Applied Mathematics and Computation 355 (2019) 325–342

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Face recognition via Deep Stacked Denoising Sparse Autoencoders (DSDSA)

Pelin Görgel*, Ahmet Simsek

Department of Computer Engineering, Istanbul University-Cerrahpasa, Turkey

ARTICLE INFO

Keywords:
Face recognition
Deep learning
Sparse autoencoders

ABSTRACT

Face recognition is still a hot topic under investigation due to many challenges of variation including the difference in poses, illumination, expression, occlusion and scenes. Recently, deep learning methods achieved remarkable results in image representation and recognition fields. Such methods extract salient features automatically from images to reduce the dimension and obtain more useful representation of raw data. In this paper, the proposed face recognition system namely Deep Stacked Denoising Sparse Autoencoders (DSDSA) combines the deep neural network technology, sparse autoencoders and denoising task. Autoencoder is used to construct a neural network that learns an approximation of an identity function by placing constraints to learn fine representations of the inputs. Autoencoders have unique capabilities in dealing with interpretation of the input data; in this way produce more meaningful results. They are successfully applied to many object recognition fields. For the classification task, two classifiers were used, namely multi-class SVM and Softmax classifier. Experimental results on known face databases including ORL, Yale, Caltech and a subset of PubFig show that the proposed system yields promising performance and achieves comparable accuracy.

1. Introduction

Face recognition has received considerable attention in researches in past decades, because of its wide area of use in various fields [1,2]. Recently, it has also become particularly competitive in smart phone applications. Despite the good success achieved, face recognition is still under investigation due to the variation of constraints and determinants of which the system has been adapted, including the difference in poses, illumination, expression, occlusion and scene. Thus, it seems that some face recognition systems successfully which are applied to some databases with specific constraints, do not achieve the same result when examined under different conditions. Deep learning is a sophisticated area of machine learning and the usage of deep neural networks leads to promising results in image representation, especially in nonlinear feature extraction of high levels. Autoencoder is a special artificial neural network which has unique capabilities in dealing with unlabeled data through machine learning with a clear efficiency in extraction of nonlinear features from high-dimensional datasets. Autoencoders have multiple formats and approaches that have been used to solve different problems in many domains.

In recent years, face recognition has been studied extensively and many methods have been raised to achieve more accurate results. Those methods can be summarized into the ones that use deep learning and don't use.

* Corresponding author.
E-mail address: paras@istanbul.edu.tr (P. Görgel).

https://doi.org/10.1016/j.amc.2019.02.071
0096-3003/

Early studies are divided into template based and feature based methods. In template based methods the goal is to create a general template for each face. Eigenface and Fisherface [3] are the most famous methods in this category and many methods of dimensionality reduction such as Principal Component Analysis (PCA) [4], Linear Discriminate Analysis (LDA) [3] and Local Linear Embedding (LLE) [5] were proposed. Nonlinear extensions based on Kernel-PCA [6] and Kernel-LDA [7] were implemented in order to solve the problem of nonlinear distribution of face images. Feature based methods LBP [8], Gabor [9], HOG [10], Curvelet Transform [11] were applied to extract robust features which were used to represent the different faces. In classification field, Support Vector Machine (SVM) [12] was proposed and used to classify the extracted features. And some further classifiers like Nearest Neighbor (NN) [13] and Nearest Subspace (NS) [14] also were proposed. Distance-based methods which computed a similarity metric were also used [15–17]. The sparse representation was also gradually utilized to solve face recognition [18–24]. However, despite some success that these methods achieved, they continue to suffer from their linear nature, generalizability problem or shallow structures.

Lately, thanks to their great successes in image representation and recognition fields, deep learning methods achieved great investigating on face recognition [25–31].

Deep Convolutional Neural Networks (CNN) have taken the main interesting in many researches [25–27,32–42]. DeepFace [26] trained a deep CNN to classify faces. Siamese network was suggested by Chopra et al. [33] for deep metric learning. And Huang et al. [25] learned a generative deep model without supervision. Sun et al. [27] used supervised multiple deep models and Zhu et al. [34] used multiple deep CNN. FaceNet [35] by Schroff et al. achieved state-of-the-art face recognition performance using deep convolutional network.

Hinton and Salakhutdinov [43] proposed a deep autoencoders network as a training algorithm. Accordingly, autoencoders have been researched extensively in the field of face recognition and related tasks. Kan et al. [44] used stacked progressive autoencoders for face recognition. Zhang et al. [45] proposed Coarse as a fine autoencoder for real time face alignment. Gao et al. [46] used stack supervised autoencoders for face recognition. Ding and Tao [47] used stacked autoencoders to learn face representations. Zhang et al. [37] investigated sparse autoencoders with Softmax classifiers to build deep hierarchical for face recognition. Recently, Li et al. [48] proposed deep autoencoders with dropout. Zeng et al. used deep sparse autoencoders for facial expression recognition [49].

