Memetic Comp. (2017) 9:199–212
DOI 10.1007/s12293-016-0185-2

REGULAR RESEARCH PAPER

# Denoising deep extreme learning machine for sparse representation

**Xiangyi Cheng**[1,2] · **Huaping Liu**[2] · **Xinying Xu**[1] · **Fuchun Sun**[2]

Received: 28 September 2015 / Accepted: 21 March 2016 / Published online: 12 April 2016


**Abstract** In recent years, a great deal of research has focused on the sparse representation for signal. Particularly, a dictionary learning algorithm, K-SVD, is introduced to efficiently learn an redundant dictionary from a set of training signals. Indeed, much progress has been made in different aspects. In addition, there is an interesting technique named extreme learning machine (ELM), which is an single-layer feed-forward neural networks (SLFNs) with a fast learning speed, good generalization and universal classification capability. In this paper, we propose an optimization method about K-SVD, which is an denoising deep extreme learning machines based on autoencoder (DDELM-AE) for sparse representation. In other words, we gain a new learned representation through the DDELM-AE and as the new "input", it makes the conventional K-SVD algorithm perform better. To verify the classification performance of the new method, we conduct extensive experiments on real-world data sets. The performance of the deep models (i.e., Stacked Autoencoder) is comparable. The experimental results indicate the fact that our proposed method is very efficient in the sight of speed and accuracy.

**Keywords** K-SVD · Extreme learning machine · Denoising · Deep ELM-AE · Representation learning

✉ Huaping Liu
hpliu@tsinghua.edu.cn

Xinying Xu
xuxinying@tyut.edu.cn

[1] Department of Electronic Information, Taiyuan University of Technology, Shanxi, China

[2] State Key Laboratory of Intelligent Technology and Systems, TNLIST, Department of Computer Science and Technology, Tsinghua University, Beijing, People's Republic of China

## 1 Introduction

Recently sparse signal reconstruction has gained considerable interests especially since Michael Elad and colleagues introduced the K-SVD algorithm [1,2]. Variations and extensions of sparse representation have been applied to a variety of areas including image denoising [3], image restoration [4,5], and image classification [6–9]. In these fields, using sparsity as a prior leads to state-of-art results [7,10]. In particular, the sparse representation-based classification (SRC) algorithm proposed in [7] uses sparse representation for face recognition. Unlike conventional methods, SRC does not need an explicit feature extraction stage and is robust to the noise. And the superior performance reported in [7] suggests that this is a promising direction for face recognition. Therefore, we can find out that there is a lot of advantages of sparse signal reconstruction.

A signal $\mathbf{y} \in \mathbf{R}^n$ can be represented by an redundant dictionary $\mathbf{D} \in \mathbf{R}^{n \times K}$ which includes $K$ atoms, $\{\mathbf{d}_j\}_{j=1}^K$ under strict sparsity constrains. Using an redundant dictionary, the test signal can be represented by a linear combination of only a few atoms from this dictionary. The effect of sparse coding therefore highly relies on the dictionary. Reference [7] employs the entire set of training samples as the dictionary for sparse coding and achieves impressive performances on face recognition.

With the growing development of deep learning, the concept of the layered architecture of regions in the human brain such as the visual cortex drew much attention [11–13]. And the revival of interest in such deep architectures is because of the discovery of methods [14,15] that proved successful at learning themselves parameters.

On the other hand, Refs. [13,14] showed that a restricted Boltzmann machine (RBM) and auto-encoders could be used for feature engineering [16,17]. In our opinion, feature engi-

neering means how to develop suitable feature descriptors for specified tasks. These engineered features then could be used to train multiple-layer neural network, or deep models. The two types of auto-encoder-based deep models are the stacked auto-encoder(SAE) [13] and the stacked denoising auto-encoder (SDAE) [18]. Both of them are constructed by stacking auto-encoders. Existing results show that deep networks outperform traditional multilayered neural networks.

However, the aforementioned works still face some issues, such as parameter adjustment and the speed of dealing with the dataset. Huang et al. [19] introduced the extreme learning machines as a single-layer feed-forward neural networks with a fast learning speed and good generalization capability, whose hidden node parameters are randomly generated and the output weights are analytically computed. Then, extreme learning machines (ELM) as an emerging technology has achieved exceptional performance in large-scale settings, and is well suited for binary and multi-class classification, as well as regression tasks. Like deep networks, Huang proposed multilayered ELM (ML-ELM) performs layer-by-layer unsupervised learning. And it also introduces the ELM based on autoencoder (ELM-AE), which represents features based on singular values. Similar to deep networks, ML-ELM stacks on top of ELM-AE to create a multilayered neural network [20,21]. It learns significantly faster than existing deep networks, outperforming DBNs, SAEs, and SDAEs and performing on par with DBMs on the MNIST5 database.

Consequently, ELM offers significant advantages such as fast learning speed, ease of implementation, and minimal human intervention [22]. It thus has strong potential as a viable alternative technique for large-scale computing and machine learning. Because of this, there are a lot of innovation and development [23–28] which applies ELM to sparse representation. In our work, the proposed method is also an optimization for the conventional K-SVD algorithm.

To optimize the conventional K-SVD algorithm, much progress has been made in different aspects. Several algorithms have been developed for the task of learning dictionaries. However, dictionary construction during training and sparse coding during testing are typically time-consuming especially there are a large number of classes. And the improvements to the dictionaries are commonly complex. Especially, when feature representations is not good, the performance of K-SVD algorithm will be affected.

Motivated by the drawbacks of the current methods, the new method is proposed in this paper. The proposed method is a preprocessing method, we can also regard it as feature extraction. It means that using the DDELM-AE can extract high level representation, which is much more better than the raw data. And the newly developed feature representation can produce the denoising dictionnary that could boost the performance of the conventional K-SVD.

As mentioned above, the main contributions of this paper are summarized in the following:

1. The proposed method can be a newly developed feature representation method. Using the denoising deep ELM-AE(DDELM-AE) can extract high level representation instead of using the raw signals (e.g., images)) that could boost the performance of the conventional K-SVD. That means, we employ a denoising "input" of the raw data to the off-the-shelf K-SVD algorithm, which is generated by (DDELM-AE) and more stable and robust than the original data.
2. Then, through the denoising "input", the conventional K-SVD can get a denoising dictionary. Such a denoising dictionary is critical to K-SVD and dictionary learning.

Last, according to the smaller restructure error gained by the new approach, we achieve the image classification problem. And our best results are much better than the simple K-SVD. Specially, we gain the test accuracies 96.1, 99.79 % on USPS and Coil-20 database.

To demonstrate the effectiveness and the advantage of the proposed method for face recognition, extensive experiments have been carried out using the commonly-used face dataset: the Extended YaleB dataset. In particular, we obtained the test accuracy 94.23 %.

This paper is organized as follows. In Sect. 2, we first briefly review the conventional K-SVD algorithm and OMP algorithm. Then we introduced the ELM and the ELM based on autoencoder in Sect. 3. Section 4 presents our proposed method in detail. Following this part, we demonstrate the experimental results and analyse the effect of different parameter in Sect. 5. Finally, Sect. 6 concludes the paper with the summary and demonstrates the superiority of our proposed method.

## 2 Related work

Many optimization methods to traditional and complex systems have been proposed [36], the conventional K-SVD algorithm is no exception. Indeed, much progress has been made in different aspects. Several algorithms have been developed for the task of learning dictionaries. Two of the most well-known algorithms are the method of optimal directions (MOD) [41] and the K-SVD algorithm [2].

