Neural Processing Letters (2020) 51:129–146
https://doi.org/10.1007/s11063-019-10079-9

ELMAENet: A Simple, Effective and Fast Deep Architecture for Image Classification

Peiju Chang[1] · Jiangshe Zhang[1] · Jinyan Wang[2] · Rongrong Fei[1]

Published online: 23 July 2019

Abstract
Deep learning has drawn extensive attention in machine learning because of its excellent performance, especially the convolutional neural network (CNN) architecture for image classification task. Therefore, many variant deep models based on CNN have been proposed in the past few years. However, the success of these models depends mostly on fine-tuning using backpropagation, which is a time-consuming process and suffers from troubles including slow convergence rate, local minima, intensive human intervention,etc. And these models achieve excellent performance only when their architectures are deeper enough. To overcome the above problems, we propose a simple, effective and fast deep architecture called ELMAENet, which uses extreme learning machines auto-encoder (ELM-AE) to get the filters of convolutional layer. ELMAENet incorporates the power of convolutional layer and ELM-AE (Kasun et al. in IEEE Intell Syst 28(6):31–34, 2013), which no longer need parameter tuning but still has a good performance for image classification. Experiments on several datasets have shown that the proposed ELMAENet achieves comparable or even better performance than that of the state-of-the-art models.

Keywords Image classification · Deep learning · Convolutional neural network · ELMAENet

1 Introduction

With the popularity of deep learning, deep neural networks have been proposed to solve the image classification. In particular, the convolutional neural network (CNN) is regarded as being most suitable for this problem due to its impressive performance on image classification. Then some variant deep models based on CNN have been proposed [1–28] to deal with image

✉ Jiangshe Zhang
jszhang@mail.xjtu.edu.cn

Peiju Chang
changpeiju1979@hotmail.com

1 School of Mathematics and Statistics, Xi’an Jiaotong University, Xi’an 710049, China

2 School of Mathematics and Information Science, North Minzu University, Yinchuan, China

classification. These deep models can always generate better performance than other kind of methods.

Most methods for training CNN depend on large amounts of labeled samples for supervised training. In the supervised training, CNN is trained via back-propagation, which has shown to perform well on image classification tasks with millions of training images and thousands of labels. However, one of the bottlenecks faced by deep learning approaches based on CNN models trained using the back-propagation algorithm is the requirement of large amounts of labeled training data. As is well known, it is time-consuming and expensive to get the labeled data. And the model gets larger as the amount of required labeled samples grows quickly. Compared to supervised learning, the unsupervised learning has shown better performance on image classification and has recently gained attention as a way of addressing the labeled data-hungry nature of supervised deep learning approaches. During the unsupervised learning, useful hidden features can be automatically discovered without relying on labeled samples and can be used to create representations that facilitate subsequent image classification. Unsupervised learning can learn consistent patterns from cheap and abundant unlabelled data. Due to the above advantages, unsupervised learning is considered as the future of deep learning [29] and recent results also prove that it outperforms supervised learning. Hence, researchers have tried to study methods to train a CNN using only unlabelled data [8–13]. But CNN also inherits the disadvantages of BP algorithm, such as local minima, time consuming etc. Furthermore, CNN needs huge computations and training set to tune numerous connection weights. Besides, deep CNN based models generally need deeper architectures to achieve wonderful performance for image classification and have a large number of parameters to be chosen.

In this paper, we present a new architecture called ELMAENet and an approach for unsupervised image classification that addresses the above mentioned problems associated with CNN-based supervised and unsupervised deep learning approaches. The proposed ELMAENet is inspired by the convolutional neural network PCANet [16], which Chan proposed in 2014 on par with the state of the art deep models and don't require the strategy of back-propagation. Based on PCANet framework, SRDANet [17], DLANet [18] and SPCANet [19] have been proposed. But until now, there is still not a neural network applied into this baseline. Therefore, we try to introduce a special neural network ELM-AE to the framework and build the proposed ELMAENet, in which the local features are learned using the structure of PCANet while ELM-AE is utilized to learn filters got from the vectorized and mean-removed patches. ELMAENet performs unsupervised learning to get good features without fine-tuning by back-propagation and demonstrates better performance than other methods, which can be seen from the below experiments.

Compared with previous work demonstrating the effective of deep models based on CNN for image classification, our work ELMAENet distinguishes itself in three ways. First, ELMAENet doesn't need to fine tune the network like most of deep models based on CNN. It is well known that fine tuning a network having millions of parameters trained on a large-scale data-set through back propagation is time and resource consuming. The use of ELM-AE in our framework allows eliminating back-propagation while providing good performance on image classification. On the Norb data-set, our ELMAENet-2 achieved an accuracy of 98.28%, on the Mnist data-set it obtained an accuracy of 99.46%, while on the USPS data-set it provided 99.1%. Second, ELMAENet performs its training in unsupervised learning. The downside of supervised learning is the need for expensive labels because the model gets larger with the increase of the amount of the labeled samples. But our ELMAENet doesn't require any labeled data as we introduce a novel training procedure. Third, ELMAENet own the ease use, fast speed and exceptional performance of ELM. The proposed ELMAENet uses

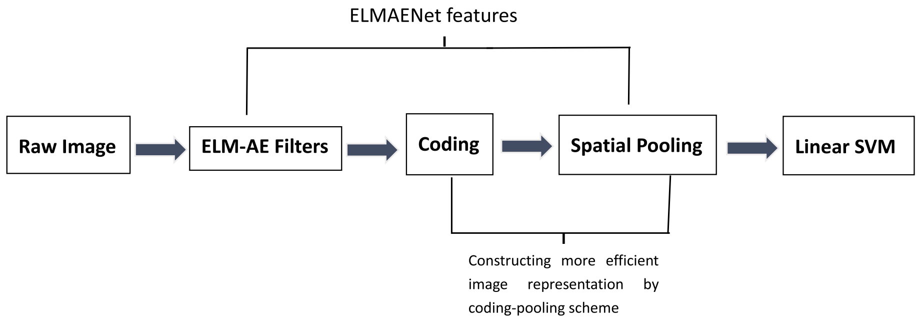

ELMAENet features
Raw Image
ELM-AE Filters
Coding
Spatial Pooling
Linear SVM
Constructing more efficient image representation by coding-pooling scheme

Fig. 1 The flow diagram of ELMAENet classification system

ELM-AE to get the filters of convolutional layer that makes the training time of ELMAENet less, which can be seen from the below experiments.

In details, ELMAENet includes three layers in each stage: convolutional filter layer, nonlinear processing layer and feature pooling layer. The detailed training process is as follows. First, ELM-AE is used to learn filter kernels and then the convolutions are performed between the original images and the learnt filters. The input of nonlinear processing layer is the output of all convolutional filter layers. Second, we binarize the above outputs using hashing method and then get the decimal values. Third, we compute the histogram of the decimal values, concatenate all the histograms into one vector and then obtain the features of images. At last, we put these features into a linear SVM for classification. The flow diagram of our classification system is illustrated in Fig. 1. The proposed model is denoted as ELMAENet-1 if there is one convolutional layer while it is denoted as ELMAENet-2 if there are two convolutional layers.