Several design choices have been proposed for the face recognition issue. Shallow methods were excluded because they allow only the linear transformation while non-linear deep architecture methods can learn more complicated representations. Furthermore, many deep architecture designs have been proposed so far. Despite the great success of CNNs, there are many determinants in their use. CNNs are mostly used for very specific tasks. Because of its complexity and local connectivity, the generalization performance issue poses a challenge especially for highly non-linear or time-varying object appearance variations also there are many more settings to manage [31]. For example, VGG is a 16 layer neural network, while Microsoft's ResNet model has 152 layers. However, the CNN is designed to handle supervised learning which depends on labeled data. Another issue is that it mostly needs large data for training, for example DeepFace [26] was trained with 4 million images and FaceNet [35] was trained with 200 million images.

On the other hand, autoencoder is a learning algorithm investigates unlabeled data. It is simple, intuitively understandable and easy to implement. Autoencoders are more general because they specify nothing about the network topology. The additional reason for choosing autoencoders in this work is they can be stacked on top of each other to form an effective feature extraction method while maintaining its simplicity.

In this paper, a deep learning based face recognition system was implemented. In the proposed Deep Stacked Denoising Sparse Autoencoders (DSDSA) system, a deep network was built to learn the feature representation of face images by automatic facial feature extraction. To reach a robust system two classifiers were used, multiclass-SVM and Softmax classifier. The experiments of the proposed (DSDSA) method provide comparable accuracy results in face recognition and it can be generalized to other recognition tasks thanks to its ability to extract robust features. The contributions of the proposed system can be outlined as follows:

- We propose a complete and robust face recognition system that also includes face detection and landmark localization and can deal with various face datasets containing unconstrained faces with variations in pose, expression, illumination and scene.
- The sparsity property effectively reduces the number of non-zero elements in the training process, so that the possibility of overfitting is substantially decreased in a trained model. In this study a sparsity constraint is specified to make the trained model lie on the balance point which is neither underfitting nor overfitting. The sparsity term also dramatically reduces the computational cost.
- To extract more robust features from variances of face images, an optimized sparse autoencoder is built in addition to denoising process.
- It can be observed that we achieve comparable results to other deep CNN and the state-of-the art methods (FaceNet [35] and DeepFace [26]) using much less data even when the training data come from a different database. This is a conclusion that may be applicable to many other tasks.
- Our proposed method is evaluated on four public datasets: Yale, Caltech, ORL and Pubfig using two robust classifiers; multiclass-SVM and Softmax classifier to reach more accurate results.

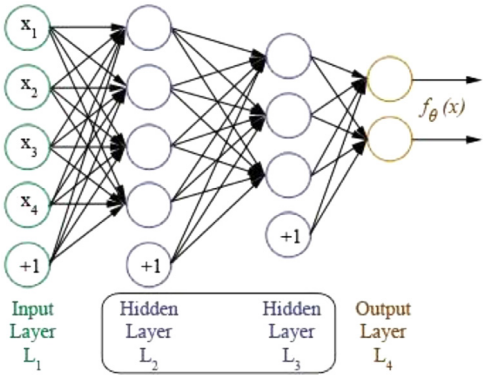


Fig. 1. Sample hierarchy of a deep network model with two hidden layers.

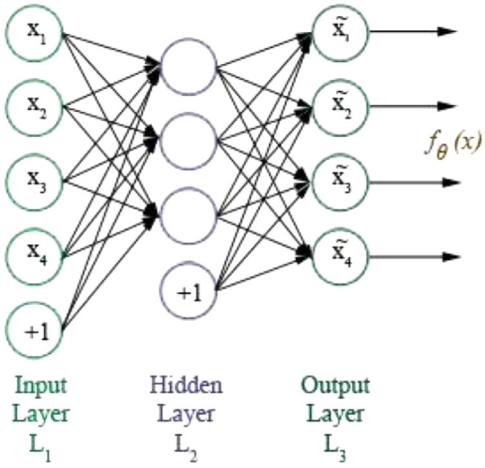


Fig. 2. A basic autoencoder.

As for the rest of the sections, the utilized methodologies are presented in Section 2 while the proposed face recognition system is wholly explained in Section 3. Section 4 gives the experimental results of the proposed system and discussion. The conclusion is provided in Section 5.

2. Method

2.1. Deep learning model

The deep learning term refers to learning network models with many hidden layers beside input and output layers, versus shallow networks containing only one hidden layer [50]. Each hidden layer in a deep model is composed of hidden units that take their inputs from the previous layer, calculate their activation according to the given weights and generate outputs related to their own features. These learnt hierarchical features enable the system to decode the output from the data automatically. Fig. 1 shows a simple hierarchy of deep network model.

2.2. Autoencoders

In the proposed face recognition system, an autoencoder was used as an unsupervised neural network aiming to minimize the error between the input data and its reconstruction (output). An autoencoder consists of three or more layers: an input layer, some number of hidden layers which form the encoding and an output layer whose units correspond to the input layer (Fig. 2). Since the outputs of the network are equal to the input, the autoencoder's goal is to learn an approximation of the identity function [46,51].

A basic autoencoder is architecturally similar to feedforward neural networks. It encloses two processes; encoding and decoding. Autoencoders use unsupervised learning to reduce input features and restore them respectively because including all features would confuse algorithms computed between layers. In autoencoders an unsupervised learning algorithm compresses representation; in this way, clustering algorithms produce more meaningful results. Let $\mathbf{x} \in [0, 1]^d$ be an input

vector for autoencoder, $\theta = \{W, b\}$ (W: weight matrix, b: bias vector) is encoding that maps input to $\mathbf{z} \in [0, 1]^{d'}$ (Eq. (1)),

$$z = f_\theta(x) = \sigma(Wx + b) \tag{1}$$

where $\sigma(\cdot)$ is a nonlinear activation function of the encoder like sigmoid or tanh.

