Neural Comput & Applic (2010) 19:1069–1079
DOI 10.1007/s00521-010-0378-4

ORIGINAL ARTICLE

Reconstruction and recognition of face and digit images using autoencoders

Chun Chet Tan · C. Eswaran

Received: 19 March 2009 / Accepted: 23 April 2010 / Published online: 20 May 2010

Abstract This paper presents techniques for image reconstruction and recognition using autoencoders. Experiments are conducted to compare the performances of three types of autoencoder neural networks based on their efficiency of reconstruction and recognition. Reconstruction error and recognition rate are determined in all the three cases using the same architecture configuration and training algorithm. The results obtained with autoencoders are also compared with those obtained using principal component analysis method. Instead of whole images, image patches are used for training, and this leads to much simpler autoencoder architectures and reduced training time.

Keywords Autoencoder · Neural network · Restricted Boltzmann machine · Image patches · Reconstruction · Recognition

1 Introduction

Autoencoder networks are feed-forward neural networks that can have more than one hidden layer. These networks attempt to reconstruct the input data at the output layer. The targets at the output layer are the same as the input data, thus the size of the output layer is also the same as the size of the input layer. Autoencoders are supervised neural networks, which are trained using a gradient descent method, such as backpropagation (BP). Since the size of the hidden layer in an autoencoder is smaller than the size of the input data, the dimensionality of input data is reduced to a smaller-dimensional code space at the hidden layer. The outputs from the hidden layer are then reconstructed into the original data at the output layer. Like principal component analysis (PCA), the autoencoders can give mappings in both directions between the data and the code space.

Using more number of hidden layers in an autoencoder neural network, a high-dimensional input data can be reduced to much smaller code space. However, training a multilayer network with more than one hidden layer is tedious, and further it will not also give good results [1]. This is due to the fact that the weights at deep hidden layers are hardly optimized. The time required for training such a network is also extremely high. There exist two techniques for solving the training problem, namely, stacking and Restricted Boltzmann Machine (RBM) [1, 2].

Though the main focus of this paper is not image recognition but only image reconstruction, we do investigate the correlation between the reconstruction and the recognition of the images. For details on the applications of the image reconstruction including image compression, one can refer to [3]. In this paper, we compare the reconstruction and recognition performances of the following three types of autoencoders:

1. Stacked autoencoder in which BP is applied multiple times i.e. once for each stacking (termed as SA-MBP).
2. Stacked autoencoder using RBM (SRBM) in which BP is applied only once at the end of last stacking (termed as SRBM-SBP) (the model from [1]).
3. SRBM in which BP is applied multiple times i.e. once for each stacking (termed as SRBM-MBP).

The reconstruction performances are compared based on the mean squared error (MSE) of the reconstructed images.

C. C. Tan (✉) · C. Eswaran
Faculty of Information Technology, Multimedia University,
63100 Cyberjaya, Selangor, Malaysia
e-mail: cctan@mmu.edu.my

C. Eswaran
e-mail: eswaran@mmu.edu.my

On the other hand, the recognition performances are compared based on the recognition rate. For all the three networks, identical architecture configurations and training methods are used for comparison. We also compare the performances of the autoencoders with that of a statistical linear method, PCA using the same datasets and same number of dimensions. Finally, experiments are conducted to measure the reconstruction and recognition performances of SRBM-MBP network with much simpler architecture that uses image patches for training.

The remainder of this paper is organized as follows: In Sect. 2, the architecture of autoencoder is described and in Sect. 3, the training methods are presented. Image reconstruction using autoencoders is explained in Sect. 4. Section 5 describes the experiments conducted for image recognition. Finally, the conclusions are given in Sect. 6.

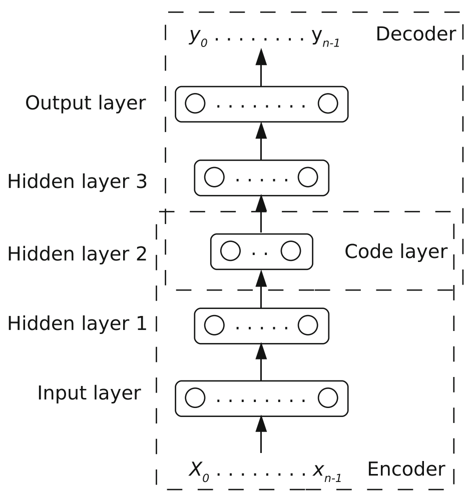


Fig. 1 Multilayer autoencoder neural network architecture

2 Architecture

The architecture of a five-layer autoencoder neural network is depicted in Fig. 1. An autoencoder is a feed-forward neural network with one or more hidden layers that attempts to reconstruct its input activities at its output. Hence, the main difference of the autoencoder and the traditional neural network is the size of the output layer. In an autoencoder, the size of the output layer is always the same as the size of the input layer. Besides that, the size of the deepest hidden layer in an autoencoder network is always smaller than the sizes of the input and output layers. As shown in Fig. 1, the autoencoder network comprises two components namely "encoder" and "decoder". The "Encoder" part of the network transforms the high-dimensional input data into a low-dimensional code and the "decoder" part (which is similar to the "encoder" network) reconstructs the original high-dimensional data from the low-dimensional code.

Sigmoid activation functions are normally used in autoencoders for nonlinear mapping. If linear activation functions are used, autoencoders will be performing similar to PCA. The networks can be trained by minimizing the mean squared error between the original and the reconstructed data. The required gradient is easily obtained using the chain rule to back propagate the error derivatives first through the decoder network and then through the encoder network.

