropy (c) over the five atoms shown in Fig. 5 (leftmost).

as high as over 11.8 M ($250 \times 19 \times 19 \times 128 = 11,829,248$). To address this issue, we first divide each feature map into $m \times m$ blocks and then only extract an 8-bit gradient histogram from each block in the same way as SIFT does. This results in a feature representation with dimension of $m \times m \times 8$ for each map (Such as if $m=3$, then the dim is only 72 bit). In this way we significantly reduce the dimensionality while preserving rich information for the subsequent task.

3.6. Multi-scale receptive field voting

Next we extend our method to exploit multi-scale information for better feature learning. Multi-scale method is a way to describe the objects of interest in different sizes of context. This would be useful since patches of a fixed size can seldom characterize an object well - actually they can only capture local appearance information limited in that size. For example, if the size is very small, information about edges could be captured but the information on how to combine these into more meaningful patterns such as motifs, parts, poselets, and object, is lost, while information about these entities at different levels is valuable in that they are not only discriminative by itself but complementary to each other as well. Most popular manually designed feature descriptors, such as SIFT or HoG, address this problem to some extend by pooling image gradients into edglets-like features, but it is still unclear, for example, how to assemble edglets into motifs using these methods. Convolutional neural network provides a simple and comprehensive solution to this issue by automatically learn hierarchies of features ranging from edglets to objects. However, during this procedure, information on where those high-level patterns are found becomes more and more ambiguous.

Since our C-SVDDNet is a single-layer network, it is not suitable to learn multi-scale information in a hierarchical way. Instead we obtain multi-scale information by using receptive fields of different sizes. In particular, we fetch patches with $S_i \times S_i$, $i = 1, 2, 3$ squares in size from training images and use these to train

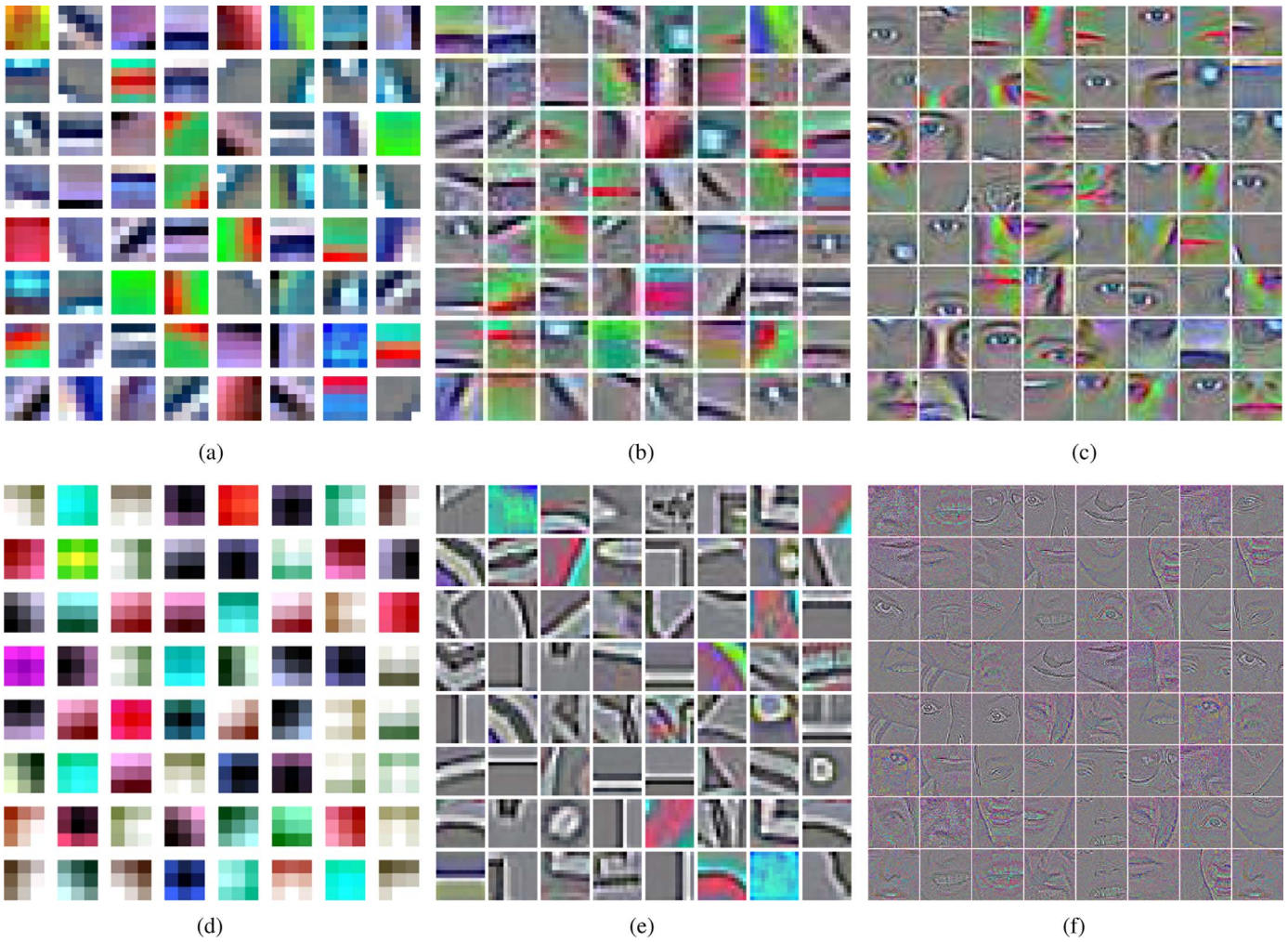

Fig. 7. Visualization of features learned from face images. (a), (b), (c) are from C-SVDDNet in 3 scales: 5×5, 10×10, 15×15 respectively. For comparison we also visualize some features from a famous convolution neural network—vggface [25]. (d), (e), (f) are from vggface in 3 different layers (low to high).

dictionary atoms with corresponding size through K-means.