The MOD algorithm updates all the atoms simultaneously. This way of updating the dictionary is essentially the idea behind the MOD method. As discussed in [2], one of the major drawbacks of the MOD method is that it suffers from the high complexity of matrix inversion during the dictionary update stage. Several other methods have also been proposed that focus on reducing this complexity. One such algorithm

is K-SVD. In the K-SVD, the dictionary update is performed atom-by-atom in an efficient way rather than using a matrix inversion. It has been observed that the K-SVD algorithm requires fewer iterations to converge than the MOD method. Therefore, the K-SVD algorithm is adopted to improve in this paper rather than MOD method.

### 2.1 Sparse representation and dictionary learning: K-SVD algorithm

It is due to the fact that signals and images of interest can be sparsely represented or compressible given an appropriate dictionary, hence, sparse and redundant signal representations have recently drawn much interest in computer vision, signal analysis and image processing [37–40].

A signal $\mathbf{y} \in \mathbf{R}^n$ can be represented by an redundant dictionary $\mathbf{D} \in R^{n\times K}$ which includes $K$ atoms, $\{\mathbf{d}_j\}_{j=1}^K$. It can be exact or approximate linear reconstruction of certain columns of $\mathbf{D}$. Finding a sparse coding vector entails solving the following optimization problem

$$\begin{aligned} &\widehat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}_0\|, \\ &s.t.\ \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \le \varepsilon, \end{aligned} \tag{1}$$

where $\varepsilon$ is an error tolerance, $\|\mathbf{x}_0\|$ is the $\ell_0$ sparsity measure that counts the number of nonzero elements in the $\mathbf{x}$, $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2$ is the mean squared error resulting from sparse approximation.

In this work, we adopt the K-SVD algorithm [2] for development. Given a set of $N$ signals $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, the goal of K-SVD algorithms is to find a dictionary $\mathbf{D}$ and a sparse coding matrix $\mathbf{X}$ which solves the following optimization problem:

$$\begin{aligned} &(\hat{\mathbf{D}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2, \\ &s.t.\ \|\mathbf{x}_i\|_0 \le T_0, \quad \forall i = 1, \ldots, N, \end{aligned} \tag{2}$$

where $\mathbf{x}_i$ represents the $i$th column of $\mathbf{X}$, $\|\mathbf{A}\|_F$ denotes the Frobenius norm of $\mathbf{A}$, and $T_0$ denotes the sparsity level. K-SVD is an iterative method that alternates between sparse-coding and dictionary update steps. First, a dictionary $\mathbf{D}$ with $\ell_2$ normalized columns is initialized. Then, the main iteration is composed of the following two stages:

1. Sparse coding: In this step, we fix $\mathbf{D}$ and solve the following optimization problem over $\mathbf{x}_i$ for each example $\mathbf{y}_i$

$$\begin{aligned} &\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2, \\ &s.t.\ \|\mathbf{x}_i\|_0 \le T_0, \qquad \forall i = 1, \ldots, N. \end{aligned} \tag{3}$$

2. Dictionary update: In this step, we fix the coding coefficient matrix and update the dictionary atom-by-atom in an efficient way.

With an update of dictionary columns and combining with an update of the sparse representations, traditional K-SVD algorithm achieves sparse signal representations from the raw signals. However, untreated data may be noisy, it is against this algorithm itself. According to the reconstruct error it inevitably leads to a poor classification result.

### 2.2 Orthogonal matching pursuit algorithm

As we know, exact determination of sparsest representations proves to be an NP-hard problem. So approximate solutions are considered instead. In the past decade, several efficient persuit algorithms have been proposed such as matching pursuit (MP) [42], basis pursuit (BP) [43]. Like MP algorithm, the problem above can be solved by greedy iterative algorithm, one of the most commonly used algorithm is the orthogonal matching pursuit (OMP) method [44].

Given a set of $N$ signals $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, which are belong to the $c$ classes. Through the above mentioned K-SVD algorithm, we can get the dictionary of each class, $\mathbf{D}_i \in \mathbf{R}^{d\times K}$, whose columns are the measurement vectors, where $i \in \{1, 2, \ldots, c\}$. And $K$ is the number of atoms learned of each dictionary. Thus, through the OMP algorithm, we can gain a linear combination of $m$ columns from $\mathbf{D}_i$, using the linear combination reconstructs a new signal, which is different from the original one. Thus, the signal recovery problem can be handled.

## 3 Principle of ELM based on autoencoder

### 3.1 Review of extreme learning machine

As an emerging technology, extreme learning machine (ELM) has been demonstrated to have excellent learning accuracy and fast speed. However, with such exceptional performance, ELM is originally derived from the single hidden layer feedforward neural networks (SLFNs) [29] shown in Fig. 1 and then extended to the generalized SLFNs. Different from traditional learning algorithms [30], like back propagation (BP) based beural networks and support vector machine (SVM), the most outstanding characteristic of ELM are learning without iteratively tuning hidden neurons in general architectures, generating the weight matrix randomly between the input layer and the hidden layer and then calculate the output weights by the least-square method.

The ELM for SLFNs shows that hidden nodes can be randomly generated. Given $N$ training samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$, the

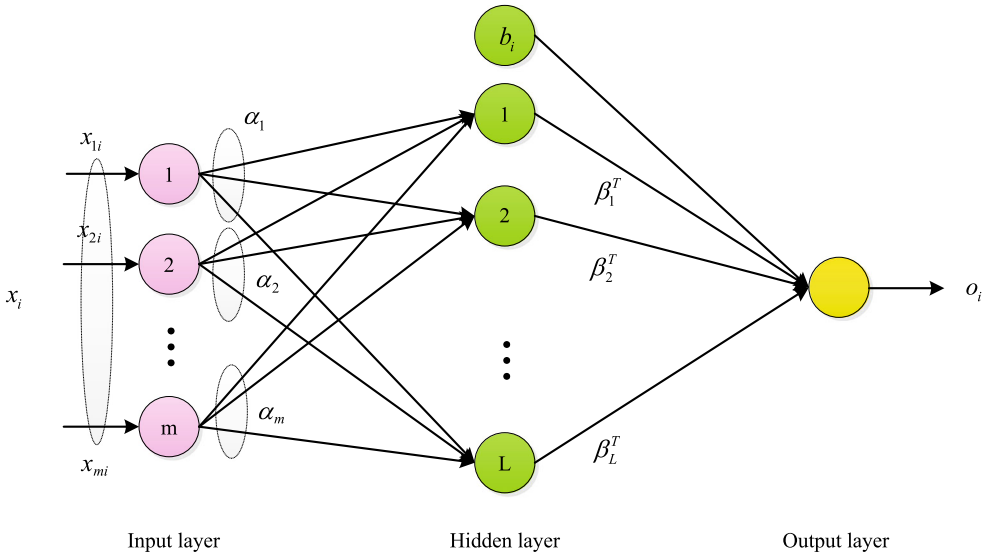



**Fig. 1** The framework of extreme learning machine

input data $\mathbf{x}_i$ is mapped to L-dimensional ELM random feature space, and the network output is

$$\mathbf{o}_i = \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = \sum_{j=1}^{L} \boldsymbol{\beta}_j \mathbf{G}(\alpha_{ji}, b_{ji}, \mathbf{x}_i), \tag{4}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^T$ is the output weight matrix between the hidden nodes and the output nodes, $\mathbf{h}(\mathbf{x}_i)$ is the output of hidden nodes.

Then, extreme learning machine can resolve the following learning problem:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{5}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$ are target labels, and

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\alpha_1, b_1, \mathbf{x}_1) & \cdots & G(\alpha_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\alpha_1, b_1, \mathbf{x}_N) & \cdots & G(\alpha_L, b_L, \mathbf{x}_N) \end{bmatrix}, \tag{6}$$

If $\mathbf{H}$ is nonsquare matrix and the smallest norm least-square solution of the above linear system is:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{7}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

To improve generalization performance and make the solution more robust, we can add a regularization term as shown [31]:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{8}$$

### 3.2 ELM based on autoencoder

In this section, we describe a common framework about deep ELM based on autoencoder [32] used for representation learning.