3 Training methods

3.1 Polak Ribiére conjugate gradient backpropagation

Polak Ribiére conjugate gradient backpropagation (PR-BP) is used to train the autoencoders in our study [4]. Conjugate backpropagation is one of the variations of backpropagation. It speeds up the training process compared to the traditional backpropagation with momentum. Line search function of conjugate is used to locate the minimum point in the error function. The first search direction is the negative of the gradient of performance. In the succeeding iterations, the search direction is computed according to the formula

$$p_k = -g_k + \beta_k p_{k-1} \tag{1}$$

where g_k is the new gradient and p_{k-1} is the previous search direction. The parameter β_k can be computed in different ways. For the Polak Ribiére variation of conjugate gradient, it is computed according to

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1} g_{k-1}} \tag{2}$$

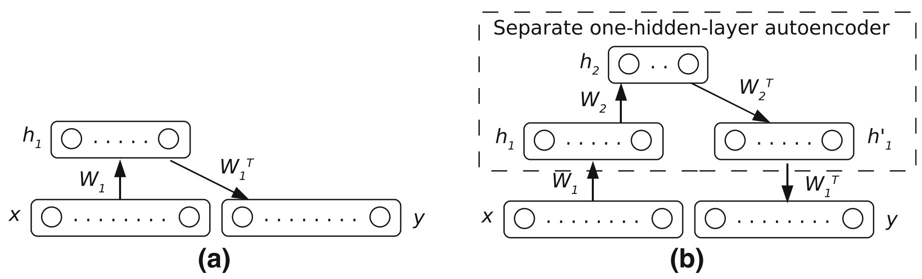
where it is a inner product of the previous change in the gradient with the current gradient divided by the norm squared of the previous gradient.

3.2 Stacked autoencoder

It is difficult to optimize the weights in autoencoders that have multiple hidden layers ($\geq$2). Autoencoders with large initial weights normally suffer from the problem of local minima. On the other hand, autoencoders with small initial weights take considerably long time to converge [1].

A method for training the deep networks is explained in [5]. In this method, the training is implemented in phases. During the first phase, the autoencoder is assumed to have three layers, namely, input layer x, output layer y and the hidden layer h_1 as shown in Fig. 2a. The weights W_1 and W'_1 are trained using backpropagation. During the second phase as depicted in Fig. 2b, a separate one-hidden-layer

Fig. 2 Stacked autoencoder



network consisting of input layer h_1, output layer h'_1 and hidden layer h_2 is stacked onto the existing autoencoder of Fig. 2a. The inputs to the input layer h_1 are x'_0, x'_1, $x'_2, \ldots, x'_{m-1}$, where $x'_j = z_j$.

Since the size and the value of the input and output layers of an autoencoder are the same, output layer h'_1 is the same as the input layer h_1. Hidden layer h_2 is a new hidden layer added onto the autoencoder. The weights of W_2 and W'_2 can be trained using backpropagation. After the training of the separate one-hidden-layer network is completed, all the weights of the autoencoder are trained together to converge to a global minima. This kind of autoencoders is called stacked autoassociators [5]. Figure 2 illustrates the iterative training procedure. With this method, a deep autoencoder with more than one single hidden layer can be trained to converge.

3.3 Restricted Boltzmann machine (RBM)

An alternative method for training the multilayer autoencoders was proposed by Hinton et al. [1]. All the weights of the multilayer autoencoder can be trained in a single step provided the weights are pre-trained with RBM. Since the weights are close to the good solution after RBM pre-training, backpropagation works well for fine-tuning the weights in the multilayer autoencoders.

For an image input, the RBM can be modeled as a two-layer network in which the energy function of pixels connected to feature detectors is given by

$$E(v,h) = -\sum_{i\in\text{pixels}} v_i b_i - \sum_{j\in\text{features}} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{3}$$

where v_i and h_j are the binary states of pixel i (visible unit) and feature j (hidden unit), b_i and b_j are the biases, respectively, and w_{ij} is the weight between the visible(input) unit i and the invisible(hidden) unit j. Given a training image, h_j of each feature j is set to 1 with probability $p(b_j + \sum_i v_i w_{ij})$ where $p(x)$ is a logistic function. The network eventually will reach a Boltzmann distribution in which the probability of the equilibrium state, v, is determined solely by the "energy" of that state vector relative to the energy of all possible binary state vectors as shown below [6]

$$P(v) = \frac{exp(-E(v,h))}{\sum exp(-E(v',h'))} \tag{4}$$

By differentiating (4) and using the fact that

$$\frac{\partial E(v)}{\partial w_{ij}} = -v_i h_j, \tag{5}$$

The change in weight can be derived as

$$\begin{aligned} \Delta w_{ij} &= \epsilon \langle \partial log P(v) / \partial w_{ij} \rangle \\ &= \epsilon (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}) \end{aligned} \tag{6}$$

where ε is the learning rate, $\langle . \rangle_{\text{data}}$ is an expected value in the data distribution and $\langle . \rangle_{\text{model}}$ is an expected value of Boltzmann sampling state vectors from its equilibrium distribution at temperature 1 [7, 8]. The temperature will be increased as the learning converges. The same learning rule is applied to the biases, b_i and b_j.

The hidden units of the Boltzmann possess properties like latent variables that allow the RBM to model the distributions over visible vectors that cannot otherwise be modeled by direct pairwise interactions between the visible units. This enables the RBM to learn higher-order structure in the data.

As reported in [8], an alternative way of using RBM is to treat the learning of RBM as a pre-training that finds a good region of the parameter space. After pre-training, the RBM is considered to be "unfolded" into an autoencoder network in which the stochastic activities of the binary hidden features are replaced with deterministic probabilities. Figure 3 shows the pre-training of a stack of RBMs and a multilayer autoencoder created from the pre-training and stacking of RBMs.

After the multilayer autoencoder is created, PR backpropagation can be used to fine-tune the weights of autoencoders provided the initial weights and biases are close to the optimum solution.

All the weights of the multilayer autoencoder can be fine-tuned in a single step provided the weights are pre-trained with RBM. Since the weights are close to a good solution after RBM pre-training, PR backpropagation works well for fine-tuning the weights in the multilayer autoencoders.