Decoding process maps $\mathbf{z}$ back to reconstructed vector $x' \in [0, 1]^{d}$,

$$x' = g_{\theta'}(z) = \sigma'\left(W'z + b'\right) \tag{2}$$

where $\sigma'(\cdot)$ is a nonlinear activation function of the decoder. The weight matrix is W' and bias vector is b' for the decoder in $\theta' = \{W', b'\}$. Autoencoder has tied weights when W' is equal to W^T.

2.3. Classifiers for face matching

The extracted features of the previous step were used to match faces and compare one face to several faces, which is known as face identification or compare two faces to see if they belong to the same person or not, called face verification. Linear classifier is a function (Eq. (3)) that can be used to calculate the score of each class.

$$f(x_i, W, b) = Wx_i + b \tag{3}$$

In Eq. (3), W is the weights and b is the bias term. If the bias is ignored, the score of class j can be written as:

$$S_j = f(x_i, W)_j = Wx_i \tag{4}$$

Two commonly used losses were used for linear classifiers in the proposed system:

2.3.1. Multi-Class SVM

It is a development of standard SVM which is for binary to the multi-class classification, where the face is given to a class that has a larger score than the others by a given margin. Common approaches in Multi-Class SVM are “one-vs-all” and “one-vs-one” which belong to “ECOC”. Given l_i as the true class of ith example, the multi-class SVM loss for ith example is:

$$L_i = \sum_{j \neq l_i} \max\left(0, S_j - S_{l_i} + margin\right) \tag{5}$$

L_i is the loss for ith example, S_j, S_{l_i} are the score of class j and correct class respectively, *margin* is a parameter defined by the user. The $\max(0, -)$ is called as the hinge loss [32].

2.3.2. Softmax classifier

It is a multi-class version of Logistic Regression. Instead of the hinge loss used in SVM, a cross-entropy loss was used here to calculate the probability of each class (Eq. (6)).

$$L_i = -log\left(\frac{\exp^{f_{l_i}}}{\sum_j \exp^{f_j}}\right) \tag{6}$$

3. Face recognition system

The proposed DSDSA face recognition system consists of two modules: Face pre-processing and face recognition. Face pre-processing includes face detection, face landmarks localization and face normalization. Fig. 3 shows the modules of the implemented system.

3.1. Face pre-processing implementation

3.1.1. Face detection

Face detection is to determine the presence of the face in a given image and returns the face’s coordinates when the face is detected. In the proposed system as a pre-processing step for feature extraction and face recognition, integrated face detectors of Viola and Jones, Haar Feature-based Cascade Classifiers [52] and Group Sparse learning [53] are applied in order to detect and locate a bounding box around the face.

The Viola and Jones (VJ) detector is an approach for frontal 2D detection. It involves exhaustive searching of an entire image for faces, with multiple scales explored at each pixel using Haar-like rectangle features boosting classification. VJ is characterized as slow training, but very fast classification. In the proposed system VJ involves an image representation based on Haar wavelets, an integral image for rapid feature detection, the AdaBoost machine-learning method for selecting a small number of important features and a cascade combination of weak learners for classification. The original VJ implementation is designed for upright frontal image. Nevertheless, VJ does not deal adequately with unrestricted images that contain large differences in lighting, pose and expression. Therefore the system firstly tries to detect faces using VJ, thanks to its high speed and accuracy in frontal faces. However, VJ has a lower performance when dealing with faces in complicated conditions. If VJ fails to detect any face, the system immediately determines to try using group sparse learning (optimized part mixture model) which is a two-stage cascaded deformable shape model.

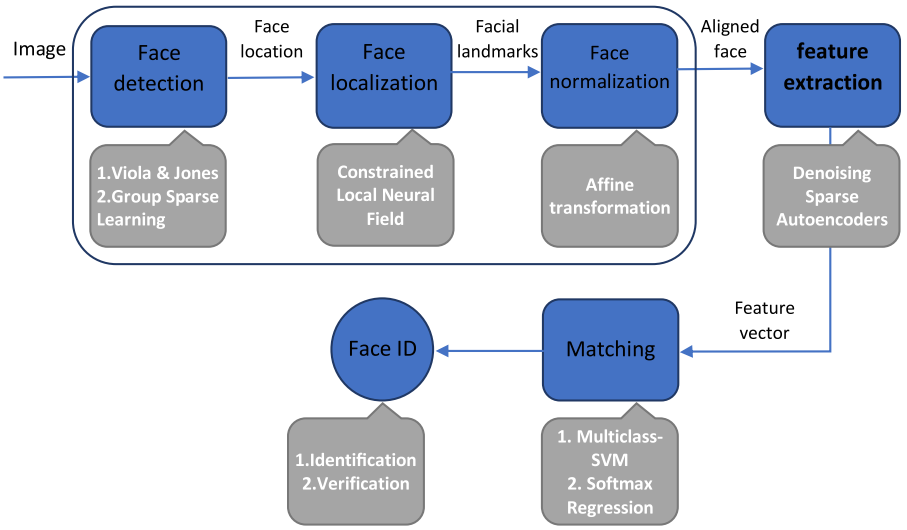


Fig. 3. Modules of the proposed system.