The rest of the paper is organized as follows. In Sect. 2, we review some variations of CNN and the theory of ELM-AE. Then the detailed feature learning algorithm of ELMAENet is described in Sect. 3. Section 4 shows the experimental results on USPS, Norb and Mnist. At last we discuss the difference of the proposed model ELMAENet with other models in Sect. 5.

2 Related Work

The proposed ELMAENet utilities convolution layer and pooling layer like CNN and employs ELM-AE to learn the filters. Therefore, in the following we describe some variants of CNN and the theory of ELM-AE.

2.1 Some Variant of DCNN

Since introduced by LeCun et al. [30], CNNs have demonstrated excellent performance in image classification. LeCun [31] proposed LeNet-5, which forms the original pattern of CNNs. However, the research of CNNs was very slow because of a large amount of computation until Hinton et al. [32] proposed greedy layer-wise unsupervised learning algorithm. After that, CNNs get extensive attention and obtain great improvement. In 2012, a classical DCNN called AlexNet was proposed by Krizhevsky et al. [5] and got the best result on LSVRC-2010 ImageNet, which attracts many researchers to pay more and more attention on CNNs. Therefore, many powerful variants of CNNs come out such as VGGNet [33], GoogleNet [34] and ResNet [35].

Recently supervised learning with CNNs has got more attention in image classification. Comparatively, unsupervised learning with CNNs has received less attention. In order to fill the gap between the success of CNNs for supervised learning and unsupervised learning, some researchers [10–19] try to study approaches for training a CNN using only unlabeled data. Dosovitskiy et al. [10] proposed a new training procedure and a discriminative objective for unsupervised feature learning by training a CNN without class labels, but the best results on several popular data-sets were got. Radford et al. [11] constructed a class of CNNs called deep convolutional generative adversarial networks (DCGANs), which had certain architectural constraints and were strong candidates for unsupervised learning. Doersch et al. [12] trained a CNN using spatial context as a source of free and plentiful supervisory signal resulting in state-of-the-art performance among algorithms. Huang et al. [13] learned and predicted visual attributes directly from data by introducing a simple yet powerful unsupervised approach and a novel two-stage pipeline made convincing results got, which consisted of unsupervised discriminative clustering and weakly-supervised hashing. The above unsupervised learning for CNNs all need back-propagation, which is a time-consuming process and even need some ad hoc tricks. Hence, Chan et al. [16] proposed a CNN without BP algorithm, adopting PCA or LCA to learn the filters, and the experimental results showed that two-layer PCANet is better than the state-of-the-art methods for some image classification. Then variants of PCANet came out such as SRDANet [17], DLANet [18], SPCANet [19], and our proposed model ELMAENet is also in this frame.

Besides the above variants of CNNs, some researchers [20–28] attempt to construct deep models based on ELM in order to improve the learning speed of CNNs. Huang et al. [20] studied the general architecture of locally connected ELM and proposed local receptive fields based ELM (ELM-LRF), in which random convolutional nodes and a pooling structure had been implemented. Bai et al. [27] used ELM-LRF as a framework for object recognition, which is on par with the best one on ETH-80 and achieves the new records for NORB and COIL. Huang et al. [26] proposed a method called extreme learning machine with multi-scale local receptive fields (ELM-MSLRF) to get feature learning and achieved the better performance on the NORB data-set. CaO et al. [21] put forward a novel hybrid deep learning model CNN-ELM in which CNN was in charge of feature extraction and ELM performed as a classifier. McDonnell et al. [28] combined RF-C-ELM with RF-CIW-ELM in a two-layer ELM to form RF-CIW-C-ELM whose performance is close to state-of-the-art results for MNIST and NORB. Wang et al. [25] proposed a rapid 3D feature learning method CAE-ELM combining the advantages of the CNN, auto-encoder and ELM, which performed better and faster than other methods. Zhu et al. [22] employed ELM-AE as the learning unit to study local receptive fields to construct H-ELM, which had much faster learning speed and achieved state-of-the-art performances. Pang et al. [24] combined the power of CNN and fast training of ELM to construct a rapid learning method, namely, deep convolutional extreme learning machine (DC-ELM), which achieved better testing accuracy on MNIST and USPS with significantly shorter training time compared with deep learning methods and other ELM methods.

2.2 ELM-AE

Guang-Bin Huang with his colleagues introduces ELM and proves the following theory of the universal approximation capability of ELM in [36].

Theorem 2.1 *Let $e_n \equiv f - f_n$ denote the residual error function for the current network f_n with n hidden nodes, where $f \in L^2(X)$ is the target function. Given any bounded noncon-*

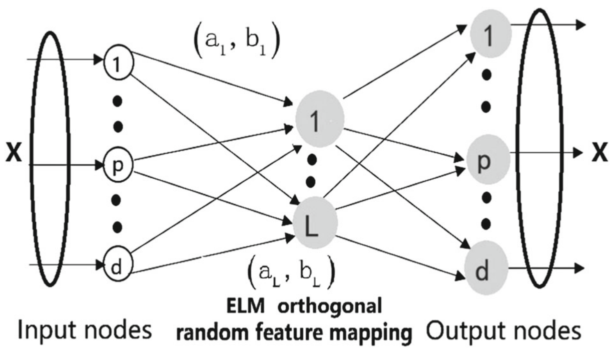


Fig. 2 The structure of ELM-AE

stant piecewise continuous function $g : R \rightarrow R$ *for additive nodes or integrable piecewise continuous function* $g : R \rightarrow R$ *and* $\int_R g(x)\,dx \neq 0$ *for RBF nodes, for any continuous target function f and any randomly generated function sequence* g_n, $\lim_{n\rightarrow\infty} \|f - f_n\| = 0$ *holds with probability one if*

$$\beta_n = \frac{< e_{n-1}, g_n >}{\|g_n\|^2}$$

It is because of the above theory that ELM algorithm can generally be efficiently used in many applications. Thus, many researchers have payed more attention on ELM because of its fast learning speed and good generalization capability. Kasun [37] *proposed Extreme Learning Machine Auto-Encoder(ELM-AE). The network structure of ELM-AE is shown in* Fig. 2.

Given N samples $\{x_i, x_i \in R^d, i = 1, 2, \ldots, N\}$, the number of hidden nodes in ELM-AE is L and the activation function is $\{G(a, b, x), x \in R^d, a \in R^L, b \in R^L\}$, where a is is input weights and b is the biases of hidden nodes. ELM-AE adopts an unsupervised learning method as follows: input data is used as the output data, random input weights and random biases of the hidden nodes are chosen to be orthogonal. Features of the input data of ELM-AE can be represented in three different architectures:

1. compressed architecture: $d > L$

In this case, ELM-AE represents features from a higher dimensional input data space to a lower dimensional feature space.

2. sparse architecture: $d < L$

In this case, ELM-AE represents features from a lower dimensional input data space to a higher dimensional feature space.

3. equal dimension architecture: $d = L$

In this case, ELM-AE represents features from an input data space dimension equal to feature space dimension.