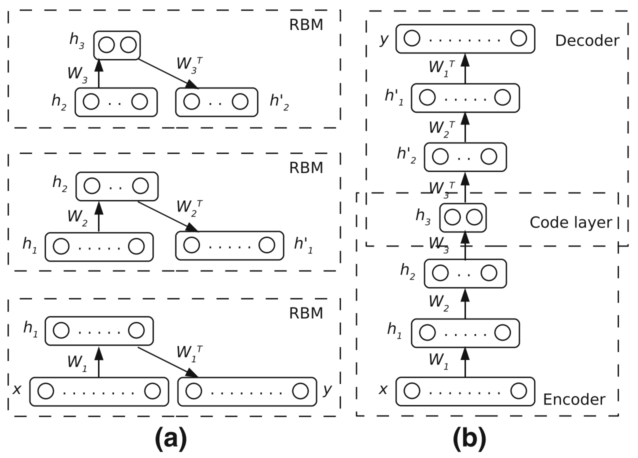


Fig. 3 **a** Pre-training of a stack of RBMs. **b** After pre-training and stacking, a multilayer autoencoder composed of RBMs is created

4 Image reconstruction using autoencoders

The image reconstruction problem starts with the training of autoencoders. The sizes of the hidden layers are prefixed. Training images are fed into the autoencoders to fine-tune the weights. The encoder part of the autoencoder reduces the dimensionality of the image pixels and represents them in a smaller code space in the middle hidden layer. The codes are then reconstructed back into images by the decoder part of the autoencoder at the output layer. The MSEs are used to measure the reconstruction performance.

In the experiment for comparing the three autoencoders, the weights and biases of the stacked autoencoder SA-MBP are initialized with the small random values. On the other hand, the weights and biases of the stacked RBMs, namely, SRBM-SBP and SRBM-MBP are initialized with the same method but pre-trained according to the Boltzmann learning rule in Eq. 6. The training of the autoencoder starts with one hidden layer, and the outputs from the hidden layer are then used for training the next autoencoder. The new autoencoder is going through the same process of training. Both the trained autoencoders will then be stacked into a single autoencoder with three hidden layers. The weights and biases of the SA-MBP are fine-tuned using the PR backpropagation after every stacking. The same fine-tune process is also applied to the SRBM-MBP. PR backpropagation is carried out only after the last stacking for the SRBM-SBP.

The second experiment is conducted to evaluate the effect of using image patches (instead of whole image) for training the autoencoder. For this experiment, only SRBM-MBP is made use of. Image patches are sampled consecutively from the original images. All the other parameters remain the same as the previous experiment except the layer sizes. The layers are of smaller sizes compared to those in the previous experiment. Using image patches reduces the layer sizes and also the training time significantly even though the training epochs are the same.

Next experiments are conducted to evaluate the performance of PCA for image reconstruction and recognition. Let Pk and F represent the k-dimensional eigenvectors that hold the principal components and the matrix of eigenimages, respectively. P_k and F are related as

$$F = P_k^T(X - \mu) \tag{7}$$

where X and μ represent, respectively, the original images and the eigenvalues. For reconstructing the original images from the k-dimensional eigenvectors, the following equation is used.

$$\hat{X} = P_k F + \mu \tag{8}$$

The difference between the reconstructed images $\hat{X}$ and the original images is calculated in terms of mean squared error.

For face image reconstruction, Olivetti Research Laboratory (ORL) face dataset is used in our experiments. On the other hand, MNIST dataset of handwritten digits is used for the handwritten digit image reconstruction. The MNIST dataset is a subset of a larger dataset available from the National Institute of Standards and Technology (NIST). The training and testing sets are divided according to most of the other benchmarking experiments carried out by other researchers.

4.1 Reconstruction on ORL face dataset

The dataset contains 400 images of size 112 × 92 for forty different people with 10 images for each person. The images are rescaled to size 37 × 30 using nearest neighbor interpolation. The pixel values of the images are then normalized to be in the range from 0 to 1. The dataset is then divided into 200 training images, which contain the first five images of each person, and 200 test images, which contain the last five images of each person.

4.1.1 Training with the whole image

The layer sizes of the autoencoders are preset as follows: (37 × 30)-500-300-100-30-100-300-500-1110. The preset sizes might not be optimum. However, they are found to be reasonably fit compared to the other configurations by experiments. The deepest hidden layer (with 30 neurons) uses linear activation function, while the other layers use sigmoid activation functions. All the layers are fully connected.

Starting from a conventional one-hidden-layer network, the stacked autoencoder is initialized with small random weights and biases ranging from 0 to 0.1. For the two architectures of stacked RBM, the weights and biases are pre-trained using RBM for 50 epochs. The number of epochs used for training the different layers of SA-MBP and SRBM-MBP is 50→10→50→10→50→10→50. For SRBM-SBP, the same number of total epochs, namely 230 is applied for training. The details of the training using PR backpropagation are illustrated in Fig. 4. The RBM pre-training is not shown in the figure.

Figure 5 shows the test results obtained for SA-MBP. It is seen from the figure that the MSE for both training and testing phases are high (spikes) at the beginning of the training of each individual network, and they have slower convergence in the end. Figure 6 shows the zoom-in view of Fig. 5. From this figure, we observe that this model has very low difference between the training and testing MSEs compared to two RBM architectures.

The plot of MSE for SRBM-SBP is shown in Fig. 7. The figure reveals that the RBM pre-trains the autoencoder effectively as seen by the good convergence of the training phase. However, this does not happen in the testing phase, since the SRBM-SBP is too focused to the training data rather than to the testing data.

Figure 8 shows the performance of the SRBM-MBP. It is clear from this figure that the RBM has proven to be a good pre-training method as it leads to a faster convergence compared to the SA-MBP. The same phenomenon, namely, better convergence for training data occurs in this architecture also and as a result, the MSE between the training and testing phases is high.

One important phenomenon observed from these experiments is that the RBM pre-training tends to make the autoencoders more focused on the training data, resulting in a low generalization for testing data. The difference between the training and testing MSEs is higher in the case of SRBM-SBP and SRBM-MBP compared to SA-MBP. This is possibly caused by the fact that the amount of training data is not large enough to estimate the Boltzmann distribution properly.