3.1.2. Face landmarks localization

Face Landmarks Localization (Facial feature detection) was used to detect landmarks positions of the facial feature points; eyes, contours, mouth, nose, and eyebrows. After face detection, facial landmarks were used in the next step to align facial images to a mean face shape. It is worth noting that the error in estimating landmarks leads to error in alignment. The Constrained Local Neural Field (CLNF) [54] model for robust facial landmark detection in the wild was used to detect 68 landmarks for each face. The eyes coordinates (landmarks) were utilized to rotate and align the face in the next step. CLNF is an undirected graphical model that can model the conditional probability of a continuous valued vector y depending on continuous x (the pixel intensity values in the support region). Given $X=\{x_1,x_2,\ldots x_n\}$ is a set of observed input variables, $y=\{y_1,y_2,\ldots y_n\}$ is a set of output variables to be predicted, $x_i \in R^m$ represents vectorised pixel intensities in patch expert support region, $y_i \in R$ is a scalar prediction at location i. The model for a particular set of observations is a conditional probability distribution with the probability density:

$$P(y|X) = \frac{\exp(\Psi)}{\int_{-\infty}^{\infty} \exp(\Psi)\, dy} \tag{7}$$

where $\int_{-\infty}^{\infty} \exp(\Psi)\, dy$ is the normalization function and Ψ is the potential function.

3.1.3. Face normalization

Face pose, position and illumination play an essential role in distinguishing between individuals. The goal of normalization is to minimize the difference of pose, position, and illumination for the same individual including the impact of non-facial details like background and clothing.

In this study the first step of normalization was done by face rotating, scaling, warping and cropping. The center points of the eyes were obtained from detected landmarks in the previous step, the line and distance between eyes centers helped to determine the scale and the angle of rotation which were used to make an affine warping. After warping, the face was cropped so that the eyes were centered in all faces. The cropping area was determined by the left eye center, while the margin around the eye is a parameter that usually takes a range between 0.1 and 0.3. Also the width and the height of cropping face are the parameters that can be determined manually. In experiments, 0.3 was chosen for margin and $[32 \times 32]$ pixels were selected for [width x height] respectively.

To obtain the photometric normalization, the illumination was normalized by the grey information. And pixels were rescaled to [0, 1] by dividing each pixel by 255, the mean was also subtracted to center the pixel values in the face image. The aligned and normalized faces were inserted into a matrix as well as saved in a -MAT file to use them independently in the following steps as 'ready:normalized' faces.

3.2. The proposed Deep Stacked Denoising Sparse Autoencoder (DSDSA) method for feature extraction and learning

Feature extraction can be considered as the heart of the system, where the face is represented by the extraction of the salient features that effectively reflect the individual differences between the faces. To obtain a greater efficiency, features were extracted from the images which were normalized in the previous step. In this study the proposed deep denoising sparse autoencoder has been used to effectively extract a robust set of features to represent each face as detailed in Section 3.

3.2.1. Training a deep model with greedy layer-wise training

Training a deep model generally suffers from time complexity arising from weights initialized randomly and local minima issues. In this study to overcome these difficulties, greedy layer-wise algorithm is used to pre-train of the DBN (deep belief networks) layers [55]. Firstly, the model's lower layer is trained with an unsupervised learning algorithm that yields initial set parameters for the first layer of the network. The obtained output behaves as the input of the next layer and it is similarly trained and initial parameters are acquired for that layer. Until the parameters for each layer are initialized, the training process maintains. The overall output of the network is specified as the final activation vector. After this unsupervised pre-training stage of stacked layers, the entire network can then be fine-tuned in the opposite direction using backpropagation in this supervised learning phase.

3.2.2. Initialization of the number of layers and weights

There is no specific rule to determine the number of hidden layers and hidden units in each layer. Different factors influence the determination of the number of hidden layers and units, including the nature and size of the input images. Although some researchers have developed rules-of-thumb to determine the number of hidden units with respect to the number of input size, output size or training size, it is still not possible to determine the validity of any of these rules for generalization. In the proposed system the number of hidden layers and units were determined by the practical experiments. According to the test evaluation, the best result determined the number of hidden layers and units.

Optimization of the weights of autoencoders is also a challenging task. Large initial weights lead to falling into a local minima. On the other hand, small initial weights make the deep model infeasible. However, if the initial weights are already close to a good solution, the optimization techniques such as gradient descent, work well. Employing a similar "greedy layer-wise" learning algorithm to stacked autoencoders is an effective method to pre-train of the deep autoencoder [43] in the proposed method. In training phase, overfitting occurs when a model with high capacity fits the noise in the data instead of the underlying relationship. To prevent overfitting the general L2 regularization term (also called a weight decay term) was preferred.