Next we will explain the training methods of ELM-AE in detail.

For compressed ELM-AE architecture, the orthogonal random weights and biases of hidden nodes project a input data x_i to a lower dimension space, as shown by Johnson-Lindenstrauss Lemma [14] and calculated by the following Equation:

$$h_i = G(ax_i + b) \in \Re^{L\times 1} \qquad a^T a = I, b^T b = I$$

For sparse ELM-AE architecture, the orthogonal hidden random parameters are calculated by the following Equation:

$$h_i = G(ax_i + b) \in \Re^{1 \times L} \qquad aa^T = I, bb^T = I$$

where $a = [a_1, \ldots, a_L]^T$ is the orthogonal random weight and $b = [b_1, \ldots, b_L]^T$ is the orthogonal random bias between the input nodes and the hidden nodes.

If the number of training samples is less than L, output weights of ELM-AE β are calculated by the below Equation:

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

If the number of training samples is larger than L, output weights of ELM-AE β are calculated by the below Equation:

$$\beta = \left(\frac{I}{C} + HH^T\right)^{-1} HX^T$$

where $X = [x_1, x_2, \ldots, x_N] \in \Re^{d \times N}$, $H = (h_1, h_2, \ldots, h_N) \in \Re^{L \times N}$.

3 Feature Learning of ELMAENet

In this section, we introduce feature learning of a simple, effective and fast deep architecture ELMAENet. ELMAENet deploys the structure of PCANet to learn the local feature but explore filters by ELM-AE. In the following, the feature learning of two-stage ELMAENet, ELMAENet-2, is described in details.

Suppose that we have N input training images $\{X_i\}_{i=1}^N$ of size $m \times n$, the patch size is $k_1 \times k_2$ for each input image and the number of filters in two stages is L_1, L_2 respectively.

3.1 The First Convolutional ELM-AE Filter Layer

First, we take image blocks for each image and then each blocks will be vectorized.

Specifically, for ith image we take a $k_1 \times k_2$ patch around each pixel, collect all patches of the image and construct data matrix $P_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,mn}) \in \Re^{k_1 k_2 \times mn}$, where $p_{i,j}$ is the jth vectorized block in X_i. For normalization, we subtract patch mean from each patch and obtain the normalized data matrix: $\overline{P_i} = (\overline{P_{i,1}}, \overline{P_{i,2}}, \ldots, \overline{P_{i,mn}})$, where $\overline{P_{i,j}}$ is a mean-removed patch. By constructing the same matrix for all training images and putting them together, we get $P = [\overline{P_1}, \overline{P_2}, \ldots, \overline{P_N}] \in \Re^{k_1 k_2 \times Nmn}$.

Second, we use ELM-AE to explore filters for all training images.

The number of hidden nodes of ELM-AE is set to be the number of filters L_1. The orthogonal random weights and biases of hidden nodes are calculated by the following Equation:

$$h_i = g(a\overline{P_i} + b) \quad a^T a = I, b^T b = I$$

Output weights of ELM-AE β are calculated by the below Equation:

$$\beta = \left(\frac{I}{C} + HH^T\right)^{-1} HP^T \in \Re^{L_1 \times k_1 k_2}$$

where $H = (h_1, h_2, \ldots, h_N) \in \Re^{L_1 \times Nmn}$.

Then we can obtain the transposition of β, which is the weight between the input layers and the hidden layers. It can be denoted by V^1.

Therefore, the ELM-AE filters can be expressed as $W_{l_1}^1 = mat_{k_1,k_2}(v_{l_1}^1)\ \ l_1 = 1, 2, \ldots, L_1$ where $mat_{k_1,k_2}(v)$ is a function that maps $v \in \Re^{k_1 k_2}$ to a matrix $W \in \Re^{k_1 \times k_2}$ and $v_{l_1}^1$ denotes the l_1th row of V^1.

At last, we perform the 2D convolution between X_i and $W_{l_1}^1$, then get the l_1th feature map of the first stage:

$$X_i^{l_1} = X_i * W_{l_1}^1 \quad i = 1, 2, \ldots, N; l_1 = 1, 2, \ldots, L_1$$

3.2 The Second Convolutional ELM-AE Filter Layer

The process of the second convolutional ELM-AE filter layer is the same as the first one.

First, we can collect all the overlapping patches of $X_i^{l_1}$, subtract patch mean from each patch and form

$$\overline{Q_i^{l_1}} = \left(\overline{q_{i,1}^{l_1}}, \overline{q_{i,2}^{l_1}}, \ldots, \overline{q_{i,mn}^{l_1}}\right) \in \Re^{k_1 k_2 \times mn}$$

where $\overline{q_{i,j}^{l_1}}$ is the jth mean-removed patch in $X_i^{l_1}$.

Then, all mean-removed patches of $X_i^{l_1}$ are further collected and we define it as follows:

$$Q^{l_1} = \left[\overline{Q_1^{l_1}}, \overline{Q_2^{l_1}}, \ldots, \overline{Q_N^{l_1}}\right] \in \Re^{k_1 k_2 \times Nmn}$$

Then we have the following matrix for all the filters: $Q = [Q^1, Q^2, \ldots, Q^{L_1}] \in \Re^{k_1 k_2 \times L_1 Nmn}$.

Second, we can get the ELM-AE filters of the second stage as follows:

$$W_{l_2}^2 = mat_{k_1,k_2}\left(v_{l_2}^2\right) \quad l_2 = 1, 2, \ldots, L_2$$

where $v_{l_2}^2$ denotes the l_2th row of V^2, V^2 is the orthogonalization of the output weight that we obtain from ELM-AE by Q.

At last, we convolves $X_i^{l_1}$ with $W_{l_2}^2$ and have L_2 outputs $O_i^{l_2} = X_i^{l_1} * W_{l_2}^2 \quad l_2 = 1, 2, \ldots, L_2$.

3.3 The Nonlinear Processing Layer

$O_i^{l_2}$ is the input of the nonlinear processing layer. For each $X_i^{l_1}$, there are L_2 outputs $O_i^{l_2}$ in the second convolutional layer. We binarize these outputs and get the following results:

$$G(X_i^{l_1} * W_{l_2}^2) \quad l_2 = 1, 2, \ldots, L_2, \quad where \quad G(x) = \begin{cases} 1 \ if\, x \geq 0 \\ 0 \ if\, x < 0 \end{cases}$$

We consider the vector of L_2 binary bits as a decimal number and convert the L_2 outputs in $O_i^{l_2}$ back into a single integer-valued image:

$$T_i^{l_1} = \sum_{l_2=1}^{L_2} 2^{l_2 - 1} G\left(X_i^{l_1} * W_{l_2}^2\right)$$

whose pixel is a decimal number in the range $[0, 2^{L_2} - 1]$.