ELM-AE's main objective to represent the input features meaningfully in three different representations [33], one is compressed, which is representing features from a higher dimensional input data space to a lower dimensional feature space, the other is equal, that means feature space dimension is equal to input data space dimension.

Hence, ELM can be modified as follows: input data is used as output data $\mathbf{t} = \mathbf{x}$, and random weights and biases of the hidden nodes are chosen to be orthogonal. Bernard Widrow et al. [34] introduced a least mean square (LMS) implementation for the ELM and a corresponding ELM based on autoencoder that uses nonorthogonal random hidden parameters (weights and biases). Orthogonalization of these randomly generated hidden parameters tends to improve ELM-AE's generalization performance.

In ELM-AE, the orthogonal random weights and biases of the hidden nodes project the input data to a different or equal dimension space, as shown by the Johnson-Lindenstrauss lemma [35] and calculated as

$$\begin{aligned} &\mathbf{h} = g(\mathbf{a}\mathbf{x} + b), \\ &\mathbf{a}^T\mathbf{a} = \mathbf{I}, b^T b = 1, \end{aligned} \tag{9}$$

where $\mathbf{a} = [\mathbf{a}_1, \ldots, \mathbf{a}_L]$ are the orthogonal random weights, and $\mathbf{b} = [b_1, \ldots, b_L]$ are the orthogonal random biases between the input and hidden nodes.

As stated above, ELM-AE's attractive property is that the output data is actually the input data, thus, for compressed and equal ELM-AE representation, we can calculate the output weights $\boldsymbol{\beta}$ as follows:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathbf{T}}\mathbf{H}\right)^{-1} \mathbf{H}^{\mathbf{T}}\mathbf{X}. \tag{10}$$

Finally, we learn representation in an unsupervised way using an ELM based on autoencoder, which in essence is a multi-layer feed-forward network whose parameters are learned by cascading multiple layers of ELM. Experimental results also turn out that the learning procedure of ELM-AE is highly efficient and has good generalization capabilities.

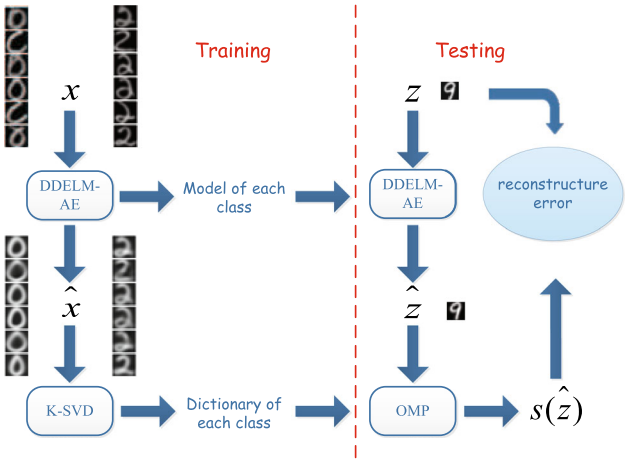



**Fig. 2** The whole process of our proposed approach

## 4 Proposed method

Several algorithms have been developed for the task of learning dictionaries. Motivated by the drawbacks of the current methods and the needs of many practical applications, the new method is proposed in this paper. The proposed method is a newly developed feature representation. It means that using the DDELM-AE can extract high level representation, which is much more better than the raw data. And high level representation can produce the denoising dictionary that could boost the performance of the conventional K-SVD.

In our paper, we use high level of representations as "input" rather than the original image, which is extracted by the denoising deep ELM based on autoencoder (DDELM-AE). See Fig. 2. On such a basis, K-SVD algorithm can gain much better dictionaries of each class and reconstructions of each test sample, and then estimate its label based on the smallest reconstruction error. Finally, all testing samples are classified as it should belong to the certain class. Results demonstrate that high level of features can be better preserved and can reduce the reconstruction error effectively.

Since the introduction of the frame of the ELM-AE, in this part, we intend to focus on our scheme for multilayered representation, and how this "deep" representation creates a meaningful learning used for the sparse representation.

### 4.1 Learning representation with denoising deep ELM-AE

For K-SVD algorithm itself, the original data are usually taken as its input. So we cannot defy that learning high level of representations is vital for achieving better performance. We can often see stacked autoencoders (SAEs) and stacked denoising autoencoders (SDAEs), whose outputs are equal the real input. Furthermore, many deep neural networks have yielded good performance in various tasks [12], but they are generally very slow in training phase. Considering the above factors, we employ the representations as "input" learned by our proposed method: denoising deep ELM-AE, which possesses obvious advantages in the calculation speed, even though the high dimensional image.

In our paper, we set the output of an ELM network equal to the input, then, we will get the new representation $\widehat{X}$ from the denoising deep ELM-AE(DDELM-AE). Figure 3 shows the process of a denoising deep ELM-AE model learning the representations of the training set $X$ and what is the representation of $X$ ultimately.

We consider a fully connected deep network with $L$ hidden layers and $\mathbf{W} = \{\mathbf{W}^1, \mathbf{W}^2, \ldots \mathbf{W}^{L+1}\}$ to denote the parameters of the DDELM-AE that need to be learned, namely, $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_{L+1}\}$. To reduce the training cost, each layer is decoupled within the network and processed as an single ELM, of which targets are the same as its inputs. As shown in Fig. 3, $\mathbf{W}^1$, in other words, $\boldsymbol{\beta}_1^{\mathrm{T}}$ is learned by considering a corresponding ELM with $\mathbf{T} = \mathbf{X}$.

The weight vectors connecting the input layer to each unit of the first hidden layer are orthonormal to each other, effectively leading to projection of the input data to a random subspace. Compared to initializing random weights independent of each other, orthogonalization of these random weights tends to better preserve pairwise distances in the random ELM feature space [35] and improves denoising deep ELM based on autoencoder generalization performance. Next, $\boldsymbol{\beta}_1$ is calculated by Eq. (7) depending on the number of nodes in the hidden layer. Note that, $\boldsymbol{\beta}_1$ re-projects the lower dimensional representation of the input data back to its original space while minimizing the reconstruction error. Therefore, this projection matrix is data-driven and hence used as the weights of the first layer ($\mathbf{W}^1 = \boldsymbol{\beta}_1^{\mathrm{T}}$).

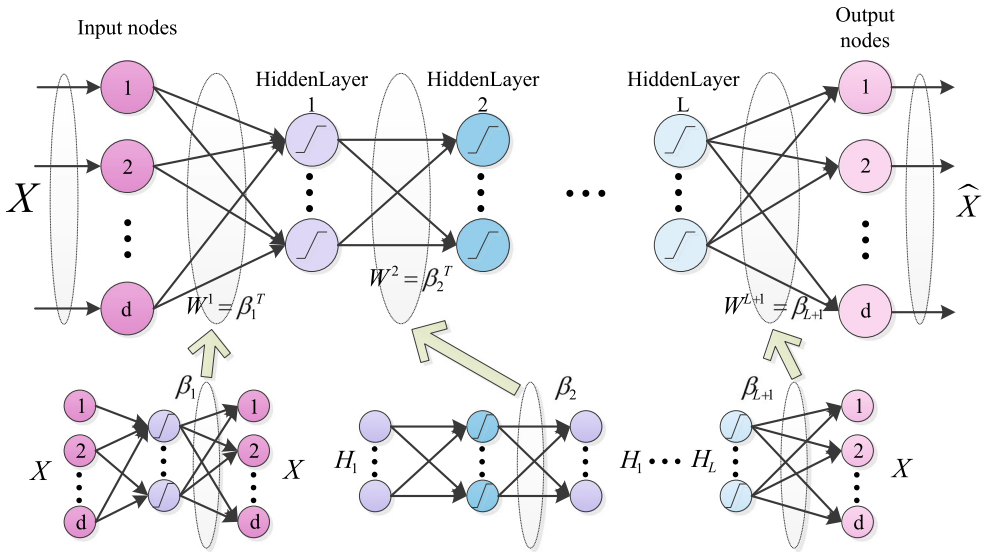



**Fig. 3** A denoising deep ELM-AE model for the samples of the training set X

$$\boldsymbol{\beta}^* = \min \|\mathbf{H}\boldsymbol{\beta} - \mathbf{X}\|_F^2, \quad s.t.\ \boldsymbol{\beta}^T\boldsymbol{\beta} = \mathbf{I}. \tag{11}$$