From Figs. 6 and 8, it is noticed that the MSEs become worse whenever a new hidden layer is stacked on to the existing autoencoder. This means that the dimensionality

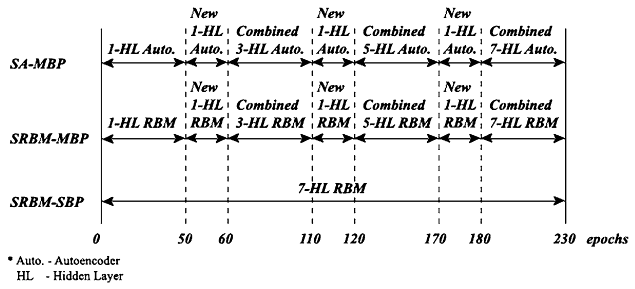


Fig. 4 Details of PR backpropagation training

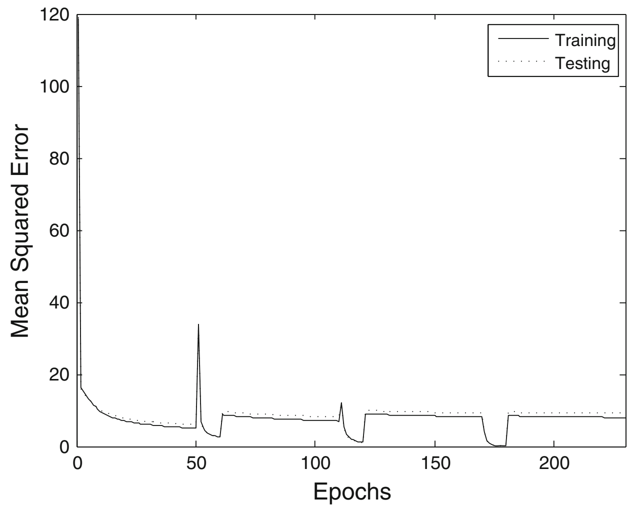


Fig. 5 MSE of SA-MBP on ORL dataset

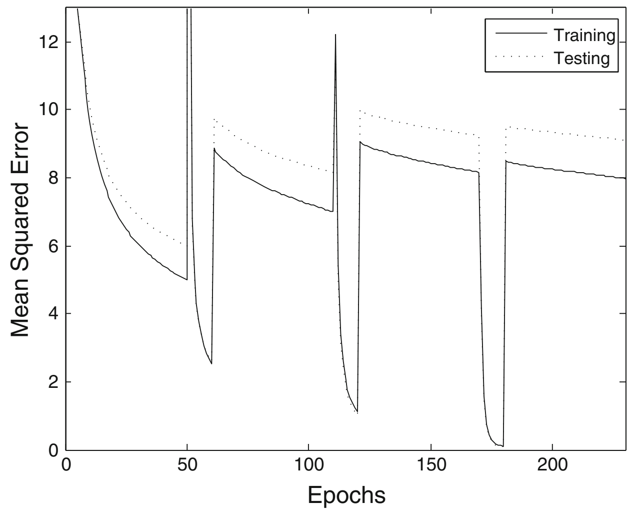


Fig. 6 MSE of SA-MBP on ORL dataset (zoom-in view)

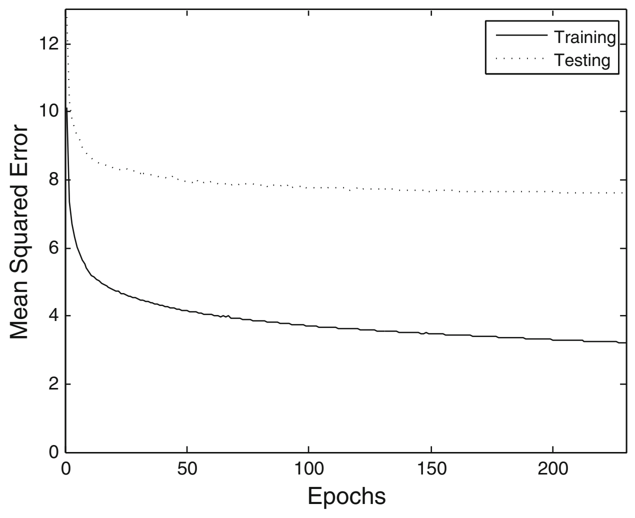


Fig. 7 MSE of SRBM-SBP on ORL dataset

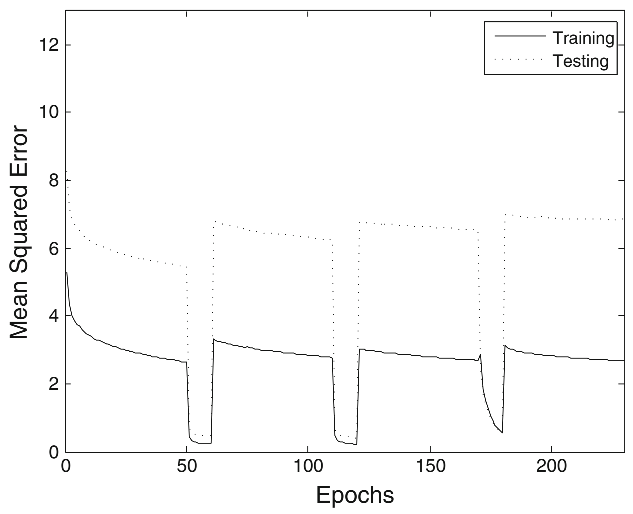


Fig. 8 MSE of SRBM-MBP on ORL dataset

reduction of autoencoders occurs at the expense of reconstruction efficiency as shown by the increase in the value of MSEs for each stacking.

Table 1 shows the testing MSEs obtained for the three architectures. From the results, we can conclude that the SRBM-MBP outperforms the other two architectures with respect to the reconstruction error. This is expected due to two reasons. First, the model is pre-trained using RBM, the weights are pre-adjusted near to global minima. Secondly, the multiple BP applied for each stacking performs better than the single BP in minimizing the MSE. Between these two conditions, the first one, namely pre-training using RBM is more vital compared to the second one, namely multiple BP. However, it is noticed that PCA performs the best. This is due to the fact that the training data size (200) used is considerably too small for training an autoencoder. As shown in Table 3, with larger datasets (MNIST), the autoencoders outperform the PCA.

4.1.2 Training with image patches

The experiments with image patches for training are conducted using the SRBM-MBP network. The ORL images are cropped and rescaled a bit at the center so that the final size is 32 × 32. Two experiments are conducted using different image sizes. Image patches of size 4 × 4 and 8 × 8 are sampled consecutively for the first and second experiments, respectively. In the first experiment, there are 64 patches extracted from each image, making up a total of 12800 patches for training. The layer sizes of SRBM-MBP used in the first experiment are (4 × 4)-16-8-4-2-4-8-16-(4 × 4). In the second experiment, 16 patches are extracted from each image, making up a total of 3200 patches for training. The layer sizes of SRBM-MBP used in the second experiment are (8 × 8)-64-32-16-8-16-32-64-(8 × 8). The number of epochs used for training the layers of both the networks is 50→10→50→10→50→10→50.