Fig. 7 shows some examples of atoms we learnt on a face dataset. For comparison, we also illustrate the results from a typical Convolutional Neural Network (ConvNet)—vggface [25]. One may notice that the features learnt by the two types of networks look very similar to each other at each scale. Specifically, with the increasing window size, the learnt features become more understandable - for example, as shown in Fig. 7(c), using a receptive field with size of 20×20 on face images of 64×64, we successfully learned facial parts such as the eyes, the mouth, and so on, while a smaller receptive field gives us some oriented filters, as shown in Fig. 7(a). Capturing such information is beneficial to the subsequent processing (c.f., Fig. 11). The major differences between our network and the vggface [25] are two folds: (1) training multi-scale features in C-SVDDNet is much easier than in ConvNets, because in ConvNets features are learned layer by layer while in C-SVDDNet they can be learnt at once from the same input; (2) there is no need of labelled data for a C-SVDDNet to learn, while this is not possible for a ConvNet.

To use the learnt multi-scale information for classification, we train a separate classifier on the output layer of the corresponding network (view) according to different receptive sizes and different pooling sizes, then combine them under a boosting framework. Particularly, assume that the total number of categories is C, and we have M scales (with K different numbers of pooling sizes for each scale), then we have to learn $M \times K \times C$ output nodes. These nodes are corresponding to $M \times K$ multi-class classifiers. Let us denote the parameter of the t-th classifier $\theta_t \in R^{D \times C}$ (D is the dimension of feature representation) as $\theta_t = [w_{t1}, w_{t2}, \ldots, w_{tC}]$, where w_{tk} is the weight vector for the k-th category. We first train these parameters using a series of one-versus-rest L_2-SVM classifiers, and then normalize the outputs of each classifier using a soft max function,

$$f_{tk}(x_i) = \frac{\exp(w_{tk}^T x_i)}{\sum_{c=1}^{C} \exp(w_{tc}^T x_i)}. \tag{13}$$

Finally, the normalized predictions f_{tk} are combined to make the final decision,

$$g(x_i) = \arg\max_c \sum_t a_{tc}^T f_t(x_i). \tag{14}$$

where $f_t = \{f_{t1}, f_{t2}, \ldots, f_{tC}\}$ is the output vector of the t-th classifier, and the corresponding combination coefficients a_{tc} are trained using the following objective,

$$\min_{a_c} \sum_i \max\left(0, 1 - \sum_t a_{tc}^T f_t(x_i)\right)^2 + \lambda \parallel a_c \parallel_2 \tag{15}$$

This is the same type of one-versus-rest L_2-SVM mentioned before.

Table 1
Default parameter settings for our methods.

Parameter	Value
#Clusters	≤500
Size of receptive field	5×5,[a] 7×7, 9×9
Size of average pooling	4×4,[a] 1×1, 3×3
λ of C-SVDD	1,[a] 0.005

[a] Default setting for the non-multi-scale network.

4. Experiments and analysis

To evaluate the performance of the proposed C-SVDDNet, we conduct extensive experiments on four datasets including two object classification datasets (STL-10 [12], MINST [2]) and two image retrieval datasets (Holiday [26], INRIA Copydays [27]).

4.1. Experiment settings

All the images undergo whitening preprocessing before feeding them into the network. The whitening operation linearly transforms the data such that their covariance matrix becomes unit sphere, hence justifying the Euclidean distance we use in the K-means clustering procedure.

Unless otherwise noted, the parameter settings listed in Table 1 apply to all experiments. The influence of some important parameters, such as the number of filters, will be investigated in more detail in the subsequent sections. For single scale network, the receptive field is set to be 5×5 by default across all the datasets, as recommended in [12], while in multi-scale version, we use receptive fields in three scales, as shown in Table 1.

For C-SVDD ball, there is a regularization parameter λ to set. This parameter allows us to control the amount of noise we are willing to tolerant to. As can be seen from Eq. (1), a small λ value encourages a tight ball. We set $\lambda=1$ by default for most datasets except for those with too noisy background are set to 0.005. Furthermore, the centers in C-SVDD are set as the same as those in k-means, so that we can safely ignore the effect of the initialization of k-means.

Throughout the experiments, we use Coates' K-means "triangle" encoding method [12] (c.f., Section 2.2) as baseline (denoted as 'K-means'), while its direct counterpart method by simply replacing "triangle" encoding with C-SVDD encoding is denoted as 'C-SVDD'. Furthermore, we denote the proposed single layer network as 'C-SVDDNet', and its multi-scale version as 'MSRV + C-SVDDNet'. In addition, we re-evaluate the baseline method [12] within the proposed network by replacing its component of C-SVDD with the K-means-based encoding, denoted as 'K-meansNet'.