Similarly, the value of $\mathbf{W}^2$ is learned by forcing that the input and output of Hidden Layer 2 to $\mathbf{H}_1$ i.e. the output of Hidden Layer 1. In this way, all parameters of the multilayered ELM can be computed step by step. However, when the number of nodes between two consecutive layers is equal, the random projection obtained in the second layer is in the same space as the input of the first layer. Using (7) does not ensure orthogonality of the computed weight matrix $\boldsymbol{\beta}$. Imposing orthogonality in this case results in a more accurate solution since the data always lies in the same space. Therefore, the output weights $\boldsymbol{\beta}$ are calculated as the solution to the orthogonal procrustes problem.

The closed form solution is obtained by finding the nearest orthogonal matrix to the given $\mathbf{M} = \mathbf{H}^T\mathbf{X}$. To find the orthogonal $\boldsymbol{\beta}^*$, we use the singular value decomposition $\mathbf{M} = \mathbf{U}\sum\mathbf{V}^T$ to compute $\boldsymbol{\beta}^* = \mathbf{U}\mathbf{V}^T$.

In DDELM-AE, the orthogonal random weights and biases of the hidden nodes project the input data to a different or equal dimension space. The deep ELM-AE models can automatically learn the non-linear structure of data in a very efficient manner. Compared with deep neural networks, DDELM-AE does not require expensive iterations of fine tuning.

### 4.2 Using a denoising representation and a denoising dictionary

According to the above described structure of DDELM-AE, we can achieve a denoising representation from the original data. Figure 4 shows a denoising representation of the procedure. Conventional autoencoder generates a simply copy of the input or similarly uninteresting ones trivially maximizes mutual information. A wide variety of modification of the traditional autoencoder framework have been proposed in order to learn sparse representations [13]. Pascal Vincent and colleagues introduced a very different strategy and defined a new representation into the mentioned below: "a good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input".

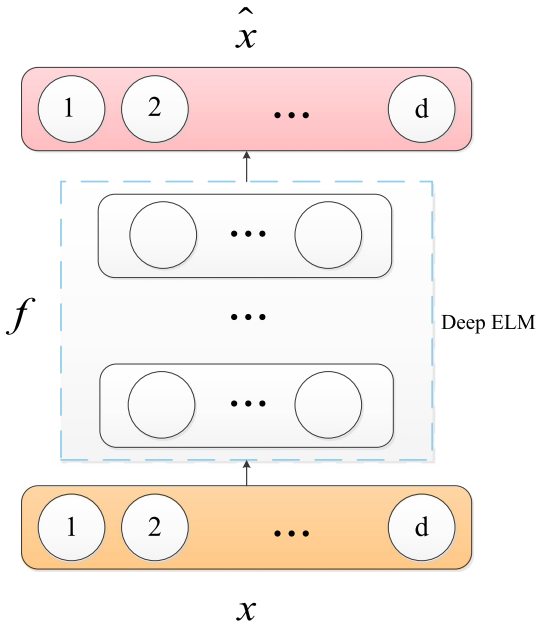



**Fig. 4** The denoising deep ELM-AE architecture

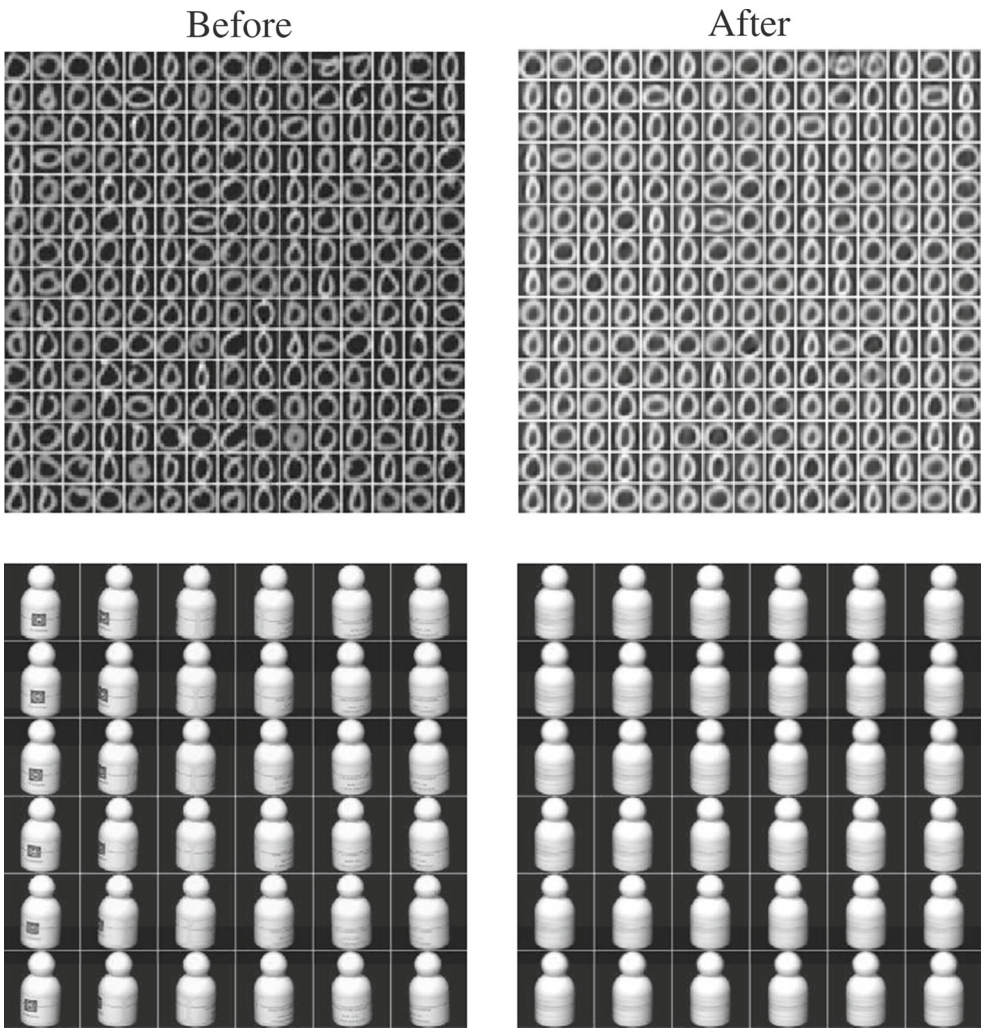



**Fig. 5** Representation before and after performing the DDELM-AE

In conclusion, here we propose a similar but different method. For the above-mentioned DDELM-AE, we can gain the weights of each layer. Based on each layer decoupled within the network and processed as an ELM, particularly, the last layer whose output is the image of input. Then, we desired not to get the same as the input but a clean or denoising "input", $\hat{\mathbf{X}}$, which is reconstructed by our network.

Using DDELM-AE, we get a clean input $\hat{\mathbf{x}} = f(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$, comparing with the initial input $\mathbf{x}$. Figure 5 shows two pairs of samples, the former is the initial input, the latter is representations applied the clever mapping $f$ of DDELM-AE. What the representation DDELM-AE has learned demonstrates the features are denoised.