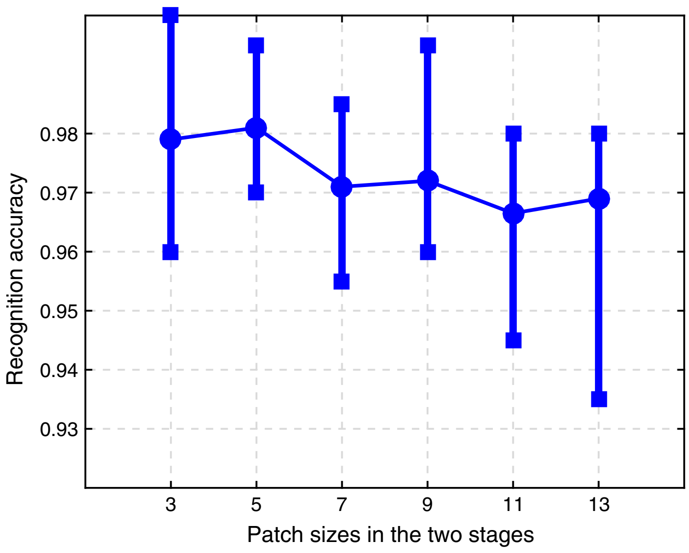


Fig. 3 Recognition accuracy of ELMAENet-2 on ORL for varying the patch size

3.4 The Feature Pooling Layer

For each of the L_1 images $T_i^{l_1}, l_1 = 1, 2, \ldots, L_1$, we partition it into B blocks (each block size is $h \times h$) and compute the histogram of the decimal values in each block. Then we concatenate all the B histograms into one vector and denote it as $Bhist(T_i^{l_1})$. After this encoding process, the feature of the input image X_i can be defined as follows:

$$f_i = [Bhist(T_i^1), Bhist(T_i^2), \ldots, Bhist(T_i^{L_1})]^T \in \Re^{2^{L_2} L_1 B}$$

```
**Algorithm 1 (Feature learning Algorithm of ELMAENet-2)**
**Input:** The training images: $\{X_i\}_{i=1}^N$ of size $m \times n$;
The patch size in two stages: $k_1 \times k_2$;
The number of filters in two stages: $L_1, L_2$
**Output:** the feature of the input image $X_i : f_i$;
**Step 1**: preprocess the image $X_i$ by subtracting patch mean from each patch denoted as $\overline{P_i}$ and obtain $P$ .
**Step 2**: obtain the $l_1 th$ ELM-AE filter $W_{l_1}^1$ using ELM-AE for P.
**Step 3**: get the $l_1 th$ filter output in the first stage and denote it as $X_i^{l_1}$.
**Step 4**: apply Step 1-3 for $X_i^{l_1}$ and obtain the $l_2 th$ filter output $O_i^{l_2}$ in the second stage.
**Step 5**: convert the $L_2$ outputs in $O_i^{l_2}$ back into a "image":
    $$T_i^{l_1} = \sum_{l_2=1}^{L_2} 2^{l_2-1} G(O_i^{l_2})$$
**Step 6**: obtain the feature of the input image $X_i$ using histogram for the $L_1$ images:
    $$f_i = \left[Bhist(T_i^1), Bhist(T_i^2), \ldots, Bhist(T_i^{L_1})\right]^T \in \Re^{2^{L_2} L_1 B}$$
```

4 Experiments

In this section, we use the proposed ELMAENet to obtain features of images and then apply them to linear SVM [38] for image classification. We evaluate the performance of the proposed ELMAENet on ORL, CMU-PIE, USPS [39], Norb [40] and MNIST [31]. In the experiments, the patch sizes are set to $k_1 = k_2 = k$. All the experiments have been made in MATLAB R2014a environment running on a PC with 3.6 GHz CPU with Intel(R) Core (TM)i7-4790 and 16 GB RAM.

4.1 2D Face Recognition on ORL Data Set

The ORL data set is composed of 400 images of size 112×92. There are 40 persons, 10 images per each person. The images were taken at different times, lighting and facial expressions. The faces are in an upright position in frontal view, with a slight left-right rotation. We down-sample the image to 32×32 pixels.

In order to investigate the impact of the patch size, we perform the experiment of ELMAENet-2 on ORL. Five images per individual are randomly chosen as training set and the rest forms the testing set. The parameters of ELMAENet-2 are set as follows: the patch size $k = 3, 5, 7, 9, 11, 13$ separately, the number of filter kernels $L_1 = L_2 = 8$ and non-overlapping block size is 7×7. We average the results over 10 random splits and the results are shown in Fig. 3. It can be seen that ELMAENet-2 achieves the best result and has the least standard deviation when the patch size is $k = 5$.

In the second experiment, we investigate the impact of the number of filter kernels of one-stage ELMAENet and two-stage ELMAENet. We randomly choose n=5 images per individual as training set and the rest of the database forms the testing set. We average the results over 50 random splits. For one-stage ELMAENet, we vary the number of filter kernels in the first stage L_1 from 4 to 14. For two-stage ELMAENet, We set the number of filter kernels in the second stage $L_2 = 8$ and vary L_1 from 4 to 14. The patch size of ELMAENet is set to $k = 7$ and the non-overlapping block size is 7×7. The results are shown in Fig. 4 and one can see that the general trend of the accuracy of one-stage and two-stage ELMAENet is to increase as L_1 rises. What's more, the average accuracies of two-stage ELMAENet is obviously much higher and more stable than these of one-stage ELMAENet.

In the following experiment, we examine the impact of the block size on histogram computation by using ORL database. The patch size is $k = 5$, the number of filter kernels $L_1 = L_2 = 8$ and non-overlapping block size on histogram computation is $h = 3, 5, 7, 9, 11, 13$ seperately. We average the results over 10 random splits and the results are shown in Fig. 5. We can see that block size $h = 5$ achieves the best performance.

Finally, we evaluate the performance of ELMAENet-2 on different number of training set. $n (= 2, 3, 4, 5)$ images per individual are chosen as training set and the rest of the database as testing set. In order to compare the performance of ELMAENet-2 with other models, we use the same parameter settings as [41]. All the results are averaged over 50 random splits. We compare our results with the other state of art methods and are shown in Table 1. From Table 1, one can see that the proposed ELMAENet achieves the best accuracy.

4.2 2D Face Recognition on CMU-PIE Data Set

For the CMU-PIE dataset, there are 41,368 pieces of pictures, captured under different lighting, poses and expressions. The CMU-PIE dataset includes 68 individuals totally and each