4.2. Analysis of the proposed method

In this section, we want to investigate various factors and components which may have critical influence on the performance the proposed method, including: (1) the number of filters: Coates's work [12] demonstrates the needs of a very big filter bank to ensure good performance, while we show that this is not necessary the case; (2) the pooling size: in our method this is viewed as a preprocess of feature maps for better global representation; (3) various way to construct the final global/output representation: we compare our SIFT based representation with other descriptors such as LBP, SURF and HOG; (4) the multi-scale information. Finally we evaluate the contribution of each stage that defines the C-SVDDNet.

We choose to conduct this series of experiments on the STL-10 dataset. The STL-10 is a large image dataset popularly used to evaluate algorithms of unsupervised feature learning or self-taught learning. Besides 100,000 unlabeled images, it contains 13,000 labeled images from 10 object classes, among which 5000 images are partitioned for training while the remaining 8000 images for testing. All the images are color images with 96×96 pixels in size. There are 10 pre-defined overlapped folds of training images, with 1000 images in each fold. In each fold, a classifier is trained on a set of 1000 training images, and tested on all 8000 testing images. In consistence with [12], we report the average accuracy across 10 folds. For unsupervised feature learning we randomly select 20,000 unlabeled data. The size of spatial pooling is 4×4, hence the size of feature maps fed for SIFT representation is 23×23. For multi-scale receptive voting we use 2 scale (5×5 and 7×7), on each of which we perform spatial pooling in 5 sizes ranging from 2×2 to 6×6.

4.2.1. Do we really need a large number of local features?

By the number of features, we mean the number of filters K used for feature extraction, which is equal to the number of dictionary atoms. One of the major conclusions of Coates et al.'s series of controlled experiments on single layer unsupervised feature learning network [12] is that compared to the choice of particular learning algorithm, the parameters that define the feature extraction pipeline, especially the number of features, have much more deep impact on the performance. Using a K-means network with 4000 features, for example, they are able to achieve surprisingly good performance on several benchmark datasets—even better than those with much deeper architectures such as Deep Boltzmann Machine [28] and Sparse Auto-encoder [12].

However, one drawback accompanying this large dictionary is that a very crude pooling size has to be adopted (e.g., 46×46 over 92×92 feature maps) to condense the resulting feature maps, otherwise the dimensionality of the final feature representation could be prohibitively high. For example, a 3×3 pooling over 4000 feature maps with 92×92 in size would lead to a total number of features over 3.8 M. Hence the first question we investigate is that whether such a large number of features are really needed all the time?

Fig. 8 gives the performance curves according to varying number of features with different methods on the STL-10 dataset. Besides the aforementioned methods, in this figure we also give

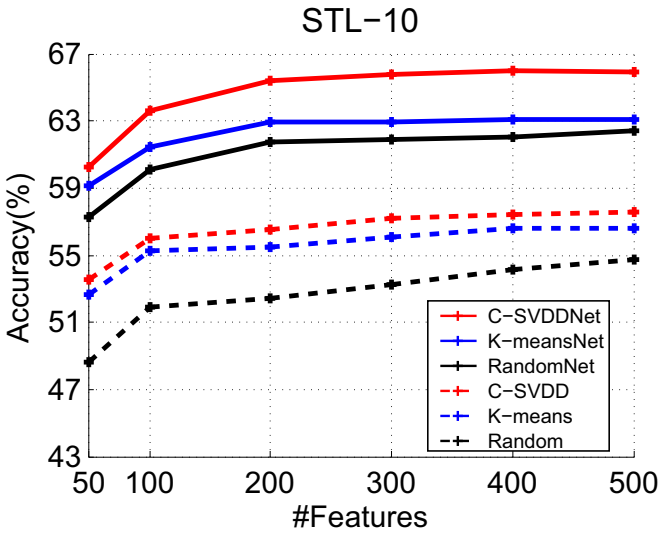

Fig. 8. The effect of different number of features on the performance with different methods on the STL-10 dataset. All parameters here (such as pooling size) are set as the same as in Fig. 9.

Table 2
Comparative performance (%) on the STL-10 dataset.

Algorithm	Accuracy (%)
Selective receptive fields [29] (2011)	60.10 ± 1.0
Trans. Invariant RBM [33] (2012)	58.70
Simulated visual fixation [34] (2012)	61.00
Discriminative sum-prod. net [35] (2012)	62.30 ± 1.0
Hierarchical matching pursuit [36] (2013)	64.50 ± 1.0
Deep feedforward networks [37] (2014)	68.00 ± 0.55
BoW(K=4800, D=4800)	51.50 ± 0.6
VLAD(K=512, D=40,960)	57.60 ± 0.6
FV (K=256, D=40,960)	59.10 ± 0.8
K-means (K=4800, D=19,200) [29] (2011)	53.80 ± 1.6
C-SVDD (K=4800, D=19,200)	54.60 ± 1.5
K-meansNet (K=500, D=36,000)	63.07 ± 0.6
C-SVDDNet (K=500, D=36,000)	65.92 ± 0.6
MSRV+K-meansNet	64.96 ± 0.4
MSRV+C-SVDDNet	**68.23** ± 0.5

the results of random dictionary (i.e, local dictionary atoms are obtained randomly without being fine tuned by k-means, denoted as "Random") and of the combination of random dictionary and SIFT representation (denoted as 'RandomNet').

It can be seen that with the increasing number of features, the performance of both K-means and C-SVDD methods rises, which is consistent with the results by Coates et al. [12]. One possible explanation is that since both K-means encoding and C-SVDD encoding use the learnt dictionary to extract non-linear features, more dictionary atoms help to disentangle factors of variations in images. In our opinion the capability to learn a large number of atoms at relatively low computational cost is one of the major advantages of K-means based methods for unsupervised feature learning over other algorithms such as Gaussian Mixture Model (GMM), sparse coding, and RBM. For example, it is difficult for a GMM to learn a dictionary with over 800 atoms [12].