Through the above high level representation, we get a denoising dictionary by the conventional K-SVD algorithm. See Fig. 6. Take the USPS database as an example. That demonstrates the new method can get a denoising dictionary in the training phase, rather than a noisy dictionary.

For a fair comparison, the learned dictionary by the conventional K-SVD is shown in the next picture.

Obviously, the dictionaries learned by the proposed method are more clear and more stable than anyone learned by the K-SVD algorithm.

## 5 Experimental results and analysis

In this section, we examine the efficiency of the new method by comparing it against the conventional K-SVD algorithm. And, considering that the state-of-the-art deep learning method, stacked auto-encoder(SAE) has an important reference for our algorithm, we use it for comparision. As a note, it is not used as a preprocessing step for K-SVD algorithm. Furthermore, we make an attempt to other technique such as sparse filtering [46] to verify whether it can also be used before K-SVD in order to boost its performance or not. Experiments present that the results are unsatisfactory.

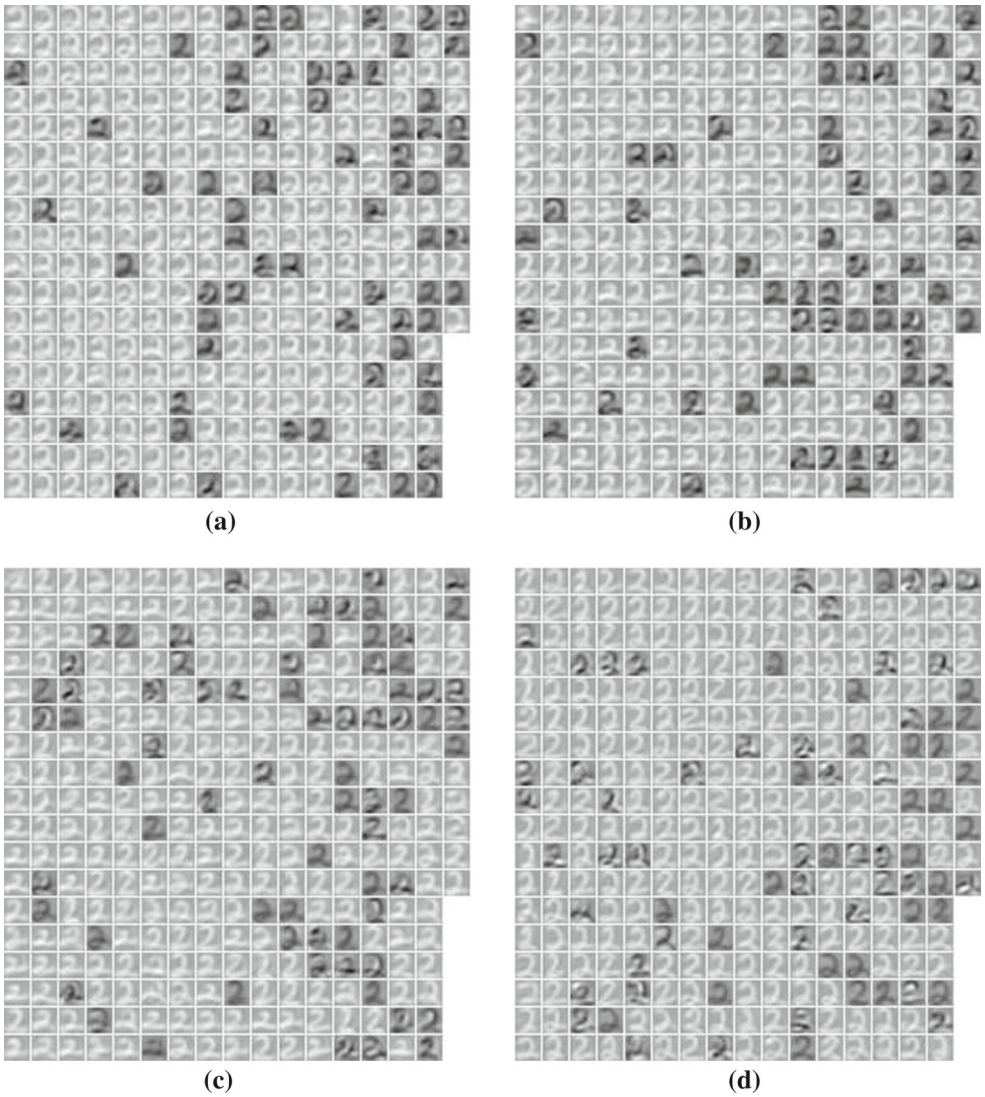



**Fig. 6** With different hidden nodes in the DDELM-AE structure, different dictionaries are used for K-SVD. **a–d** are in the case of 20, 30, 50, 100 hidden nodes respectively

In the same case of K-SVD's parameters, such as $T_0 = 5$, maximum number of training iterations is set to 80, the above framework includes a number of parameters that can be changed : (i) the number of hidden layers, $L$, (ii) the number nodes of hidden layers of DDELM-AE, (iii) the ridge parameter $C = [C_1, C_2, C_3]$. In this section, we present our experimental results on the impact of these parameters on performance.

First, we will evaluate the effects of these parameters on the USPS dataset and Coil-20 dataset respectively. Furthermore, in order to demonstrate the advantage of the proposed method, experiments are conducted on the Extended YaleB dataset (Fig. 7). Secondly, we will report the results achieved on all of them. Besides, the parameter settings that our analysis suggests is best overall (i.e., in our final results, we use the same setting for K-SVD algorithm).

We conducted the experiments on a laptop with a core i5-3337U 1.80GHz processor and 4 Gbytes of RAM running Matlab 2013a.

### 5.1 Digit recognition

We apply our approach on the real-world handwritten digits classification problem. We use the USPS database [45] shown in Fig. 8, which contains ten classes of 256-dimensional handwritten digits. In other words, input training samples are the vectorization of USPS digit images with the dimension of $n = 256$. For each class, we select $N_{training} = 500$ samples for training and $N_{test} = 200$ samples for testing. Specifically, we choose the following parameters for learning the dictionaries for all classes: each class dictionary is

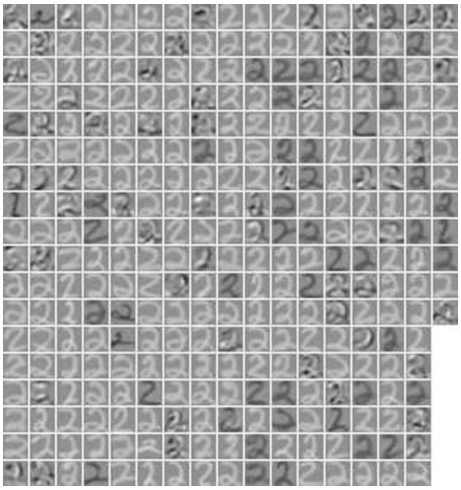


**Fig. 7** The dictionary learned by the conventional K-SVD

**Fig. 8** Random samples on the USPS database [45]

learned with $K = 300$ atoms, $T_0 = 5$, maximum number of training iterations is set to 80.

We use $\mathbf{Y}_i = [\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,N}] \in \mathbf{R}^{256\times 500}$ to represent the set of training samples of the $i$-th class, where $i \in \{1, \ldots, 10\}$. During our training procedure, there are two sequential stages: we first learn the stable and robust representations through a DDELM-AE. In the meanwhile, we also get 10 different kinds of deep ELM-AE model used to reconstruct testing samples later. The whole learning process is a very efficient and rapid manner in comparison to autoencoder.

In the second stage, K-SVD is applied to get the dictionary of the training set, $\mathbf{D}_i \in \mathbf{R}^{256\times 300}$, where $i \in \{1, \ldots, 10\}$. In other words, all above is to get the model of each class and the dictionary of every training class.