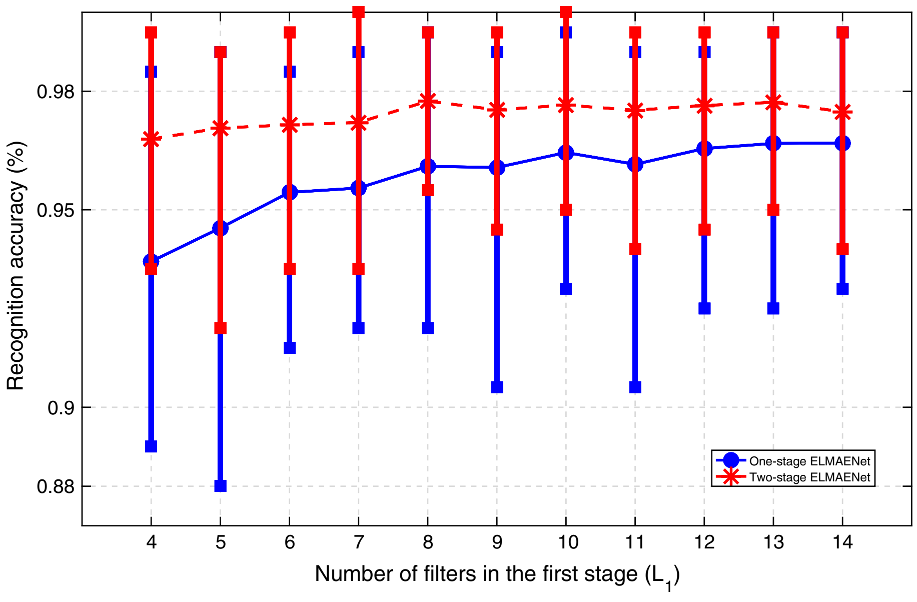


Fig. 4 Recognition accuracy of ELMAENet on ORL for varying number of kernels in the first layer L_1

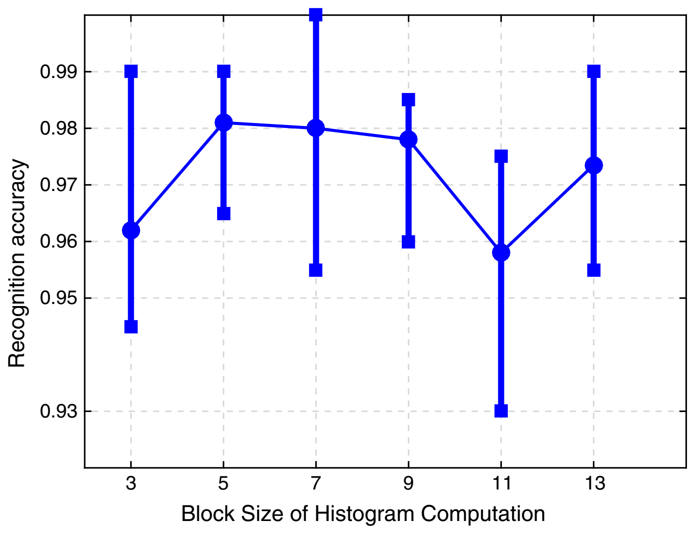


Fig. 5 Recognition rate of ELMAENet on ORL for different histogram-block size

person has 43 different illumination conditions with 13 different poses. We choose two types of them to finish our experiment: five near frontal poses and all different illuminations, including 11,554 images in total. Each individual contains approximately 170 images.

The parameters of ELMAENet-2 are set as follows: the patch size $k = 3$, the number of filter kernels $L_1 = L_2 = 8$ and non-overlapping block size is 5×5. We randomly choose n(= 5, 10, 20, 30) images per individual as training set and the rest to form the testing set. For each given n, we average the results over 10 random splits and the results are shown in

Table 1 Comparison of face recognition rates of various methods on ORL datasets (mean ± std(%))

Method	2 Train	3 Train	4 Train	5 Train
PCA	71.06 ± 3.58	79.32 ± 3.78	83.67 ± 3.02	87.80 ± 3.64
S-LDA [42]	81.71 ± 2.5	88.67 ± 2.46	92.58 ± 1.74	94.98 ± 1.63
SRDA [43]	80.93 ± 3.1	88.61 ± 2.32	92.08 ± 1.94	94.34 ± 1.52
SRC+PCA [44]	78.32 ± 2.45	86.46 ± 2.16	90.84 ± 1.71	93.54 ± 1.64
C-LBP [45]	83.1 ± 2.63	89.21 ± 1.72	94.21 ± 1.63	95.25 ± 1.43
DFD (S = 5) [46]	75.76 ± 2.16	82.57 ± 2.47	88.70 ± 1.82	92.35 ± 1.34
LDANet [15]	83.04 ± 5.2	89.92 ± 5.48	95.21 ± 2.14	97.04 ± 1.87
SRDANet [41]	84.94 ± 2.6	92.47 ± 2.46	96.42 ± 1.47	97.68 ± 1.24
ELMAENet	86.80 ± 2.27	93.6 ± 2	96.42 ± 1.27	97.75 ± 1.09

Table 2 Comparison of face recognition rates of various methods on CMU-PIE datasets (mean ± std(%))

Method	5 Train	10 Train	20 Train	30 Train
PCA	58.04 ± 1.3	74.6 ± 1.21	85.51 ± 0.53	89.35 ± 0.33
S-LDA [42]	74.85 ± 1.1	86.52 ± 0.64	92.6 ± 0.32	94.55 ± 0.34
SRDA [43]	75.56 ± 0.7	87.18 ± 0.73	92.61 ± 0.54	94.34 ± 0.22
SRC+PCA [44]	68.96 ± 0.72	77.24 ± 0.74	85.46 ± 0.71	91.98 ± 0.64
C-LBP [45]	70.79 ± 1.12	86.85 ± 0.34	94.65 ± 0.33	96.98 ± 0.23
LDANet [15]	76.95 ± 4.43	89.39 ± 2.93	96.89 ± 0.74	97.99 ± 0.43
SRDANet [41]	77.99 ± 2.6	91.42 ± 1.03	96.96 ± 0.44	98.29 ± 0.34
ELMAENet	82.70 ± 1.57	93.38 ± 1.27	97.86 ± 0.43	98.84 ± 0.27

Table 3 Performance on USPS (with 7000 training samples)

Methods	Training time(s)	Training error (%)		Testing error (%)	
		Mean	Std	Mean	Std
ELM-LRF	337.5	1.51	0.0045	3.4	0.0024
DC-ELM	204.6	2.17	0.0034	3.07	0.0028
CNN	2480.3	2.6	0.0023	3.87	0.0021
DBN	2536.6	0.89	0.0057	3.01	0.0038
ELMAENet-2	193.32	0	0	1.04	0.0013

Table 2. It can be seen that ELMAENet achieves the best performance for all test sets. It is also observed that the standard deviation of ELMAENet is less than the standard deviation of LDANet and SRDANet.

4.3 Handwritten Recognition on USPS Data Set

The USPS data set consists of 11000 samples of handwriting digits 0-9 with image size of 16×16 which is collected from different writers.