On the other hand, a too large dictionary can increase the redundancy and decrease the efficiency. Hence it is desirable to reduce the number of features while not hurting the performance too much. Fig. 8 shows that our C-SVDD encoding method consistently works better than the K-means encoding at different number of features, and combining C-SVDD encoding and SIFT-based representation dramatically reduces the needs for large dictionary without scarifying the performance. Actually, Table 2 and Fig. 8 show that using our C-SVDD encoding and the SIFT feature representation, the dictionary size reduces by 10 times (from 4800 [29] to 500) while the performance improves by 12% (from 53.80% [29] to 65.92%).

As for the random dictionary (denoted as 'Random' and 'RandomNet' in Fig. 8), it is interesting to see that when the number of atoms is small, random atoms perform much worse than those finetuned by k-means. But as the size of dictionary increases, the performance difference between the random dictionary and K-means dictionary begins to reduce. For example, at 500 features, using random atoms gives a performance of 54.77%, slightly worse than that of k-means (56.63%), and the performance of RandomNet (62.45%) is also close to that of K-meansNet (63.07%). However the performance of both random methods is all much lower than that of the C-SVDD based methods.

4.2.2. *Effect of the pooling size*

To investigate the effect of different pooling sizes on the performance using the proposed method, we conduct a series of experiments on the STL-10 dataset. Particularly, for a 96×96 original image, we use a receptive field of 5×5 in pixel for feature extraction and obtain a layer of feature maps with 92×92. The pooling blocks are set to be $m \times m$ such that the size of final

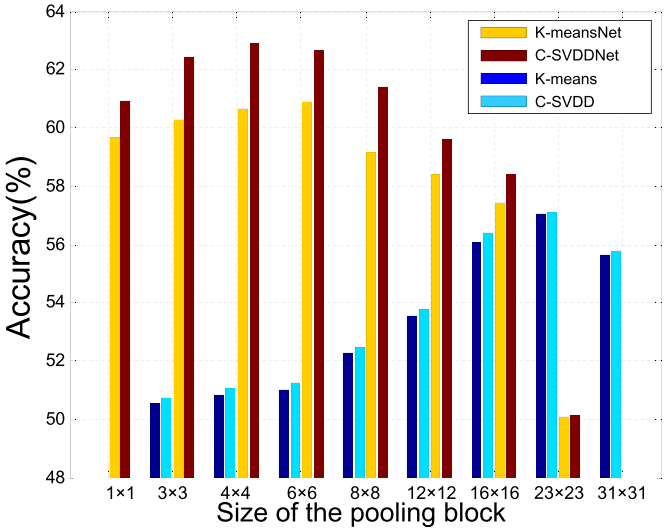

Fig. 9. The effect of different pooling sizes on the performance of the proposed method on the STL-10 dataset.

feature maps after pooling is $\frac{92}{m} \times \frac{92}{m}$. We vary $m \times m$ from 1×1 to 31×31 and record the yielded accuracy. Fig. 9 gives the results under different settings. We can see from the figure that generally for the one layer K-means-based network we need bigger block sizes for improved translation invariance, but adding a robust SIFT encoding layer after pooling effectively reduces the needs for large pooling size while obtaining better performance. One possible reason is that this tends to characterize more detailed information of the objects to be represented.

4.2.3. *Alternative ways to construct global representation*

As illustrated in Fig. 2, the final global/output representation of the proposed method is constructed based on a modified SIFT descriptors. To verify the effectiveness of this method, we also replace this component with several other popular feature descriptors, including LBP [30], SURF [31] and HOG [32]. For fair comparison, we set the parameters of different descriptors in the same configuration, as listed in Table 1.

Fig. 10 gives the results. One can see that the performance of all

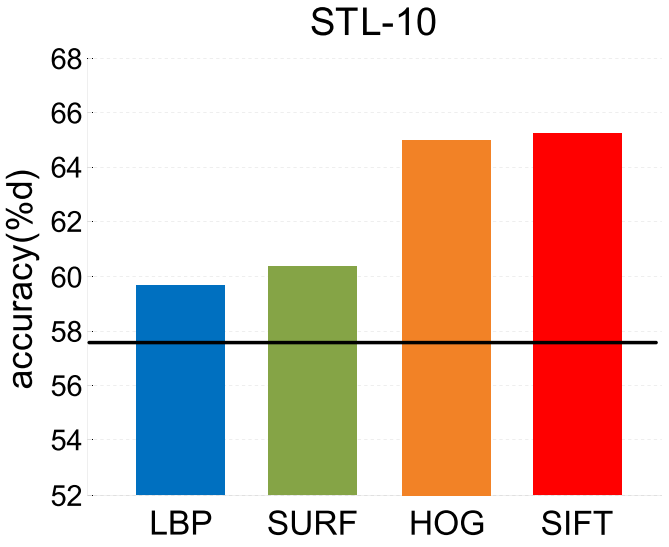

Fig. 10. Comparative performance on STL-10 dataset in the C-SVDDNet with its global representation constructed by various descriptors, i.e., LBP, SURF, HOG and SIFT. The black line is the baseline performance without using any feature descriptors.