In the test phase, given a query image $\mathbf{z} \in \mathbf{R}^{256\times 1}$, we first perform the DDELM-AE to get its denoising representation and then implement OMP algorithm (defined function $s$) separately for each $\mathbf{D}_i$, as shown in Fig. 3, to get the sparse code $\mathbf{x}_i$. The sparse setting is the same as the training phase, namely $T_0 = 5$. Finally, the reconstruction error $r_i$ is computed as:

**Table 1** Comparison of digit recognition accuracies for various hidden nodes in the case of two hidden layers

| Structure | Accuracy (%) | Structure | Accuracy (%) |
|---|---|---|---|
| 200-50 | 85.1 | 100-50 | 85.55 |
| 200-100 | 85.3 | 100-100 | 95.4 |
| 200-200 | 95.3 | 100-200 | 84.35 |
| 50-50 | 95.65 | 20-20 | 95.45 |
| 50-100 | 84.4 | 20-50 | 84.9 |
| 50-200 | 84.4 | 20-100 | 84.8 |

$$r_{(i,N)} = \|\mathbf{z} - \mathbf{D}_i\mathbf{x}_i\|_F^2 = \|\mathbf{z} - (\mathbf{Y}_i\mathbf{X})\mathbf{x}_i\|_F^2,$$
$$i \in \{1, \ldots, 10\}, N = \{1, \ldots, 2000\}, \quad (12)$$

where $\mathbf{X} \in \mathbf{R}^{256\times 300}$ is sparse coefficient matrix of training samples, $\mathbf{x}_i \in \mathbf{R}^{300\times 1}$ is sparse coefficient matrix of each testing sample and $N$ is the total number of testing samples. The test sample is simply classified to the class that gives the smallest reconstruction error.

### *5.1.1 Number of hidden layers*

Our experiments consider that how many hidden layers are favorable for the denoising DELM-AE. Through extensive experiments we find that 3 hidden layers based on our method is better than 2 layers. Furthermore, we also realize that increasing the number of hidden layer is not too good in surprise.

Before we present classification results, we first show the effect of different hidden nodes between hidden layers.

Table 1 shows that the effect of with different nodes in the case of two hidden layers. Extensive experiments turn out 3 hidden layers of DDELM-AE and 50 hidden nodes in each layer will perform better.

### *5.1.2 Ridge parameters*

Through the above experiments, we get a certain conclusion that we should adapt 3 hidden layers. And we speculate we may get the best result with 50 neurons in each layer. Next, we will further confirm the parameter $C = [C_1, C_2, C_3]$. We set the parameter $C$ in the range from $\{10^{-2}, 10^{10}\}$ from the first hidden layer to the last one. We follow this protocol: for example, we make $C_1$ range from $10^{-2}, 10^{10}$ with fixing the other two parameters $C_2$ and $C_3$. Then we can get the best accuracy using the most proper $C_1$. And so on, for each parameter of $C$.

Experiments are repeated with different nodes but all is 3 layers. There are 4 groups of figures which show the effect of $C_1, C_2$ and $C_3$ to relevant layer with different nodes. In

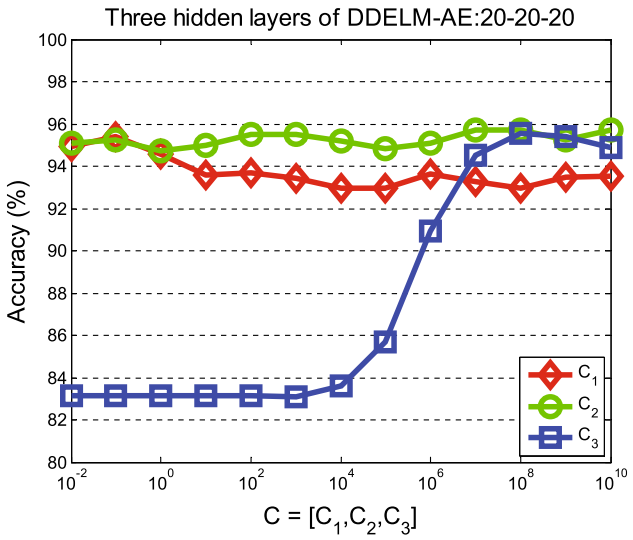



**Fig. 9** Effect of different parameters $C = [C_1, C_2, C_3]$ to each layers with 20 nodes. We follow this protocol: for example, we make $C_1$ range from $\{10^{-2}, 10^{10}\}$ with fixing the other two parameters $C_2$ and $C_3$. Then we can get the best accuracy using the most proper $C_1$. And so on, for each parameter of $C$

**Table 2** Test recognition accuracy and consumed time on USPS. The unit of the consumed time is second

| Algorithm | Accuracy (%) | Consumed Time |
|---|---|---|
| K-SVD | 93.10 | 2067.46 |
| SAE | 95.25 | 60 |
| Proposed method | 96.10 | 2011.57 |

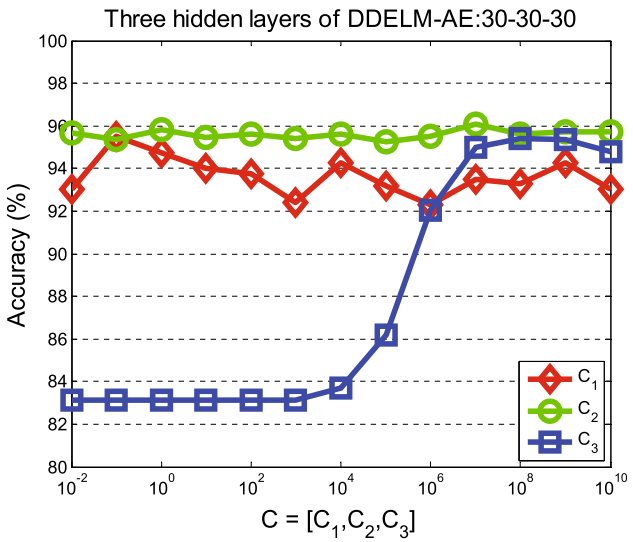



**Fig. 10** Effect of different parameters $C = [C_1, C_2, C_3]$ to each layers with 30 nodes

Figs. 8 and 9, we can see that with 20 or 30 nodes in each layer, $C_1 = 0.1$, $C_2 = 10^7$, $C_3 = 10^8$ leads to the best result 95.7 and 96.1 % respectively (Figs. 10, 11, 12, 13, 14).