Table 4 Performance on USPS (with 10,000 training samples)

Methods	Training time (s)	Training error (%)		Testing error (%)	
		Mean	Std	Mean	Std
ELM-LRF	537.5	0.89	0.0036	1.98	0.0016
DC-ELM	320.6	1.21	0.0026	1.56	0.0018
CNN	3490.3	2.5	0.0016	2.86	0.0029
DBN	3526.51	1.12	0.0068	2.4	0.0065
ELMAENet-2	277.43	0	0	0.90	0.0027

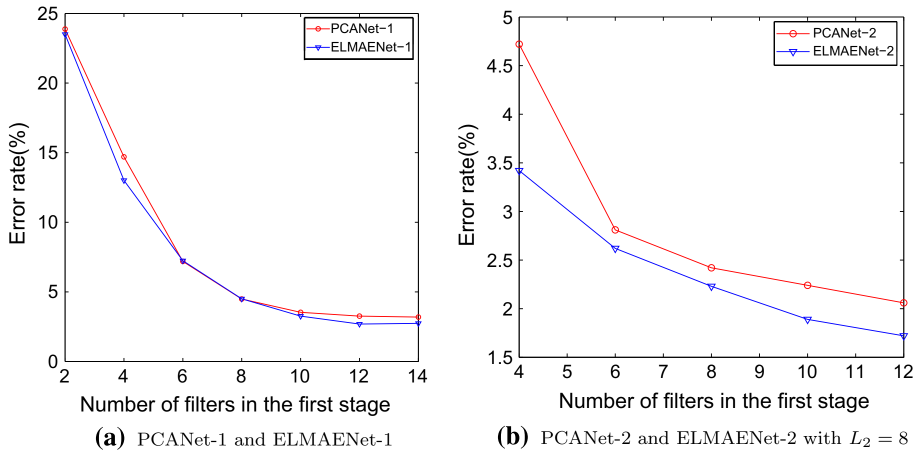


Fig. 6 Error rate of PCANet and ELMAENet on Norb testing set for different numbers of filters in the first stage

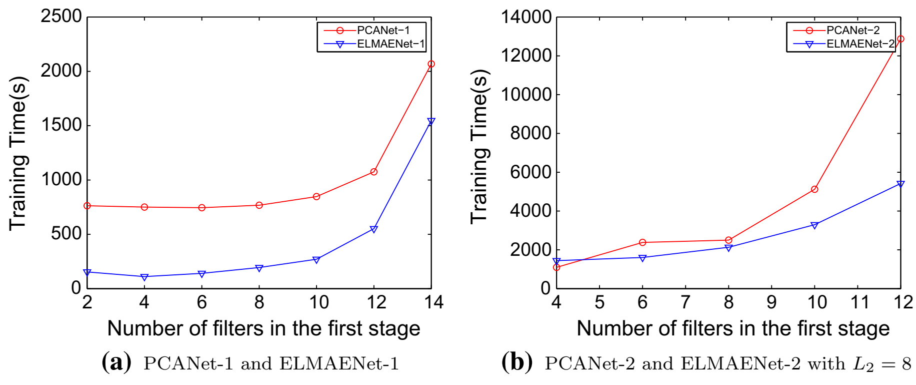
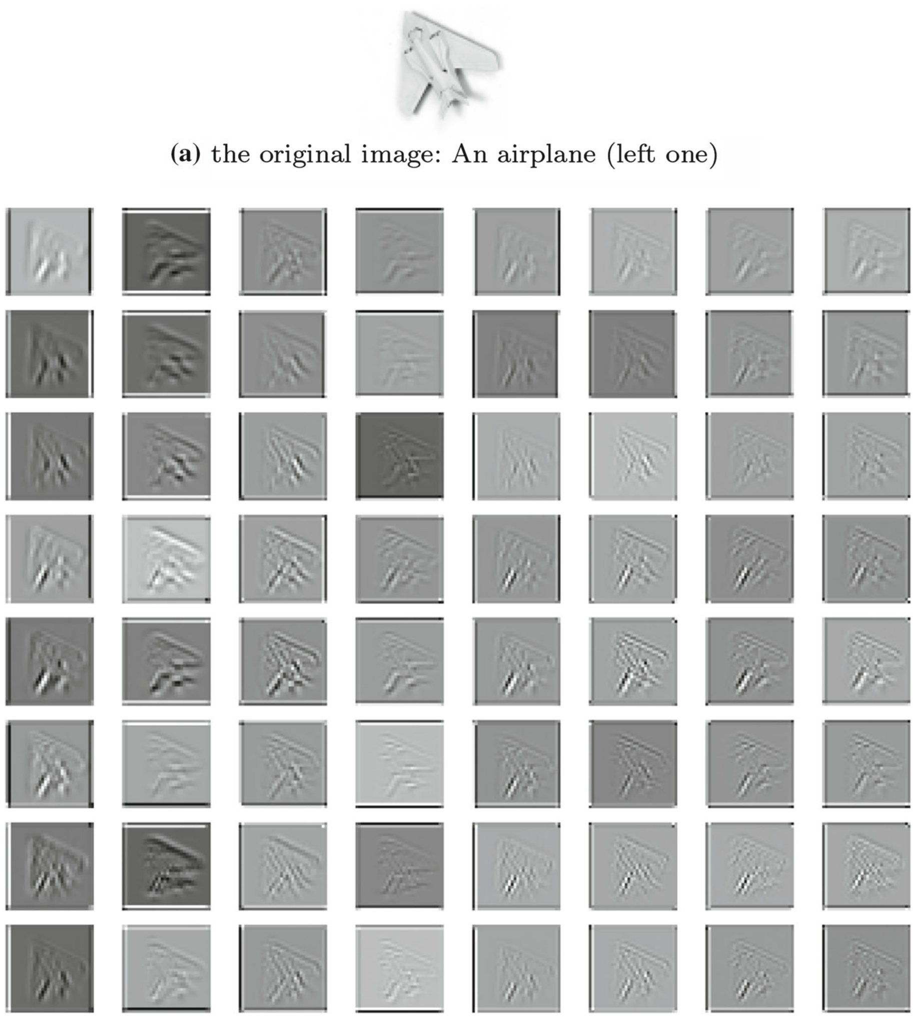


Fig. 7 Training time of PCANet and ELMAENet on Norb test set for different numbers of filter in the first stage

We perform the ELMAENet-2 on USPS and the parameter setting is as follows. The patch size of ELMAENet-2 is $k = 3$ in two stages, the number of filters is 8 and 8 separately, block size is 7×7 and the overlapping region between blocks is half of the block size. We compare ELMAENet-2 with ELM-LRF [27], DC-ELM [24], CNN and DBN. The parameter setting



(a) the original image: An airplane (left one)

(b) the 64 feature maps

Fig. 8 The comparison between the original image and the corresponding feature maps

for ELM-LRF, DC-ELM, and CNN is referred to [24]. All the methods are evaluated on 7000 and 10000 training samples separately, which are randomly selected from USPS data set. We run 30 times of all the methods separately for each case. The results are listed in Tables 3 and 4.

From Tables 3 and 4, it can be seen that ELMAENet-2 achieved the best testing accuracy on USPS data set under two cases. We can also see that ELMAENet-2 is much faster than other deep models.