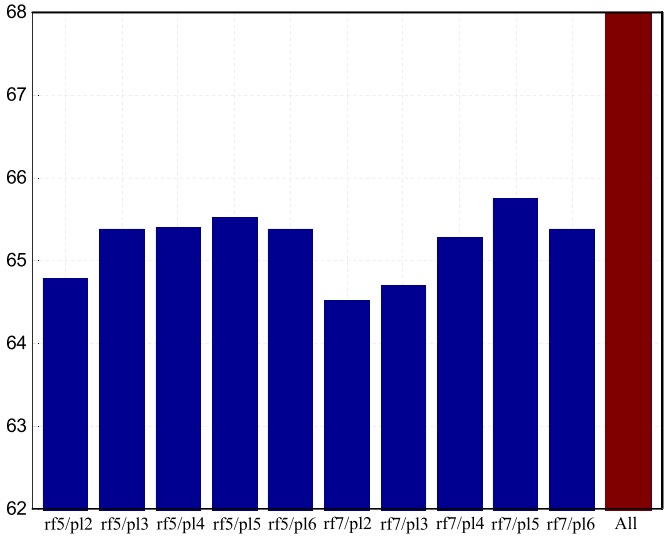

Fig. 11. Detailed performance of 10 different representations and their ensemble on the STL-10 dataset. These representations are obtained by combining a different receptive field size (rf*s*) and a pooling size (pl*s*), where rf*s* indicates a receptive field of $s \times s$, and pl$m$ denotes a pooling block of $m \times m$ in pixel.

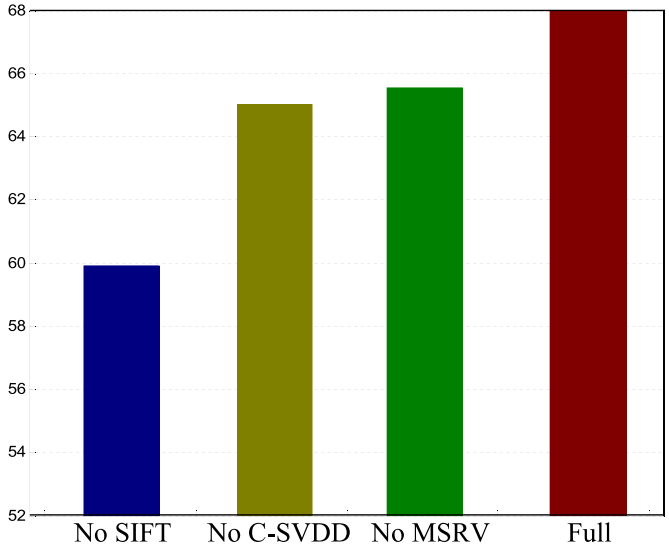

Fig. 12. The contribution of the three major components of the proposed method to the performance.

the descriptors is higher than that of the baseline (i.e, without adopting any feature descriptors, c.f., the black horizontal line in Fig. 10). Note that although SURF is faster than the other three, the modified SIFT and the HoG descriptors perform better in terms of recognition accuracy, showing that the gradient histogram based representation is beneficial for those images in the wild. The LBP descriptor may not be a good choice in our network due to the huge dimension it results in.

4.2.4. *Effect of the multi-scale receptive field voting*

Fig. 11 gives the detailed accuracy of 10 representations using 2 sizes of receptive fields and 5 sizes of pooling blocks. One can see that a different representation leads to a different prediction accuracy but combining them leads to better performance. This shows that the representations captured with different receptive fields and pooling sizes are complementary to each other.

4.2.5. *Contribution of components*

To illustrate the contributions of the individual stages of the proposed method (i.e., C-SVDD local encoding, SIFT-based global representation and multi-scale voting), we conduct a series of experiments on the STL-10 dataset by removing each of the three main stages in turn while leaving the remaining stages in place (the comparison is thus against our full method). Fig. 12 gives the results. In general each stage is beneficial and (not shown) the results are cumulative over the stages, but the SIFT stage seems to contribute most to the performance improvement. This suggests that taking spatial information into global representation is of importance.

4.3. *Object classification*

4.3.1. *STL-10 dataset*

Table 2 gives our results on the STL-10 dataset. The major challenges of this dataset lie in that its images are captured in the wild with cluttered background, objects in various scales and poses. As before, we compared our method with several feature learning methods with state-of-the-art performance. One can see that our one scale C-SVDD network obtains 65.92% accuracy, using a filtering dictionary of 500 atoms, outperforms several other feature encoding methods, such as Bag of Words (BoW), Vector of Linearly Aggregated Descriptors (VLAD), Fisher vector (FV) and other unsupervised deep learning methods (e.g, *Trans. Invariant* RBM (TIRBM) [33], Selective Receptive Fields (SRF) [29], and Discriminative Sum-Product Networks (DSPN) [35]). This also indicates that spatial information preserving using SIFT is indeed useful in unsupervised feature learning. Also note that replacing the proposed C-SVDD encoding with K-means encoding leads to nearly 3.0% performance loss, while fusing the multi-scale information gives us about 2.3% improvement in accuracy, exceeding the current best performer [37] on this challenging dataset.

4.3.2. *MINST dataset*

The MNIST is one of the most popular datasets in pattern recognition. It consists of grey valued images of handwritten digits between 0 and 9. It has a training set of 60,000 examples, and a test set of 10,000 examples, all of which have been size-normalized and centered in a fixed-size image with 28×28 in pixel. In training we use a dictionary with 400 atoms for feature mapping, and after pooling/subsampling we break each feature map into 9 blocks to extract SIFT features. For multi-scale receptive voting, we use 3 types of receptive fields: 5×5, 7×7 and 9×9. Combined these with two settings for pooling sizes (i.e., 1×1 and 2×2, respectively), 6 different views/representations can be obtained for each image in this dataset.