Table 2 shows the test MSE values obtained for the SRBM-MBP using image patches of different sizes. In both the experiments, 128 dimensions (64 × 2 for the first experiment and 16 × 8 for the second experiment) are used to represent each face image. Even though more dimensions are used compared to the cases of whole image and PCA, the results are worse. This is another example of poor performance of autoencoders due to small size datasets. The results are totally different when large datasets are used as shown in the next section. From Table 2, it is noticed that the architecture with more weights gives better reconstruction over architecture with lesser weights.

4.2 Reconstruction on MNIST handwritten digit dataset

The MNIST dataset contains handwritten digit images of size 14 × 14. There are 60000 training images and 10000 test images. The pixel values of images are normalized to be in the range of 0 to 1 before feeding into the autoencoders.

4.2.1 Training with the whole image

For experiments comparing the performances of the three autoencoders, 6000 and 1000 images are used for training and testing, respectively. The images are sampled randomly from the original MNIST dataset according to the distribution ratio of the digits. The three autoencoders are preset to comprise 7 hidden layers in addition to the input and output layers. The layer sizes are as follows: (14 × 14)-1000-500-250-30-250-500-1000-196. Normally, the hidden layers of the encoder network are of smaller size than the input layer for dimensionality reduction purpose.

Table 1 MSE of testing phase after 230 epochs on ORL dataset

Models	Testing MSE
SA-MBP	9.1092
SRBM-SBP	7.6178
SRBM-MBP	6.8483
30-dimensional PCA	4.921

Table 2 Test MSE of SRBM-MBP using image patches on ORL dataset

No. of patches	Layer sizes of SRBM-MBP	Testing MSE
64	(4 × 4)-16-8-4-2-4-8-16-(4 × 4)	9.2715
16	(8 × 8)-64-32-16-8-16-32-64-(8 × 8)	8.1828

However, more hidden units are used in the experiments to retain more information from the input data. The deepest hidden layer (with 30 neurons) uses linear activation function, while the other layers use sigmoid activation functions.

The stacked autoencoder SA-MBP is initialized with small random weights and biases ranging from 0 to 0.1, while the two architectures of stacked RBM, namely, SRBM-SBP and SRBM-MBP are pre-trained for 50 epochs. The number of epochs used for training different layers of SA-MBP and SRBM-MBP is 50→10→50→10→50→10→50. In the case of SRBM-SBP, the same number of total epochs, namely 230, is applied.

Figure 9 shows the experimental results obtained for SA-MBP. Figure 10 shows the zoom-in view of Fig. 9. It is observed that the MSE for both training and testing phases are extremely high at the beginning of training of each individual network. However, due to the large number of training datasets employed (6000 images), a good convergence is achieved at the end of the fine-tuning. A very low difference between the MSEs of training and testing can be observed from the figures.

The plot of MSE versus Epochs for SRBM-SBP is shown in Fig. 11. It is clear from this figure that the RBM pre-trains the network effectively as seen by the convergence of the training and testing phases. The difference between the MSEs of training and testing phases is much smaller compared to the one obtained on ORL face dataset as seen from Figs. 7 and 11. This is because the larger dataset (MNIST) used for training the autoencoders makes the autoencoders to have better generalization.

Figure 12 shows the performance of the SRBM-MBP. It is clear from this figure that the RBM has proven to be a good pre-training method as the spikes of MSEs are not very high compared to those of SA-MBP.

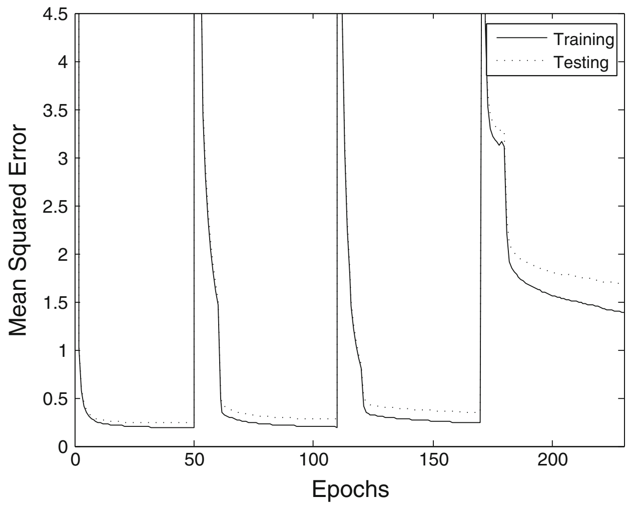


Fig. 10 MSE of SA-MBP on MNIST dataset (zoom-in view)

The difference between the MSEs of training and testing phases is much lower in the case of MNIST dataset compared to the one obtained on ORL dataset. This is mainly due to the large training data available in MNIST dataset. From Figs. 10 and 12, it is seen that in the case of SA-MBP and SRBM-MBP, the MSEs become worse whenever a new hidden layer is stacked on to the existing structure. This means that the dimensionality reduction of autoencoders occurs at the expense of reconstruction efficiency as shown by the increase in the value of MSEs after each stacking.

Table 3 shows the testing MSEs obtained from the experiments. For the same number of training epochs, it is once again found that the SRBM-MBP model has the minimum testing MSE among all the architectures. The MSEs of all the three architectures are lower compared to