4.4 Object Recognition on NORB Dataset

NORB is often used as a benchmark database by deep learning community, which contains 24,300 stereo images for training and 24,300 stereo images for testing, each belonging to 5

Table 5 Testing errors of different methods for Norb

Groupings	Methods	Error in testing (%)	Reference
Non-ELM deep models	MCRNN-MLA	7.9	[47]
	MCSRNN	7.56	[48]
	DBN	7.2	[49]
	SAE	6.5	[24]
	SDAE	5.6	[24]
	CNN	6.6	[25]
	Tied CNN	3.9	[25]
	K-means+soft activation	2.8	[25]
	GPU-based CNN	2.53	[50]
	fast-learning shallow CNN	2.2	[51]
	PCANet-2	2.06	[15]
Past deep models based-ELM	ELM-LRF	2.74	[27]
	CNN+ELM	10.35	[52]
	RF-C-ELM	6.24	[53]
	ELM-MSLRF	2.5	[26]
	ELM-SSLRF	3.5	[25]
	CAE-ELM	5.5	[25]
This paper	ELMAENet-2	1.72	This paper

generic categories with many variabilities such as 3D pose and lighting. Each sample has 2 images (left and right sides) with normalized object size and uniform background. We downsize them to 32×32 without any other preprocessing. The filter size of the networks is $k = 3$, block size is 7×7 and the overlapping region between blocks is half of the block size without special instructions in the experiments on Norb.

In order to study the effective of ELM-AE and the impact of the number of filters, we compare the testing error of PCANet and ELMAENet with different numbers of filters in the first stage. We vary the number of filters L_1 in the first stage from 2 to 14 for one-stage networks. For two-stage networks, we set $L_2 = 8$ and change L_1 from 4 to 12. The results are shown in Fig. 6. One can see that the performance of ELMAENet-1 are almost better than that of PCANet-1 except $L_1 = 6, 8$ and ELMAENet-2 outperforms PCANet-2 for all the cases. Besides, we also compare the training time of ELMAENet-1 and PCANet-1 as well as that of ELMAENet-2 and PCANet-2. From the Fig. 7, we can see that ELMAENet-1 spend less training time than PCANet-1 for all the cases and the training time of ELMAENet-2 is less than that of PCANet-2 except $L_1 = 4$.

In the second experiment, we illustrate feature maps to show the ability of extracting features of ELMAENet-2 using one data sample (airplane, the left image) for shown. From Fig. 8, we can see that the outlines of these feature maps are similar as they all are generated from the same input image (an airplane) while each map has its distinct highlighted part, so that diverse representations of the original image are obtained and good classification result can be easily achieved.

In the third experiment, we compare our model ELMAENet-2 with the other existing algorithms. In this experiment, we set $L_1 = 12$, $L_2 = 8$. From Table 5, we can see that our ELMAENet-2 achieves the best accuracy with a significant gap compared to those results reported in literature.

Table 6 Testing errors of different methods for standard Mnist

Methods	Error in testing(%)	References
ELM-SSLRF	2.41	[26]
ELM-MSLRF	1.43	[26]
CAE-ELM	1.13	[25]
CNN+ELM	0.67	[21]
ML-ELM	0.97	[37]
K-NN-SCM	0.63	[54]
Mathematical Model of CNN (best)	0.6	[55]
Local R2FP(AW)Global+R2FP	0.45	[56]
Deep ConvNet(drop connect)	0.57	[57]
Maxout Networks	0.94	[58]
Deep L2-SVM	0.87	[59]
DELM	0.81	[28]
Deep Fried Convnets	0.71	[60]
RF-C-ELM	0.57	[53]
ResNet	0.63	[61]
VGGNet	0.68	[62]
CapsuleNet	0.43	[63]
DCNNs	0.28	[64]
RandNet-2	0.63	[15]
PCANet-2 [37]	0.66	[15]
LDANet-2 [37]	0.62	[15]
ELMAENet-2	0.54	This paper

4.5 Digit Recognition on MNIST

MNIST is a widely-used benchmark for testing hierarchical representation. We use standard MNIST: 60000 for training and 10000 for testing. All the images are of size 28×28.

In the experiment, the filter size of the networks is $k = 5$, the number of filters is $L_1 = L_2 = 8$, block size is 7×7 and the overlapping region between blocks is half of the block size. We compare the testing errors of our model ELMAENet-2 with that of other models reported in literatures and the results are summarized in Table 6. From the above results, it can be seen that ELMAENet-2 produces better accuracy than some other models. The testing error of our proposed ELMAENet-2 is only 0.26% more than that of the best DCNNs until now and is better than that of the recent models, such as ResNet, VGGNet.

5 Conclusion

In this paper, we construct a simple, effective and fast unsupervised convolutional deep architecture–ELMAENet, whose training doesn't need back-propagation to learn parameters of the model and spend less time. And experimental results on USPS, Norb and Mnist show that ELMAENet is better than or close to the previous state-of-the-art models.

ELMAENet combines the power of CNN and ELM and is similar to PCANet, but it is different from the past deep architectures [16–20] and [22]. ELMAENet has the same framework with [16–19], but ELMAENet uses ELM-AE to get the filters of convolutional layer and gets better performance with less training time. Especially, DLANet in [18] is under the framework of Locality-constrained Linear Coding-Spatial Pyramid Matching (LLC-SPM) and need big memory as ELM-LRF in [20] while ELMAENet can be implemented in normal computer. ELM-LRF studies the local receptive fields from the randomness of hidden neurons, but ELMAENet uses ELM-AE to learn the filters and gets more efficient image representation so as to achieve better performance on Norb. ELMAENet doesn't need any pre-processing of those patches while these patches in HELM [22] need to be transformed by the pre-processing LCN and whitening operators.

Our proposed ELMAENet combines the power of convolutional layer and ELM-AE to construct more efficient image representation. And the training of ELMAENet doesn't need fine-tuning by backpropagation, which makes ELMAENet less time consuming. Experiments on USPS, Norb and Mnist show that our proposed model ELMAENet achieves comparable or better performance than that of the previous state-of-the-art models.

Acknowledgements This work is supported by the National Key Research and Development Program of China (No.2018YFC0809001).

References

1. Park DC (2000) Centroid neural network for unsupervised competitive learning. IEEE Trans Neural Netw 11(2):520–528
2. Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of international conference on machine learning
3. Masci J, Meier U, Cirean D, Schmidhuber J (2011) Stacked convolutional auto-encoders for hierarchical feature extraction. In: International conference on artificial neural networks
4. Yim J, Ju J, Jung H, Kim J (2015) Image classification using convolutional neural networks with multi-stage feature. In: Robot intelligence technology and applications 3. Advances in Intelligent Systems and Computing, vol 345. Springer, Cham
5. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. Proc Adv Neural Inf Process Syst 25:1090–1098
6. Norouzi M, Ranjbar M, Mori G (2009) Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In: IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009. IEEE, pp 2735–2742
7. Le QV, Ngiam J, Chen Z, Chia D, Koh PW, Ng AY (2010) Tiled convolutional neural networks. In: NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems, vol 1, pp 1279–1287
8. Cho K et al (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of conference on empirical methods in natural language processing, pp 1724–1734
9. Wang X, Gupta A (2015) Unsupervised learning of visual representations using videos. In: Proceedings of the IEEE international conference on computer vision, pp 2794–2802
10. Dosovitskiy A, Springenberg JT, Riedmiller M, Brox T (2014) Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems, vol 1, pp 766–774
11. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434
12. Doersch C, Gupta A, Efros A A (2015) Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE international conference on computer vision, pp 1422–1430
13. Huang C, Change Loy C, Tang X (2016) Unsupervised learning of discriminative attributes and visual representations. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5175–5184
14. Mairal J, Koniusz P, Harchaoui Z, Schmid C (2014) Convolutional kernel networks. In: Advances in neural information processing systems 27, pp 2627–2635