Table 3 gives our experimental results on the MINST dataset. It

Table 3
Comparative performance (%) on the MINST dataset.

Algorithm	Error (%)
Deep Boltzmann machines [28] (2009)	0.95
Convolutional deep belief networks [38] (2009)	0.82
Multi-column deep neural networks [39] (2012)	0.23
Network in network [40] (2013)	0.47
Maxout networks [41] (2013)	0.45
Regularization of neural networks [42] (2013)	0.21
PCANet [22] (2014)	0.62
Deeply-supervised nets [43] (2014)	0.39
K-means (1600 features)	1.01
C-SVDD (1600 features)	0.99
K-meansNet (400 features)	0.45
C-SVDDNet (400 features)	0.43
MSRV+K-meansNet	0.36
MSRV+C-SVDDNet	**0.35**

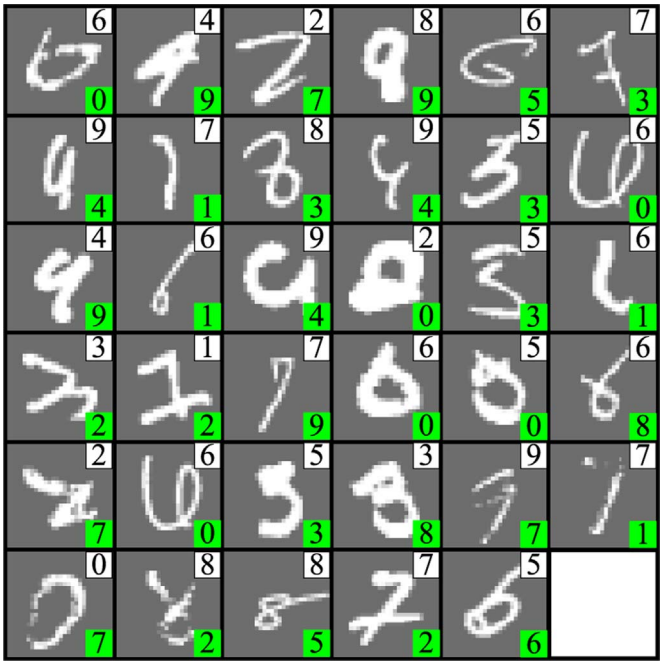

Fig. 13. All misclassified 35 handwritten digits among 10,000 test examples by our method. The small digit in each white square is the ground truth label of the corresponding image, and the one in the green square is the prediction made by our method. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

is well-known that deep learning has achieved great success on this task of digit recognition. For example, only 95 among 10,000 test digits are misclassified by the Deep Boltzmann Machines [28], while Convolutional Deep Belief Networks [38] and Maxout Networks [41] respectively reduce this number to 82 and 45. Our simple single layer network (MSRV+ C-SVDDNet) achieves an error as low as 0.35 %, which is highly competitive to other complex methods using deep architecture. Fig. 13 shows all the 35 misclassified digits by our method, and one can see that these misclassified digits are very confusing even for human beings. Compared to the original K-means network [12], the proposed method reduces the error rate by 65%, with much smaller number of filters. This reveals that at least on this dataset with clean background, it is very beneficial to focus more on the representation of the details of the image, rather than emphasizing too much on its global aspects using a large number of filters and a large pooling size.

4.4. Image retrieval

4.4.1. Holiday dataset

INRIA Holiday dataset consists of 1491 images from personal holiday photos. There are 500 queries, most of which have 1–2 ground truth images. mAP (mean average precision) is employed to measure the retrieval accuracy. We resize all the images to 96×96. In training we use a dictionary with 256 atoms for feature mapping, and after pooling/subsampling we break each feature map into 4 blocks to extract SIFT features. Thus the dimension of final representation is 8196. And we also run PCA for dimensionality reduction as [19]. For multi-scale receptive voting, we use 2 types of receptive fields: 5×5 and 7×7. Combined these with four settings for pooling sizes (i.e., 3×3, 4×4, 5×5 and 6×6, respectively), 8 different views/representations can be obtained for each image in this dataset. Note that in image retrieval task, we cannot train classifiers so that we just concatenate all the views' representations to combine multi-scale information. In retrieval stage we use Euclidean distance in nearest neighbor searching as in [19] and [44], facilitating a fair comparison between various feature representation methods on this task.

Table 4 gives our experimental results on this dataset. We compare our method with BOW, VLAD, FV under different dimensions (reduced through PCA). BoW takes a 20k sized filter bank but has the lowest mAP (45.2%). Replacing BoW with K-means triangle encoding improves mAP by 10% (55.2%), but still needs a large filter bank of 3.2 K.

Previous state-of-art unsupervised feature learning methods, i.e., VLAD and FV [19], can achieve a high mAP of 62.1% 62.6% respectively. And both of them only take a small set of filters of size 256. In [44] Arandjelovic combines VLAD with adaptive filter bank and a new normalization to achieve an accuracy of 64.6%. Our proposed C-SVDDNet can get a mAP of 66.0% with 256 filters as well. It outperforms VLAD by 3.9% and VLAD+adapt+innorm by 1.4%. Even if we reduce its dimension to smaller sizes with PCA, it consistently achieves the best performance among the compared ones.