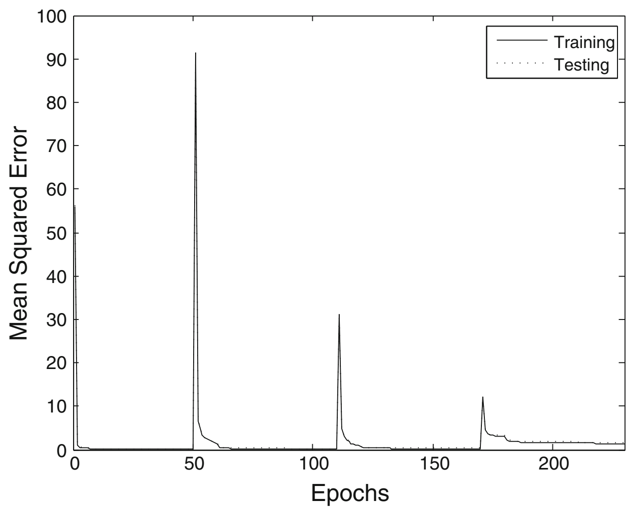


Fig. 9 MSE of SA-MBP on MNIST dataset

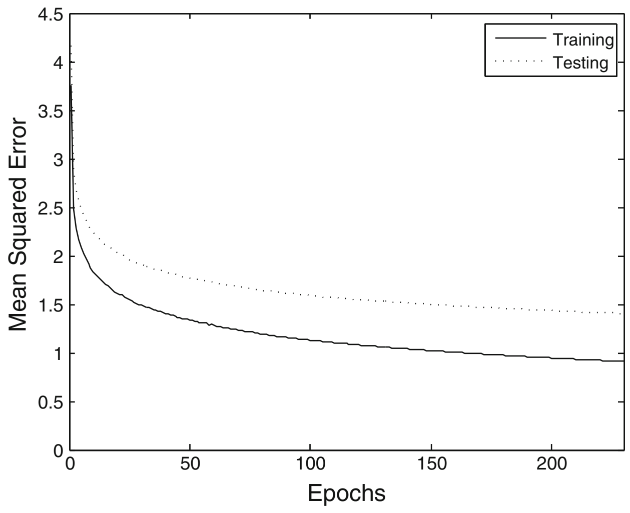


Fig. 11 MSE of SRBM-SBP on MNIST dataset

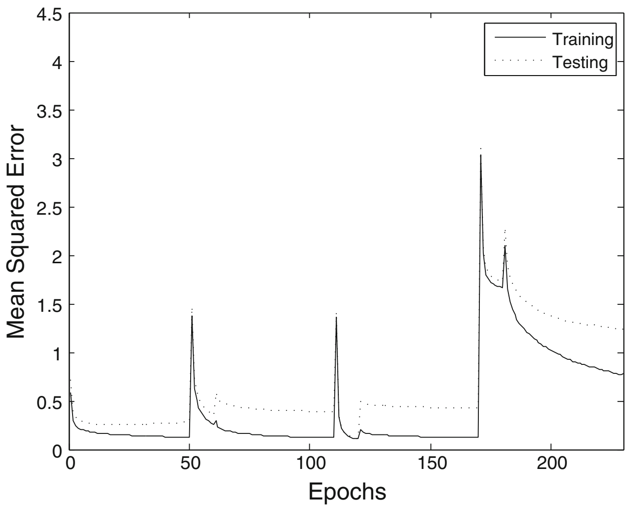


Fig. 12 MSE of SRBM-MBP on MNIST dataset

that of PCA in this case. As mentioned earlier, the results show that the autoencoders outperform the PCA when they are trained with a large dataset (MNIST).

Figure 13 shows the original and reconstructed images for the different models. The MSEs for the last four rows are 1.231, 1.403, 1.685 and 3.193, respectively. We can conclude that the SRBM-MBP outperforms the other two architectures and the PCA with respect to the reconstruction error.

Figure 14 shows the training time required for the three architectures using an AMD Athlon 64 X2 Dual-Core 5200+ 2.6GHz processor with 3G of RAM.

It is noticed that the SRBM-SBP is the most time consuming model compared to the other two architectures. This is because in this model, the fine-tuning process of the weights is performed only after the last stacking. Since the whole network is huge (with many hidden layers), the fine-tuning process is more time consuming.

4.2.2 Training with image patches

Experiments are conducted with SRBM-MBP architecture where image patches, instead of whole image are used for training. In these experiments, the MNIST images are cropped with padding of 1 pixel to make the image size to be 12 × 12. Image patches of size 4 × 4 are sampled

Table 3 MSE of testing phase after 230 epochs on MNIST dataset

Models	Testing MSE
SA-MBP	1.685
SRBM-SBP	1.403
SRBM-MBP	1.231
30-dimensional PCA	3.193

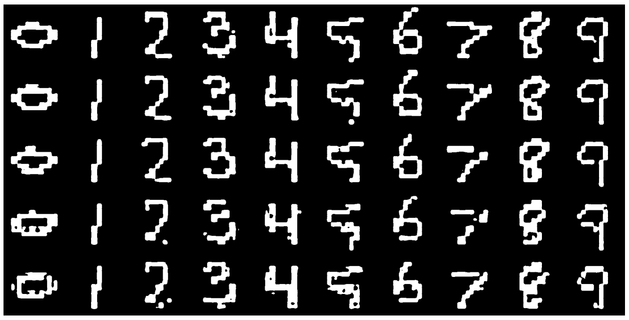


Fig. 13 Original and reconstructed images. *Row 1*: original; *row 2*: SRBM-MBP; *row 3*: SRBM-SBP; *row 4*: SA-MBP; *row 5*: 30-dimensional PCA

consecutively from the original images. Thus, there are 9 patches extracted from each image, making up a total of 54000 patches from 6000 training images. The layer sizes of SRBM-MBP are (4 × 4)-16-8-4-2-4-8-16-(4 × 4). The number of epochs used for training the different layers is 50→10→50→10→50→10→50. From the experiment, it is found that the result for 9 patches is bad. It is mainly due to the reason that most of the patches are with mean value of 0 (totally black), making the training samples imbalanced. Another experiment is conducted, in which 4 patches of size 4 × 4 are extracted from the center part of each image, making a total number of 24000 training samples. The same architecture of SRBM-MBP and training epochs are used. Table 4 shows the values of test MSE obtained in both of the experiments.