15. Perronnin F, Larlus D (2015) Fisher vectors meet neural networks: a hybrid classification architecture. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3743–3752
16. Chan T-H, Jia K, Gao S, Lu J, Zeng Z, Ma Y (2015) PCANet: a simple deep learning baseline for image classification? IEEE Trans Image Process 24(12):5017–5032
17. Tian L, Fan C, et al (2015) SRDANet: an efficient deep learning algorithm for face analysis. In: International conference on intelligent robotics and applications, pp 499–510
18. Feng Z, Jin L, Tao D, Huang S (2015) DLANet: a manifold-learning-based discriminative feature learning network for scene classification. Neurocomputing 157:11–21
19. Tianl L, Fan C (2015) Stacked PCA network (SPCANet): an effective deep learning for face recognition. In: IEEE international conference on digital signal processing, pp 1039–1043
20. Huang G-B, Bai Z, Kasun LLC, Vong CM (2015) Local receptive fields based extreme learning machine. IEEE Comput Intell Mag 10(2):18–29
21. Cao W, Gao J et al (2015) A hybrid deep learning CNN-ELM model and its application in handwritten numeral recognition. J Comput Inf Syst 11:2673–2680
22. Wentao Z, Jun M, Qing LY et al (2015) Hierarchical extreme learning machine for unsupervised representation learning. In: Neural networks (IJCNN), pp 1–8
23. Gurpnar F, Kaya H, et al (2016) Kernel ELM and CNN based facial age estimation. In: Computer vision and pattern recognition workshops, pp 785–791
24. Pang S, Yang X (2016) Deep convolutional extreme learning machine and its application in handwritten digit classification. Comput Intell Neurosci. https://doi.org/10.1155/2016/3049632
25. Wang Y, Xie Z et al (2016) An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. Neurocomputing 174:988–998
26. Huang J, Yu ZL, Cai Z et al (2017) Extreme learning machine with multi-scale local receptive fields for texture classification. Multidimens Syst Signal Process 7(28):995–1011
27. Bai Z, Kasun LLC, Huang GB (2015) Generic object recognition with local receptive fields based extreme learning machine. Proc Comput Sci 53:391–399
28. Tissera MD, McDonnell MD (2016) Deep extreme learning machines: supervised autoencoding architecture for classification. Neurocomputing 174:42–49
29. Bengio Y, Goodfellow IJ, Courville A (2015) Deep learning. Nature 521:436–444
30. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1:541–551
31. LeCun Y, Bottou L, Bengio Y, Haner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
32. Hinton GE, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313:504–507
33. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
34. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
35. Szegedy C, Ioffe S, Vanhoucke V, et al (2016) Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261
36. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental networks with random hidden computation nodes. IEEE Trans Neural Netw 17(4):879–892
37. Kasun LLC, Zhou H, Huang G-B, Vong CM (2013) Representational learning with extreme learning machines for big data. IEEE Intell Syst 28(6):31–34
38. Chang C-C, Lin C-J (2001) LIBSVM: a library for support vector machines (Online). Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
39. Zhou D, Bousquet O, Lal TN, Weston J, Scholkopf B (2004) Learning with local and global consistency. Adv Neural Inf Process Syst 16(3):21–328
40. LeCun Y, Huang FJ, Bottou L (2004) Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedinngs of international conference on computer vision pattern recognition, vol 2, pp II-97-104
41. Tian L, Fan C, Ming Y, Shi J (2015) SRDANet: an efficient deep learning algorithm for face analysis. ICIRA 8:499–510
42. Cai D, He X, Hu Y, Han J, Huang T (2007) Learning a spatially smooth subspace for face recognition. In: Proceedings of IEEE conference on computer vision and pattern recognition machine learning (CVPR 2007)
43. Cai D, He X, Han J (2008) SRDA: an efficient algorithm for large-scale discriminant analysis. IEEE Trans Knowl Data Eng 20(1):1–12

44. Wright J, Yang AY, Ganesh A, Sastry SS, Ma Y (2009) Robust face recognition via sparse representation. IEEE Trans Pattern Anal Mach Intell 31(2):210–227
45. Guo Z, Zhang D (2010) A completed modeling of local binary pattern operator for texture classification. IEEE Trans Image Process 19(6):1657–1663
46. Lei Z, Pietikainen M, Li SZ (2014) Learning discriminant face descriptor. IEEE Trans Pattern Anal Mach Intell 36(2):289–302
47. Yin Y, Gelenbe E (2016) Deep learning in multi-layer architectures of dense nuclei. arXiv preprint arXiv:1609.07160
48. Yin Y, Gelenbe E (2017) FIEEE. Single-cell based random neural network for deep learning. In: International joint conference on neural networks
49. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
50. Ciresan DC, Meier U, Masci J et al (2011) Flexible, high performance convolutional neural networks for image classification. In: IJCAI proceedings-international joint conference on artificial intelligence, vol 22(1), p 1237
51. McDonnell M D, Vladusich T (2015) Enhanced image classification with a fast-learning shallow convolutional neural network. In: International joint conference on neural networks (IJCNN). IEEE, pp 1–7
52. Keonhee L, Dong-Chul P (2015) Image classification using fast learning convolutional neural networks. Adv Sci Technol Lett 113:50–55
53. McDonnell MD, Tissera MD et al (2015) Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine' algorithm. PLoS ONE 10(8):e0134254
54. Yu K, Lin Y, Lafferty J (2011) Learning image representations from the pixel level via hierarchical sparse coding. In: CVPR
55. Nemkov RM, Mezentseva OS, Mezentsev D (2016) Using of a convolutional neural network with changing receptive fields in the tasks of image recognition. In: Proceedings of the first international scientific conference
56. Wei X, Zhang L, Dua B, Tao D (2017) Combining local and global: Rich and robust feature pooling for visual recognition. Pattern Recognit 62:225–235
57. Wan L, Zeiler M, Zhang S, LeCun Y, Fergus R (2013) Regularization of neural networks using DropConnect. In: Proceedings of the 30th international conference on machine learning, Atlanta, Georgia, USA; JMLR: WCP volume 28
58. Goodfellow IJ, Warde-Farley D, Mirza M, Courville A, Bengio Y (2013) Maxout networks. Technical Report Arxiv report 1302.4389, Universite de Montreal
59. Tang Y (2013) Deep learning using linear support vector machines. In: Workshop on challenges in representation learning, ICML
60. Yang Z, Moczulski M, Denil M, de Freitas N, Smola A, Song L, Wang Z (2015) Deep fried convnets. In: ICV
61. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778. 1, 5, 6, 7, 8
62. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 1, 5, 6, 7, 8
63. Sabour S, Frosst N, Hinton GE (2017) Dynamic routing between capsules. arXiv preprint arXiv:1710.09829, 5, 6, 7
64. Ma B, Xia Y (2018) Autonomous deep learning: a genetic DCNN designer for image classification. arXiv:1807.00284