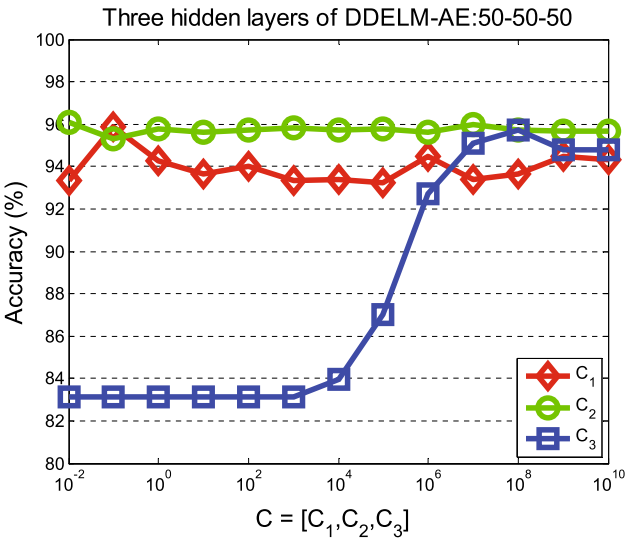



**Fig. 11** Effect of different parameters $C = [C_1, C_2, C_3]$ to each layers with 50 nodes

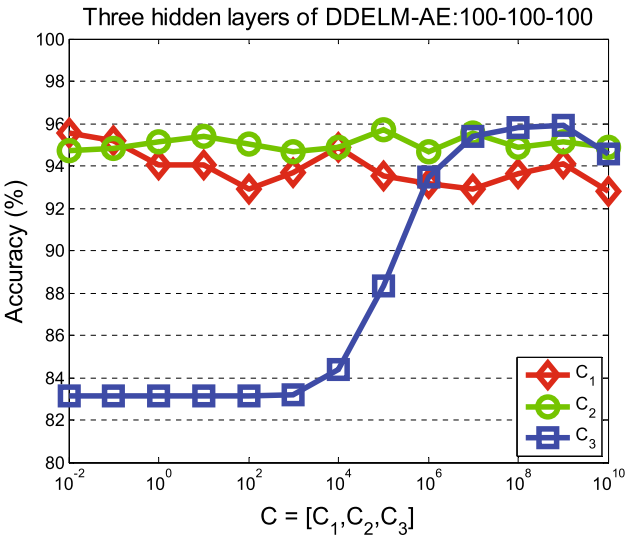



**Fig. 12** Effect of different parameters $C = [C_1, C_2, C_3]$ to each layers with 100 nodes

We can also obtain the best testing accuracy 96.1 % in the case of 50 nodes when $C_1 = 0.1, C_2 = 0.01, C_3 = 10^8$. Moreover, the result with 100 nodes is a little poor, only 95.7 % when $C_1 = 0.1$, $C_2 = 10^5$, $C_3 = 10^8$.

### *5.1.3 Final classification results*

As we have mentioned, we employ the best and the most applicable number of neurons and hidden layers on the USPS database. In Table 2 we can see that our proposed method performs better obviously compared with the traditional K-SVD algorithm and SAE algorithm.


**Fig. 13** Examples of 20 classes in Coil-20

### 5.2 Coil-20 recognition

In this part, we will report the image set classification results on the Coil-20 database, which is established by Columbia Object Image Library and contains 20 classes objects as shown in the following figure.

The data set consists of gray-level images with $128 \times 128 = 16{,}384$ pixels in 20 classes. Each class includes 72 images, we take 50 of them as training samples, the rest are chosen to test. So there are 1000 samples for training and 440 samples for testing in total.

As same as the setting of USPS, we set $T_0 = 5$, maximum number of training iterations is still set to 80, and input training samples are the vectorization of Coil-20 images with the dimension of $n = 16{,}384$. Similarly, we employ $\mathbf{Y}_i = [\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,N}] \in \mathbf{R}^{16{,}384\times 50}$ to represent the set of training samples of the $i$-th class, where $i \in \{1, \ldots, 20\}$. And it should be emphasized that each class dictionary is learned with $K = 30$ atoms to use the K-SVD algorithm. There are still two sequential stages during our training procedure like the experiments on the USPS dataset.

In the test phase, given a query image $\mathbf{z} \in \mathbf{R}^{16384\times 1}$, we first perform the DDELM-AE to get its denoising representation and then implement OMP algorithm(defined function $s$) separately for each $\mathbf{D}_i$. The test sample is classified to the class that gives the smallest reconstruction error.

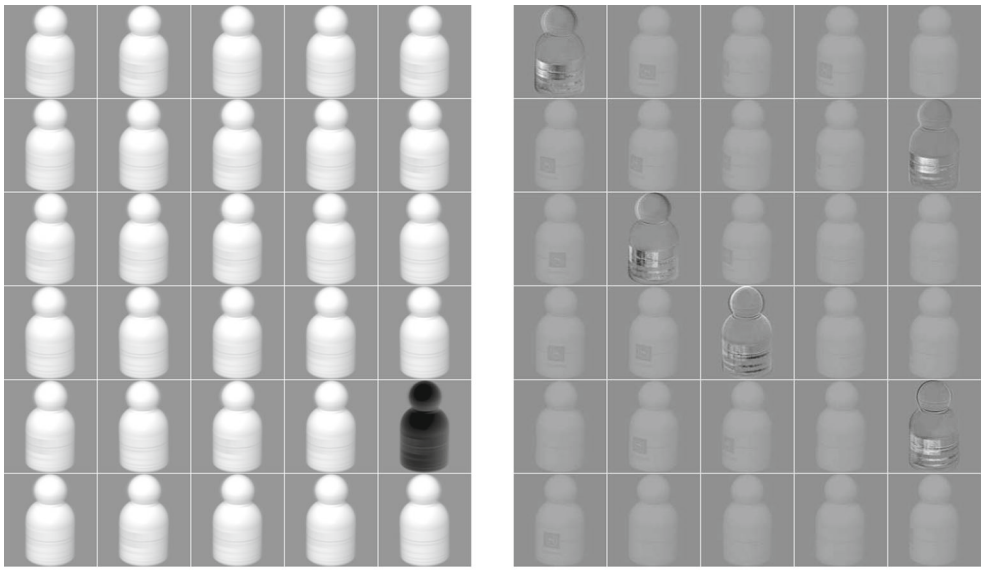
**Table 3** Comparison of Coil-20 recognition accuracies for various hidden nodes in the case of two hidden layers

| Structure | Accuracy (%) |
|---|---|
| 20-20 | 99.55 |
| 20-50 | 99.41 |
| 20-100 | 99.37 |
| 20-200 | 99.32 |

#### *5.2.1 Number of hidden layers*

We use the same parameters setting and verify that 3 hidden layers in the DDELM-AE are better than 2 layers, and more layers are not useful to the result. Table 3 states the results with different nodes in 2 layers.

Therefore, we decide to use 3 hidden layers with 20 neurons in each layer.



**(a)** **(b)**

**Fig. 14** The denoising dictionary and noising dictionary

**Table 4** Test recognition accuracy and consumed time on Coil-20

| Algorithm | Accuracy (%) | Consumed Time |
|---|---|---|
| K-SVD | 91.36 | 4051.39 |
| SAE | 76.01 | 21,600 |
| Proposed method | 99.79 | 3410.40 |

#### *5.2.2 Ridge parameters*

As same as the method of USPS, we still perform the same experiments on Coil-20 to verify the effect of various $C$ for the results. We find $C_1 = 10$, $C_2 = 10^3$, $C_3 = 10^8$ generates the best testing accuracy.

#### *5.2.3 Final classification results*

Experimental results show that the proposed method outperforms the conventional K-SVD in the view of classification rate, so the proposed method has more practical value. Our best result is illustrated in Table 4 and in Fig. 15.

### 5.3 Extended YaleB recognition

The Extended YaleB database contains about 2414 frontal face images of 38 individuals and around 64 images under different illuminations per individual. Further, Professor Cai and colleagues resized them to 32 × 32 pixels shown in Fig. 16. We randomly split the dataset into two halves (Table 5). One half, which contains 51 images for each person, was used for training the dictionary. The others were used for testing.

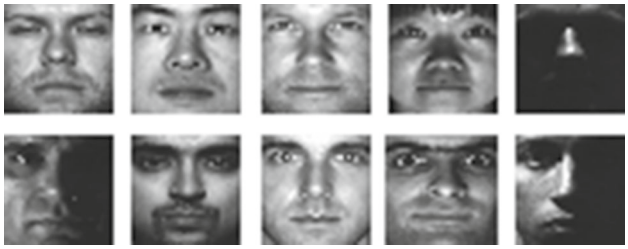


**Fig. 16** Examples of 10 classes in extended YaleB

**Table 5** Test recognition accuracy and consumed time on Extended YaleB

| Algorithm | Accuracy (%) | Consumed Time |
|---|---|---|
| K-SVD | 84.88 | 5555.89 |
| SAE | 62.60 | 2220 |
| Proposed method | 94.23 | 3097.46 |

Due to the complexity of the face data, the setting is different from the first two datasets. We set $T_0 = 16$ in the K-SVD algorithm, only maximum number of training iterations is still set to 80. For the Extended YaleB, input training samples are the vectorization of images with the dimension of $n = 1024$. Similarly, we employ $\mathbf{Y}_i = [\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,N}] \in$


**Fig. 15** The confusion matrix repots the results on the Coil-20 database

$\mathbf{R}^{1024\times 51}$ to represent the set of training samples of the $i$-th class, where $i \in \{1, \ldots, 38\}$. And it should be emphasized that each class dictionary is learned with $K = 31$ atoms to use the K-SVD algorithm. There are still two sequential stages during our training procedure like the first two datasets.