Also note that replacing K-means encoding with C-SVDD encoding results in significant improvement (from K-means 55.2% to C-SVDD 57.4%, and from K-meansNet 62.5% to C-SVDDNet 66.0%). When concatenating multi-scale representation from 8 views, we are able to achieve the highest mAP of 70.2%, without using any supervision information.

4.4.2. Copydays dataset

INRIA Copydays dataset was designed to evaluate near-duplicate detection [18]. The dataset contains 157 original images. To obtain query images relevant in a copy detection scenario, each image of the dataset has been transformed with three different types of transformation: image resizing, cropping (Here we use

Table 4
Comparative performance (mAP %) on the Holiday dataset.

Algorithm	K	D	Holidays (mAP %)					
			Best	$D' = 2048$	$D' = 512$	$D' = 128$	$D' = 64$	$D' = 32$
BoW [19] (2012)	20,000	20,000	45.2	41.80	44.9	45.2	44.4	41.8
FV [19] (2012)	256	16,384	62.6	62.6	57.0	53.8	50.6	48.6
VLAD [19] (2012)	256	16,384	62.1	62.1	56.7	54.2	51.3	48.1
(2013)	256	16,384	64.6	–	–	62.5	–	–
K-means	3200	12,800	55.2	54.5	54.9	51.6	48.3	44.5
C-SVDD	3200	12,800	57.4	56.8	57.0	53.1	50.5	46.8
K-meansNet	256	8192	62.5	59.8	62.5	61.3	55.5	49.5
C-SVDDNet	256	8192	**66.0**	**63.7**	**65.8**	**64.7**	**59.3**	**52.1**
MSRV+K-meansNet	256	8192×8	66.5	65.0	66.3	65.3	58.3	51.5
MSRV+C-SVDDNet	256	8192×8	**70.2**	**68.6**	**69.8**	**68.5**	**62.5**	**53.8**

Table 5
Comparative performance (mAP %) on the Copydays dataset.

Algorithm	K	D	Crop 50%		Transformations	
			Best	$D' = 128$	best	$D' = 128$
BoW [19] (2012)	20k	20k	100.0	**100.0**	54.3	29.6
FV [19] (2012)	64	4096	98.7	92.7	59.6	41.2
VLAD [19] (2012)	64	4096	97.7	94.2	59.2	42.7
K-means	3200	12,800	95.2	91.5	47.6	32.80
C-SVDD	3200	12,800	97.4	93.8	52.2	36.60
K-meansNet	256	8192	96.8	94.3	55.4	37.80
C-SVDDNet	256	8192	**100.0**	98.1	**62.2**	**52.0**
MSRV+K-meansNet	256	8192×6	99.7	97.9	58.2	41.8
MSRV+C-SVDDNet	256	8192×6	**100.0**	**100.0**	**65.6**	**55.3**

only the queries with the cropping parameter fixed to 50%), strong transformations (print and scan, occlusion, change in contrast, perspective effect, blur, etc). There is in total 229 transformed images, each of which has only a single matching image in the database. All images are resized to 75×75. We use 2 types of receptive fields 5×5 and 7×7, together with three pooling sizes (i.e., 3×3, 4×4 and 5×5 respectively), which result 6 different views. To challenge ourself in this experiments we also merge the database with 10k web images as [19] does.

It is a large scale retrieval task. Table 5 gives our experimental results on this dataset. We can see that in the 50% cropped circumstance, our C-SVDDNet with only 256 filters can be robust enough to achieve a mAP of 100% as BoW with 20k filters. When reducing its dimension to 128 bits, it still performs the second best. In the strong transformation setting, our C-SVDDNet achieves a mAP of 62.2% which outperforms VLAD (59.2%) and FV (59.6%) by nearly 3%.

Furthermore, one can see that C-SVDD encoding allows our C-SVDDNet improve upon K-meansNet by 6.8% in terms of mAP. When reduced to 128 bits, our C-SVDDNet achieves a mAP of 52% under the difficult cases of strong transformation, which outperforms other compared methods by more than 10%, while our multi-scale version improves the mAP by 3%.

5. Conclusion

In this paper, we propose a simple one-layer neural network termed C-SVDDNet for unsupervised feature learning. One of the major advantages of the proposed method is that it allows effective feature representation for many applications, such as object classification and image retrieval, by exploiting unlabeled data which are often cheap and readily available. We show that when properly combined with the SIFT descriptors, such representation could be made even more efficient and discriminant. Extensive experiments on several challenging object classification datasets and image retrieval datasets demonstrate that the proposed method significantly outperforms previous state-of-the-art unsupervised feature learning methods such as Bag of Word, VLAD [19], and FV [20].

Additionally, we show that for feature representation, a very big dictionary is not necessary, as one could accumulate rich information in each feature map and preserve them with compact encoding (e.g, using the proposed method). This significantly reduces the computational cost. Last but not least, we show that one can use multi-scale information to further improve the performance without training many layers of networks—after all training several shallow networks is much easier than training a deep one.

Acknowledgments

This work is partially supported by National Science Foundation of China (61373060) and Qing Lan Project.