From Table 4, it can be noticed that significant improvement is achieved with 4 patches (from the center portion) even though less training samples are used. It is unfair to compare the above results with the result obtained using the whole image for training. This is because the

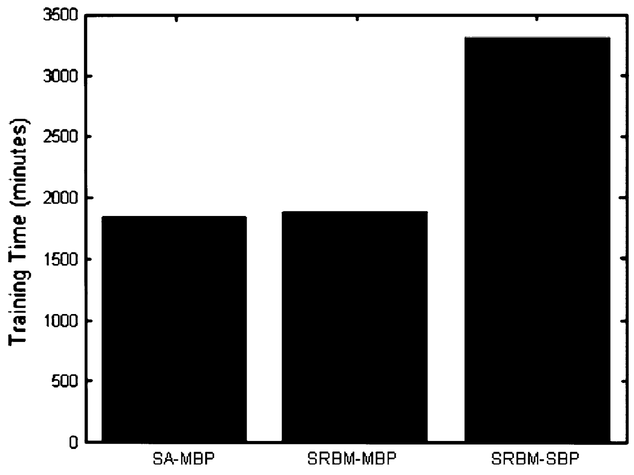


Fig. 14 Training time for the three architectures

Table 4 Test MSE of SRBM-MBP using image patches on MNIST dataset

Number of patches per image	Test MSE (whole image)	Test MSE (per patch)
9 patches (size 4 × 4, from the whole image)	7.6453	0.8495
4 patches (size 4 × 4, from the center portion of the image)	2.6635	0.6659

9-patch and 4-patch experiments with SRBM-MBP above use dimensions of 18 (9 × deepest layer size) and 8 (4 × deepest layer size), respectively to represent the code space, whereas a dimension of 30 (1 × deepest layer size) is used for representing the code space when the whole image is used for training. The 4-patch experiment is repeated with a different architecture comprising layers of sizes: (4 × 4)-32-32-16-7-16-32-32-(4 × 4). In this case, a dimension of 28 (4 × deepest layer size) is used to represent the code space. Table 5 shows the result obtained for the experiment, and it is noticed that the MSE is reduced significantly.

The result in Table 5 shows the advantages of using image patches for training the autoencoders as the test MSE value in this case is even lower than the values obtained with the three types of autoencoders and PCA using the whole image for training. From the results shown in Tables 3 and 5, it is clear that the essential factor in getting low MSE depends mainly on the complexity of the autoencoder architecture. The weights from a large autoencoder with the whole image as the input, W_{whole}, produces greater MSE compared to n patches of weights, W_{patch}, with a simpler architecture. In other words, we can say that $MSE(W_{whole}) > n \times MSE(W_{patch})$. This is because of the fact that the weights in the deeper layers are easier to be fine-tuned in the simpler architecture using image patches. Another important factor that contributes good results is the size of the hidden nodes. With less number of hidden nodes, it is not possible to get good generalization, while too many hidden nodes increases the complexity of the architecture. It can be observed that the hidden nodes in the first two layers of the autoencoders hold the most important representations of the input. Thus, the size of the hidden nodes in the first two layers must be the same or greater than the size of the input layer to have enough weights to hold the features of inputs.

Table 5 Test MSE of SRBM-MBP with 28 dimensions

Number of patches per image	Test MSE (whole image)
4 patches (28 dimensions)	0.8232

It is expected that similar results will be obtained with the other autoencoder architectures, namely SA-MBP and SRBM-SBP, when image patches are used for training. Thus, it can be concluded that using image patches instead of the whole image will not only reduce the training time but will also improve the performance with reduced MSE values. These findings are essential for image compression using autoencoders.

5 Image recognition using autoencoders

It is also of interest to investigate the correlation between the reconstruction and the recognition of an image using autoencoders. The image recognition problem starts with the training of autoencoders. The sizes of the hidden layers are prefixed. Training images are fed into the autoencoders to fine-tune the weights. The autoencoders learn the kernel features of the images and represent them in the code space (corresponding to the deepest layer). Those images with strong relationship and which are identical are situated close to each other in the code space. On the other hand, the images that are not correlated are located far away in the code space.

After the training (learning) of autoencoders, the feature codes for the corresponding image are obtained from the code (deepest) layer of the autoencoders. To recognize an image, the test image is fed into the trained autoencoder to produce the relevant feature codes for the particular image. Subsequently, k-nearest neighbor classifier is employed to calculate the Euclidean distances of the feature codes between the training images and the test images [9]. The minimum distance of the feature code determines the group to which that test image belongs. In this case, the autoencoders are playing the roles of feature extractors and the k-nearest neighbor is used as a classifier.

The recognition problem using the autoencoder can be implemented using the following steps:

1. Train the autoencoder with the images to be recognized.
2. Obtain the feature codes for each training image using the trained autoencoder.
3. Apply the test image and obtain the corresponding feature codes.
4. Calculate the Euclidean distances between the feature codes of the test image and the feature codes of all the training images.
5. Based on the minimum Euclidean distance, the cluster (group) to which the test image belongs is identified.

In the experiments conducted for comparing the performances of the three autoencoders, the feature codes from the deepest hidden layer are extracted for classification. The performance results of the autoencoders are also compared with that of PCA. Experiments are also conducted using image patches for training. Image patches are sampled consecutively from the original images. The SRBM-MBP is used in this experiment. The hidden layers are of smaller sizes compared to the ones used in the previous experiment with the whole image. Using image patches reduces the training time significantly even though the number of training epochs remains the same.

A discriminative method to digit recognition is to train the autoencoder to output one of the ten classes based on the input images. It is also possible to train ten autoencoders by fitting separate digit to each model. Recognition is based on the model with the highest density to the input image [10, 11]. The former method is used in our experiments. For face recognition, ORL face dataset is used in our experiments. On the other hand, MNIST dataset of handwritten digits is used for the handwritten digit recognition. The MNIST dataset is a subset of a larger dataset available from the NIST.

5.1 Recognition on ORL face dataset

The dataset contains 400 images of size 112 × 92 for forty different people with 10 images for each person. The images are rescaled to size 37 × 30 using nearest neighbor interpolation. The pixel values of the images are then normalized to be in the range from 0 to 1. The dataset is then divided into 200 training images, which contain the first five images of each person, and 200 test images, which contain the last five images of each person.

5.1.1 Training with the whole image

The layer sizes of the autoencoders are preset as follows: (37 × 30)-500-300-100-30-100-300-500-1110. The deepest hidden layer (with 30 neurons) uses linear activation function, while the other layers use sigmoid activation functions. All the layers are fully connected. The SA-MBP is initialized with small random weights and biases ranging from 0 to 0.1, while the other two architectures, namely SRBM-SBP and SRBM-MBP are pre-trained for 50 epochs. The number of training epochs used for training the different layers of SA-MBP and SRBM-MBP is 50→10→50→10→50→10→50. The same number of total epochs, namely 230 is applied to the SRBM-SBP.