Moreover, judge from the previous experiments and parameter analysis, when $C_1 = 10, C_2 = 10^5, C_3 = 10^9$, the new method used in the Extended YaleB perform the best.

It is worth noting that the best recognition performance of SAE is only 62.60 % in comparison with 94.23 % of the proposed method.

## 6 Conclusion

In this work we have conducted extensive experiments on the USPS dataset, Coil-20 dataset and the Extended YaleB.

The point that we want to make in this work is that using the representation learned by the denoising deep ELM-AE can achieve the denoising dictionary. Then, we have carried out several experiments to explore the recognition performance with different parameters. When combining our denoising representation with the K-SVD algorithm, we have shown more importantly that these elements such as ridge parameter $C$ can be as significant as our proposed method itself.

There are many classical and distinguished method about denoising such as SAE, even so, compared with more complex algorithms, which may have greater representational power simple, fast algorithms like our denoising deep ELM-AE can be highly competitive.

**Acknowledgements** This work was supported in part by the National Key Project for Basic Research of China under Grant 2013CB329403, the National Natural Science Foundation of China under Grant 61327809, the National High-Tech Research and Development Plan under Grant 2015AA042306, the Natural Science Foundation of Shanxi Province under Grant 2014011018-4, the Shanxi Scholarship Council of China under Grant 2013-033, and the Shanxi Scholarship Council of China under Grant 2015-045

## References

1. Aharon M, Elad M, Bruckstein A (2005) K-SVD: design of dictionaries for sparse representations. Proc Spars 5:9–12
2. Aharon M, Elad M, Bruckstein A (2006) K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans Image Process 54(11):4311–4322
3. Elad M, Aharon M (2006) Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans Image Process 15(12):3736–3745
4. Mairal J, Elad M, Sapiro G (2008) Sparse representation for color image restoration. IEEE Trans Image Process 17(1):53–69
5. Christopher P, Chopra S, Cun YL (2006) Efficient learning of sparse representations with an energy-based model. In: Advances in neural information processing systems (NIPS), pp 1137–1144
6. Yang J, Yu K, Gong Y, Huang T (2009) Linear spatial pyramid matching using sparse coding for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
7. Wright J, Yang A, Ganesh A, Sastry S, Ma Y (2009) Robust Face Recognition via Sparse Representation. IEEE Trans Pattern Anal Mach Intell 31(2):210–227
8. Wang J, Su G, Xiong Y, Chen J, Shang Y, Liu J, Ren X (2013) Sparse representation for face recognition based on constraint sampling and face alignment. Tsinghua Sci Technol 1:62–67
9. Zheng Y, Sheng H, Zhang B, Zhang J, Xiong Z (2015) Weight-based sparse coding for multi-shot person re-identification. Sci China Inf Sci 58(10):1–15
10. Cheng H, Liu Z, Yang L, Chen X (2013) Sparse representation and learning in visual recognition: theory and applications. Sig Process 93(6):1408–1425
11. Bengio Y, LeCun Y (2007) Scaling learning algorithms towards AI. In: Bottou L, Chapelle O, DeCoste D, Weston J (eds) Large-scale kernel machines, vol 34. MIT Press, pp 321–359
12. Bengio Y (2009) Learning deep architectures for AI. Found Trends Mach Learn 2:1–55
13. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. Mach Learn Res 11:3371–3408
14. Hinton GE, Osindero S (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554
15. Hinton Geoffrey E, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
16. Scott S, Matwin S (1999) Feature engineering for text classification. Int Conf Icml:379-388
17. Dong J, Karianakis N, Davis D, Hernandez J, Balzer J, Soatto S (2015) Multi-view feature engineering and learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3251–3260
18. Salakhutdinov R, Larochelle H (2010) Efficient learning of deep Boltzmann machines. Mach Learn Res 9:693–700
19. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
20. Yang Y, Wu J (2016) Mutilayer extreme learning machine with subnetwork nodes for representation learning. IEEE Trans Cybern (in press)
21. Tang J, Deng C, Huang GB (2016) Extreme learning machine for multilayer perceptron, IEEE Trans Neural Netw Learn Syst (in press)
22. Cao J, Lin Z (2015) Extreme learning machine on high dimensional and large data applications: a survey. Math Probl Eng:1–12
23. Cao J, Lin Z, Huang G-B, Liu N (2012) Voting based extreme learning machine. Inf Sci 185(1):66–77
24. Shojaeilangari S, Yau WY, Nandakumar K, Li J, Teoh EK (2015) Robust Representation and Recognition of Facial Emotions Using Extreme Sparse Learning. IEEE Trans Image Process 24(7):2140–2152
25. Sun Z, Yu Y (2015) Sparse coding extreme learning machine for classification. In: Extreme Learning Machine
26. Peng Y, Lu BL (2015) Discriminative extreme learning machine with supervised sparsity preserving for image classification. In: Extreme Learning Machine
27. Bai Z, Huang GB, Wang D (2015) Sparse extreme learning machine for regression. In: Extreme Learning Machine
28. He B, Xu D, Nian R, van Heeswijk M, Yu Q, Miche Y, Lendasse A (2013) Fast face recognition via sparse coding and extreme learning machine. Cogn Comput 6(2):264–277
29. Huang G, Zhu Q, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. Neural Netw 2:985–990

30. Rumelhart David E, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
31. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern 42(2):513–529
32. Tang J, Deng C, Huang GB (2015) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst:1–13
33. Cambria E, Huang GB (2013) Extreme learning machines. IEEE Trans Syst 28(6):30–59
34. Widrow B, Greenblatt A, Kim Y, Park D (2013) The No-Prop algorithm: a new learning algorithm for multilayer neural networks. Neural Netw 37:182–188
35. Johnson W, Lindenstrauss J (1984) Extensions of Lipschitz mappings into a Hilbert space. Modern Anal Probab 26:189–206
36. Pavone M, Coello CAC (2012) Optimization on complex systems. Memetic Computing 4(3):163–164
37. RubinsteinR, Bruckstein AM, Elad M (2010) Dictionaries for sparse representation modeling. Proc IEEE 98(6):1045–1057
38. Wright J, Ma Y, Sapiro G, Huang TS, Yan S (2010) Sparse representation for computer vision and pattern recognition. Proc IEEE 98(6):1031–1044
39. Elad M, Figueiredo Mario AT, Ma Y (2010) On the role of sparse and redundant representations in image processing. Proc IEEE 98(6):972–982
40. Bruckstein AM, Donoho DL, Elad M (2010) From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Rev 51(1):34–81
41. Engan K, Aase SO, Husoy Hskon J (1999) Method of optimal directions for frame design. IEEE Trans Signal Process 5:2443–2446
42. Mallat SG, Zhang Z (1993) Matching pursuits with time-frequency dictionaries. IEEE Trans Signal Process 41(12):3397–3415
43. Chen SS, Donoho DL, Saunders MA (2001) Atomic decomposition by basis pursuit. SIAM Rev 43(1):129–159
44. Tropp JA, Gilbert AC (2007) Signal recovery from random measurements via orthogonal matching pursuit. IEEE Trans Inf Theory 53(12):4655–4666
45. Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550–554
46. Ngiam J, Koh PW, Chen Z, Bhaskar S, Ng AY (2011) Sparse filtering. Adv Neural Inf Process Syst 2:1125–1133