3.2.3. Optimization of the autoencoder

In this study the optimization is done by minimizing the average reconstruction error:

$$\theta^*, \theta'^* = arg_{\theta,\theta'} min \frac{1}{m} \sum_{i=1}^{m} \left(x^i, x'^i\right)$$

$$\theta^*, \theta'^* = arg_{\theta,\theta'} min \frac{1}{m} \sum_{i=1}^{m} \left(x^i, g_\theta\left(f_\theta\left(x^i\right)\right)\right) \tag{8}$$

In Eq. (8) x is the input data, x' is the reconstruction, $g_\theta(h)$ is the decoder output, $h = f_\theta(x)$ is the encoder output and θ contains the weights W and the bias b [43].

Here $\mathcal{L}(\cdot)$ is a loss function such as mean squared errors:

$$\mathcal{L}\left(x, x'\right) = \|x - x'\|^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - x'_i)^2 \tag{9}$$

One of the common loss used with autoencoder is the cross-entropy:

$$\mathcal{L}\left(x, x'\right) = \sum_i x_i \log \frac{1}{x'_i} = -\sum_i x_i \log x'_i \tag{10}$$

Despite the mean squared errors loss is suitable for pre-training of autoencoders, the cross-entropy function was used in the final stacked autoencoder. A softmax layer was added at the top as a classification layer to train the classification model.

3.2.4. Regularization term (Weight decay term)

The most common form of regularization to prevent overfitting is to use L2 regularization. That is, the sum of the square of the weights W is added to the objective (loss function).

$$J_{weights} = \frac{1}{2} \lambda \| W \|_2^2 = \frac{\lambda}{2} \sum_{l=1}^{L} \sum_{i=1}^{N_l} \sum_{j=1}^{N_{l+1}} (W_{ji}^{(l)})^2 \tag{11}$$

In Eq. (11) λ is the parameter to control the strength of regularization, L is the number of hidden layers and N_l is the number of neurons in layer.

3.2.5. Sparsity specification for the sparse autoencoder

While the identity function is being learnt, a challenging structure of the data is discovered by bounding the number of hidden units and placing constraints on the network. But even when the number of hidden units is large, other constraints on the network can be imposed. If a sparsity constraint is imposed on the hidden units, the autoencoder can still discover

Fig. 4. Random samples from Yale dataset after applying Gaussian denoising with corruption level = 0.3.

the challenging structure of the data [28]. In this study the sparsity is done by adding a sparsity constraint on the neurons. $\hat{\rho}_j$ is the average activation of hidden unit j in Eq. (12).

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j\right]_i \tag{12}$$

In this study the constraint $\hat{\rho}_j = \rho$ was enforced approximately where ρ is a sparsity parameter and typically is a small value close to zero. To achieve this, an extra penalty term was added to our optimization objective that penalized $\hat{\rho}_j$ deviating significantly from ρ, this penalty term was based on Kullback–Leibler divergence given by Eq. (13) [28].

$$KL\left(\rho \| \hat{\rho}_j\right) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \left(\frac{1-\rho}{1-\hat{\rho}_j} \right) \tag{13}$$

The sparsity penalty term J_{sparse} can be given in the following equation (Eq. (14))

$$J_{sparse} = \beta \sum_{j=1}^{N} KL\left(\hat{\rho} \parallel \rho\right) \tag{14}$$

where β is a weight of sparsity, N is the number of hidden units and $KL(\hat{\rho} \parallel \rho)$ is the Kullback–Leibler divergence as defined above.

Let $\mathcal{L}(x, x')$ be the loss function as mentioned previously (Eqs. (9) and (10)), J_{weight} is the regularization term (Eq. (11)) and J_{sparse} is the sparsity penalty (Eq. (14)), then the overall cost function can be written as:

$$J_{cost} = \mathcal{L}\left(x, x'\right) + J_{weight} + J_{sparse} \tag{15}$$

3.2.6. Denoising the autoencoder

In the proposed method instead of using the input x in the autoencoder, a partially corrupted copy of input $\tilde{x}$ is used to obtain a good representation. However, training still tries to reconstruct the original input. Thus, Eq. (1) above is rewritten as:

$$z = f_\theta(\tilde{x}) = \sigma(W\tilde{x} + b) \tag{16}$$

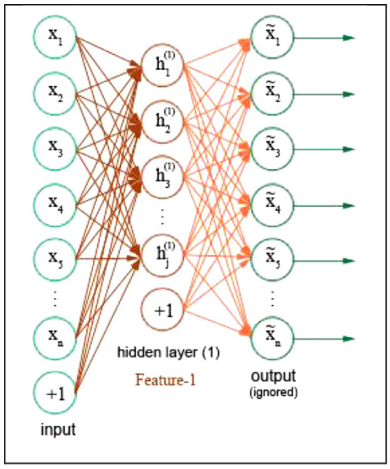
The input can be corrupted in many ways; the simple way is done by setting a certain percentage of random pixels to zero or by adding random Gaussian noise to these pixels. This percentage is called the corruption level and usually it is selected between 0 and 0.5.

The experiments demonstrate that the denoising autoencoders can improve the generalization performance of the network and able to learn Gabor-like edge detectors. Fig. 4 shows some samples from Yale dataset after applying Gaussian denoising with a corruption level of 0.3.