Table 6 Recognition rate of 30 dimensional codes on ORL dataset

Models	Recognition rate (%)
SA-MBP	80.5
SRBM-SBP	86
SRBM-MBP	86
30-dimensional PCA	88

Table 6 shows the recognition results for the three architectures and PCA.

It is noted that the PCA outperforms all the three autoencoders. This may be due to three possible reasons. First, the dataset is not large enough for training the autoencoders. Secondly, the training parameters or architectures of the autoencoders may not be optimum ones. Lastly, it might be due to the so-called inadequacy problem for assessing perceptual information using sum squared error function [12].

5.1.2 Training with image patches

The experiments with image patches for training are conducted using SRBM-MBP. Image patches of size 4 × 4 and 8 × 8 are sampled from the original image size of 32 × 32, making a total of 64 and 16 patches from each image, respectively. The layer sizes used for SRBM-MBP for these two cases are as follows: (4 × 4)-16-8-4-2-4-8-16-(4 × 4) and (8 × 8)-64-32-16-8-16-32-64-(8 × 8), respectively. The codes from the deepest hidden layer are extracted for each image patch. It may be noted that in both these cases a dimension of 128 (number of image patches × deepest hidden layer size) is used for the feature space. Table 7 shows the recognition rates obtained in both the experiments. As in the previous case, image patches of bigger sizes yield better results.

5.2 Recognition on MNIST handwritten digit dataset

5.2.1 Training with the whole image

Table 8 shows the recognition rates of the three architectures. It is found that the SRBM-MBP that gives the best result for reconstruction (Table 3) performs poorly for the recognition problem. In fact, its performance is the worst among the three. The SRBM-SBP has the maximum recognition rate, showing that the recognition performance is better for the autoencoder which is trained overall at one shot. For SRBM-MBP, by changing the activation functions at the deepest hidden layer to sigmoid functions, an improvement on recognition accuracy is obtained. It is believed that the improvement of recognition rate using the sigmoid functions at the deepest hidden layer is applicable to the other two architectures as well. For more detailed comparisons of the performances of the autoencoders and

Table 7 Recognition rate of SRBM-MBP using image patches on ORL dataset

Number of patches per image	Recognition rate (%)
64 patches (size 4 × 4)	86
16 patches (size 8 × 8)	88.5

Table 8 Recognition rate on 30 dimensional codes on MNIST dataset

Models	Recognition rate (%)
SA-MBP	92.6
SRBM-SBP	92.8
SRBM-MBP	91.9
SRBM-MBP (sigmoid functions at the deepest layer)	93.1
30-dimensional PCA	91.9

Table 9 Recognition rate of SRBM-MBP using image patches on MNIST dataset

Layer sizes of SRBM-MBP	Recognition rate (%)
(4×4)-32-32-16-7-16-32-32-(4×4)	90.0
(6×6)-72-32-16-7-16-32-72-(6×6)	91.3

other techniques (PCA, SVM, etc.) on MNIST dataset, refer to [1] and [5]. Recognition error rate of 1.2% is reported in [1], where 60,000 images are used for training the autoencoders that resemble SRBM-SBP.

5.2.2 *Training with image patches*

For experiments using image patches, 4 patches of sizes 4×4 and 6×6 are extracted from the center portion of each image. The layer sizes used in these experiments are (4×4)-32-32-16-7-16-32-32-(4×4) and (6×6)-72-32-16-7-16-32-72-(6×6) for image patches of size 4×4 and 6×6, respectively. The number of epochs used for training the both networks is 50 $10 \rightarrow 50 \rightarrow 10 \rightarrow 50 \rightarrow 10 \rightarrow 50$. Table 9 shows the recognition rate obtained in the experiments. It is found that the performance of SRBM-MBP (with 28 dimensions in both experiments) using image patches is not as good as the one using the whole image for training. However, these experiments clearly demonstrate the feasibility of using image patches for training, which saves training time and also reduces the architecture size. It is noticed from the experiments that using bigger patches yields better recognition rates over the smaller patches.

6 Conclusions

We have shown that the autoencoders can be used effectively for image reconstruction and recognition applications. The experimental results show that the autoencoders outperform PCA if large training samples are provided. Among the three types of autoencoders, the best performance is achieved by the SRBM-MBP for image reconstruction application. On the other hand, the SRBM-SBP architecture gives the best result for image recognition application. Experiments have also been performed to show that the autoencoders can be trained successfully using image patches instead of the whole image. The most important contribution of this paper is in showing that significant improvements on image reconstruction can be obtained using image patches instead of whole images. Further investigations on image recognition using different SRBM-MBP architectures and image patches are needed to improve the recognition rate.

References

1. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313:504–507
2. Bengio Y, Lamblin P, Popovici D, Larochelle H (2006) Greedy layer-wise training of deep networks. Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, Tech Rep 1282, Aug 2006
3. Tan CC, Eswaran C (2009) Using autoencoders for mammogram compression. J Med Syst. doi:10.1007/s10916-009-9340-3
4. Polak E, Ribiére G (1969) Note sur la convergence de methods de directions conjures. Revue Francais Information Recherche Operationnelle 16:35–43
5. Larochelle H, Erhan D, Courville A, Bergstra J, Bengio Y (2007) An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th international conference on machine learning (ICLM). ACM, Corvalis, Oregon, USA, pp 473–480
6. Hinton GE, Boltzmann Machine. Online. Scholarpedia, 2(5):1668. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-5586
7. Tee YW, Hinton GE (2000) Rate-coded Restricted Boltzmann Machines for face recognition. In: Neural information processing systems 13. MIT Press, Cambridge, pp 908–914
8. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning. Corvallis, OR, USA, pp 791–798
9. Cover T, Hart P Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27
10. Hinton GE, Revow M, Dayan P (1995) Recognizing handwritten digits using mixtures of linear models. In: Tesauro G, Touretzky DS, Leen TK (eds) Advances in neural information processing systems 7. MIT Press, Cambridge, pp 1015–1022
11. Hinton GE, Dayan P, Revow M (1997) Modeling the manifolds of images of handwritten digits. IEEE Trans Neural Netw 8(1):65–74
12. Hinton GE, Salakhutdinov RR (2006) Supporting online material for reducing the dimensionality of data with neural networks. Science 313(5786), Online. Available: http://www.sciencemag.org/cgi/content/full/313/5786/504/DC1