References

[1] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, arXiv preprint arXiv:1206.5538, 2012.
[2] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[3] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition? in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 2146–2153.
[4] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1872–1886.
[5] Q.V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, A.Y. Ng, Building high-level features using large scale unsupervised learning, arXiv preprint arXiv:1112.6209, 2011.
[6] G. Carneiro, J.C. Nascimento, A. Freitas, The segmentation of the left ventricle of the heart from ultrasound data using deep learning architectures and derivative-based search methods, IEEE Trans. Image Process. 21 (3) (2012) 968–982.
[7] P.P. San, S.H. Ling, Nuryani, H. Nguyen, Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system, IEEE Trans. Cybern. 44 (8) (2014) 1338–1349.
[8] L. Shuai, Y. Li, Nonlinearly activated neural network for solving time-varying complex Sylvester equation, IEEE Trans. Cybern. 44 (8) (2014) 1397–1407.
[9] A. Agarwal, B. Triggs, Hyperfeatures—multilevel local coding for visual recognition, in: 2006 Proceedings of Ninth European Conference on Computer Vision, Springer, 2006, pp. 30–43.
[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[11] M.A. Cueto, J. Morton, B. Sturmfels, Geometry of the restricted Boltzmann machine, in: M. Viana, H. Wynn (Eds.), Algebraic Methods in Statistics and Probability, Contemporary Mathematics, vol. 516, AMS, 2010, pp. 135–153.
[12] A. Coates, H. Lee, A.Y. Ng, An analysis of single-layer networks in unsupervised feature learning, Ann Arbor 1001 (2010) 48109.
[13] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[14] J. Xu, J. Yao, L. Ni, Fault detection based on svdd and cluster algorithm, in: 2011 International Conference on Electronics, Communications and Control (ICECC), IEEE, 2011, pp. 2050–2052.
[15] A. Banerjee, P. Burlina, Efficient particle filtering via sparse kernel density estimation, IEEE Trans. Image Process. 19 (9) (2010) 2480–2490.
[16] D. Wang, X. Tan, Centering svdd for unsupervised feature representation in object classification, in: Neural Information Processing, Springer, 2013, pp. 376–383.
[17] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[18] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, in: Workshop on Statistical Learning in Computer Vision, ECCV, vol. 1(1–22), Prague, 2004, pp. 1–2.
[19] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, C. Schmid, Aggregating local image descriptors into compact codes, IEEE Trans. Pattern Anal. Mach. Intell. 34 (9) (2012) 1704–1716.
[20] J. Sánchez, F. Perronnin, T. Mensink, J. Verbeek, Image classification with the fisher vector: theory and practice, Int. J. Comput. Vis. 105 (3) (2013) 222–245.
[21] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.
[22] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, Pcanet: a simple deep learning baseline for image classification? arXiv preprint arXiv:1404.3606, 2014.
[23] K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, BMVC 2 (4) (2011) 8.
[24] F. Nie, J. Yuan, H. Huang, Optimal mean robust principal component analysis, in: Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1062–1070.
[25] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, Proc. Br. Mach. Vis. 1 (3) (2015) 6.
[26] H. Jégou, M. Douze, C. Schmid, Hamming embedding and weak geometry consistency for large scale image search-extended version, Technical report, INRIA, 2008.
[27] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, C. Schmid, Evaluation of gist descriptors for web-scale image search, in: Proceedings of the ACM International Conference on Image and Video Retrieval, ACM, 2009, p. 19.
[28] R. Salakhutdinov, G.E. Hinton, Deep Boltzmann machines, in: International Conference on Artificial Intelligence and Statistics, 2009, pp. 448–455.
[29] A. Coates, A.Y. Ng, Selecting receptive fields in deep networks, in: NIPS, vol. 5, 2011, p. 8.
[30] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28

(12) (2006) 2037–2041.
[31] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, in: Computer Vision—ECCV 2006, Springer, 2006, pp. 404–417.
[32] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, IEEE, 2005, pp. 886–893.
[33] K. Sohn, H. Lee, Learning invariant representations with local transformations, arXiv preprint arXiv:1206.6418, 2012.
[34] W.Y. Zou, A.Y. Ng, S. Zhu, K. Yu, Deep learning of invariant features via simulated fixations in video, in: NIPS, 2012, pp. 3212–3220.
[35] R. Gens, P. Domingos, Discriminative learning of sum-product networks, in: NIPS, 2012, pp. 3248–3256.
[36] L. Bo, X. Ren, D. Fox, Unsupervised feature learning for rgb-d based object recognition, in: Experimental Robotics, Springer, 2013, pp. 387–402.
[37] B. Miclut, Committees of deep feedforward networks trained with few data, in: Pattern Recognition, Springer, 2014, pp. 736–742.
[38] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 609–616.
[39] D. Ciresan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3642–3649.
[40] M. Lin, Q. Chen, S. Yan, Network in network, CoRR, vol. abs/1312.4400, 2013. [Online]. Available from: http://arxiv.org/abs/1312.4400.
[41] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, arXiv preprint arXiv:1302.4389, 2013.
[42] L. Wan, M. Zeiler, S. Zhang, Y.L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.
[43] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, arXiv preprint arXiv:1409.5185, 2014.
[44] R. Arandjelovic, A. Zisserman, All about vlad, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2013, pp. 1578–1585.

Dong Wang is a PhD student in Nanjing University of Aeronautics and Astronautics. His research interests are feature representation/selection, Bayesian modeling.

Xiaoyang Tan received a PhD degree from Department of Computer Science and Technology of Nanjing University, China, in 2005. From 2006 to 2007, he worked as a postdoctoral researcher in the LEAR team at INRIA, France. His research interests are face recognition, machine learning, pattern recognition, and computer vision.