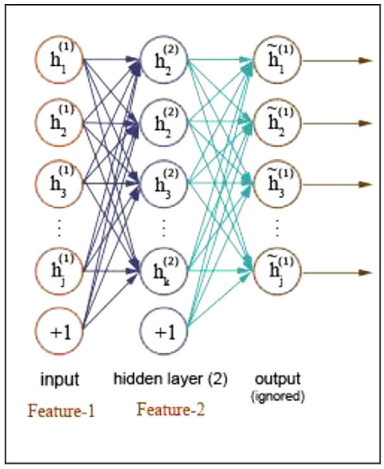
3.2.7. Fine-tuning of the parameters

The autoencoder involves many parameters to be tuned including layers number, neurons, weight decay term, sparsity and weights. The weights were tuned as mentioned in Section 3.2.2, while the grid search strategy was used to tune the other parameters. It was also coupled with cross validation technique for evaluation.

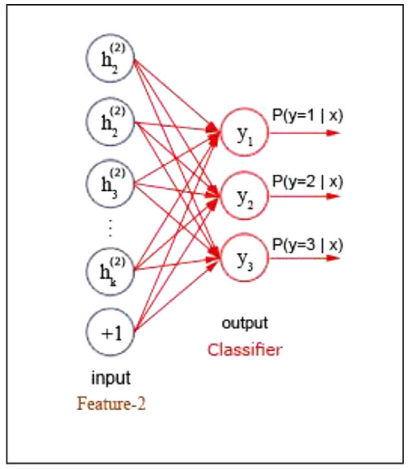
Fine-tune the stacked autoencoder: Fine-tuning step was applied to adjust the parameters of the stacked autoencoder to significantly improve the performance. In this study fine-tuning was done by training the pre-trained layers as a single



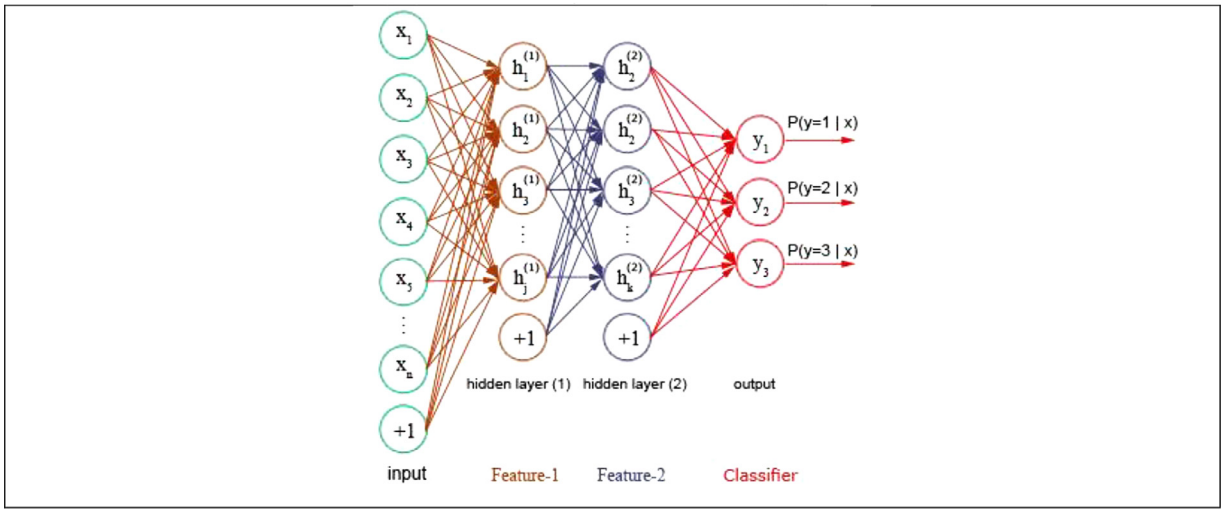
(a) First denoising sparse autoencoder to learn primary features $h^{(1)}$ on the raw input.



(b) Second denoising sparse autoencoder uses the primary features $h^{(1)}$ as the "raw input" to learn secondary features $h^{(2)}$.



(c) Classifier layer: Uses secondary features $h^{(2)}$ as "raw input" to a classifier, training it to map secondary features to predict labels.



(d) Full stacked autoencoder: Combines all three layers together to form a stacked autoencoder with 2 hidden layers and a final classifier layer

Fig. 5. The proposed stacked autoencoder with 2 pre-training autoencoders and 3 output classes.

model using supervised learning mechanism to update the weights. The backpropagation algorithm was used to get the gradients to adjust the weights of whole network.

3.2.8. The deep stacked denoising sparse autoencoder

In the proposed method, denoising and sparse autoencoders were stacked together to form a robust deep model called Deep Stacked Denoising Sparse Autoencoder (DSDSA). Each denoising sparse autoencoder took its input from the activation of the previous layer and pre-trained independently. The goal of pre-training is to optimize some similar objective to put the parameters of all the layers in a region of parameter space layer wise. It is seen that stacked autoencoders can extract useful features by a series of learning stages.

Stacked autoencoder training followed the greedy layer-wise strategy which is outlined in Section 3.2.1. The overall output after fine-tuning served as an input to the classification algorithm and both (multi-class) SVM and Softmax classifiers were used in this study. The experimental results indicate that the efficiency is close when using both methods, with a little preference for the SVM predominantly. Fig. 5 shows the hierarchy and procedure of DSDSA system with two hidden layers and three output classes.

Table 1
The detection results on four datasets.

Dataset	Number of faces	Correctly detected (TP)	Undetected faces (FN)	False detection (FP)
Yale	165	165	0	0
ORL	400	391	9	0
Caltech	450	447	3	74
PubFig15	750	750	0	0

4. Experimental results

The (DSDSA) system has been tested on four known face recognition datasets, which are PubFig15,[1] ORL,[2] Yale,[3] Caltech[4] databases.

ORL: The ORL face database, by the Olivetti Research Laboratory in Cambridge, UK.

- A total of 400 images for 40 individuals.
- There are 10 different images for each individual.
- Almost frontal, up-right and with dark background.
- Contains different expressions, varying lighting and occlusions (sun glasses).

Yale: The Yale Face Database:

- A total of 165 images for 40 individuals.
- There are 11 different images for each individual.
- Contains different expressions, varying lighting and occlusions (sun glasses).

Caltech 1999: Collected by California Institute of Technology

- A total of 445 images for 26 individuals.
- There are different number of images for each individual.
- Almost frontal, with unconstrained backgrounds.
- Contains different expressions, varying lighting and occlusions.

PubFig15: A chosen subset of PubFig database which introduced by Kumar et al. [56]. The PubFig dataset contains 200 subjects and various numbers of images for different subjects.

- A total of 750 samples for 15 individuals.
- There are 50 different images for each individual.
- The images may have out-of-plane as well as in-plane rotation.
- Contains non-frontal, with unconstrained backgrounds.
- The alignment is not perfect but the majority of the face area is visible.
- Contains different expressions, varying in lighting and occlusions.

4.1. Face Pre-processing experiments

Fig. 6 demonstrates some face detection and localization results on Yale database. Table 1 shows the true positive, false negative and false positive detection numbers using the related datasets. For normalization the faces were automatically cropped to size 32×32 pixels, aligned (centered with respect to eyes) and converted to grayscale. Fig. 7 shows the detected, aligned and cropped faces from ORL and Caltech datasets.

4.2. Face recognition experiments

The efficiency of the (DSDSA) system was measured using the datasets mentioned above. Accuracy is the ratio of faces that are correctly classified to the total number of faces in test dataset. Below are the main settings used during the training of DSDSA:

- The input image size is of 32×32 pixels. Thus, input dimensionality is 1024.

[1] PubFig: http://www.cs.columbia.edu/CAVE/databases/pubfig/ PubFig83: http://vision.seas.harvard.edu/pubfig83/A subset of 750 samples (50 images for 15 subjects) were chosen from available PubFig83, and was given a name of Pub Fig 15.
[2] ORL: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[3] Yale: http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html2.
[4] Caltech1999: http://www.vision.caltech.edu/html-files/archive.html.

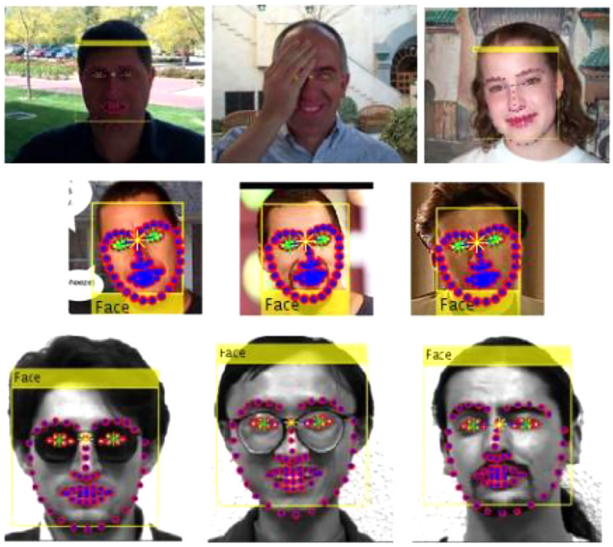


Fig. 6. Some face detection and localization results on Yale database.

Fig. 7. Aligning and cropping applied to ORL (left) with [0.25] margin, and Caltech1999 (right) with [0.3] margin.

- Number of hidden layers and nodes in each layer is a hyperparameter that varies from one database to another; however, experiments show that using two hidden layers has given a sufficiently good performance taking into account the cost in the time complexity. However, hidden units have different efficiency with different datasets, the next part shows the effect of these differences.
- The corruption levels for denoising were 0.1 and 0.5. Other options were tested as well.
- Regularization parameter (L2 normalization) λ was set to 3e-4.
- The sigmoid activation function was used for both input and output units in pre-training autoencoders, Softmax was used for output activation during fine-tuning.
- Weights initialization: Weights should be initialized randomly, it is important not to give similar values or zeros, otherwise it will learn the same activation, which is not useful. Common method is to sample a uniform $(-r, r)$ where $r = 4\sqrt{6/(\mathrm{U_{in}} + \mathrm{U_{out}})}$ and $\mathrm{U_{in}} + \mathrm{U_{out}}$ is the number of inputs/outputs units. Bias was initialized with zeros.

Table 2 shows the parameters (k-fold numbers and two hidden layer neuron numbers) producing highest recognition accuracy on different face datasets using Softmax and SVM classifiers. Both for training the entire network and the classification (output) layer, nested cross validation is used. Tables 3 and 4 demonstrate different performance metrics using Softmax and SVM classifiers, respectively.

Fig. 8 shows the recognition loss (error) over iterations of the proposed (DSDSA) on varying datasets. According to the demonstrated results the effect of the dataset size is clear.

The proposed DSDSA was compared against some recent methods based on deep learning and non-deep learning methods. The top eight methods are based on deep learning in Table 5 which are the deep sparse autoencoder by Zhang et al. [37] symbolized with SAE, spatial domain sparse representation with autoencoders [38] symbolized with SDSRA,

Table 2
The recognition accuracy on the Yale, ORL, Caltech and Pubfig15.

Dataset	Net. k-folds	Class. k-folds	Neurons	Acc.% (Softmax)	Acc.% (SVM)
Yale	10	5	[121,121]	97.57	98.16
ORL	5	5	[529,529]	97.50	97.50
Caltech	8	5	[121,121]	97.76	98.21
PubFig15	5	2	[121,64]	95.20	95.07

Table 3
Performance metrics using Softmax classifier.

Dataset	Acc.%	Sensitivity	Specificity	Precision	Recall	F_1 score	Kappa
Yale	97.57	0.9767	0.9983	0.9678	0.9767	0.9698	0.8668
ORL	97.50	0.9750	0.9994	0.9817	0.9750	0.9737	0.9500
Caltech	97.76	0.9598	0.9991	0.9500	0.9598	0.9534	0.9154
PubFig15	95.20	0.9520	0.9966	0.9567	0.9520	0.9518	0.8671

Table 4
Performance metrics using SVM classifier.

Dataset	Acc.%	Sensitivity	Specificity	Precision	Recall	F_1 score	Kappa
Yale	98.16	0.9833	0.9987	0.9778	0.9833	0.9787	0.8668
ORL	97.50	0.9750	0.9994	0.9817	0.9750	0.9737	0.9500
Caltech	98.21	0.9650	0.9992	0.9569	0.9650	0.9596	0.9154
PubFig15	95.07	0.9507	0.9965	0.9549	0.9507	0.9506	0.8671

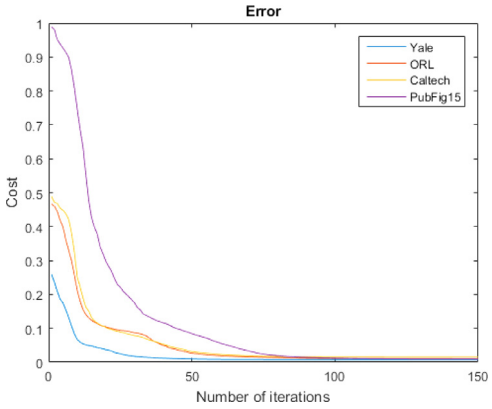


Fig. 8. The recognition loss of the proposed (DSDSA) on different datasets after 150 iterations.

Gabor features in deep belief networks [39] symbolized with GFDBN, deep convolutional neural networks (CNN) symbolized with DCNN1 [40] and DCNN2 [42], hybrid local binary pattern and deep CNN symbolized with LBPCNN [41] and finally two state-of-the-art systems in face recognition field FaceNet [35] and DeepFace [26] using deep CNN as well.

The non-deep learning methods listed for comparison in Table 5 are Laplacianfaces [57], systolic architecture based on principal component neural network symbolized with PCNN [58], principal component analysis symbolized with PCA [3], memory-based face recognition algorithm which matches a reduced resolution version of the image against a database of previously collected exemplars using a similarity metric symbolized with Arena [59], local binary pattern based method symbolized with LBP [8], Gabor filters based method symbolized with Gabor [9], histograms of oriented gradients based method symbolized with HOG [10], curvelet transform based method symbolized with CT [11], nearest neighbor based method symbolized with NN [13] and the method using mean sequence sparse representation-based classification symbolized with MSSRC [60]. According to the comparative results in Table 5, the deep learning methods achieve recognition accuracy higher than 90% while non-deep learning methods can not exceed the accuracy of 90% in some referred studies [57,58,60]. Therefore it is explicit that deep learning is an effective technology for the face recognition challenge.

It is seen that most of the algorithms are based on CNNs in the studies implemented so far in deep learning area. The two state-of-the-art systems related to face recognition using CNN are FaceNet [35] and DeepFace [26]. In this study we implement DSDSA instead of using deep CNN. CNN is designed to handle supervised learning which depends on labeled data. It mostly needs large data for training; for example DeepFace [26] was trained with 4 million images and FaceNet [35] was trained with 200 million images. DSDSA uses a different network architecture. To extract more robust and discriminative

Table 5
Comparison results between the proposed DSDSA and other deep learning and non-deep learning methods in the same field.

Method	Dataset	Acc.%
DeepFace	LFW [61]	97.35
FaceNet	LFW [61]	98.87
SAE	Yale	94.67
SDSRA	ORL	93.99
GFDBN	ORL	94.98
DCNN1	JAFFE [62]	94.21
LBPCNN	Faces94 [63]	93.60
DCNN2	Own dataset	97.60
Laplacianfaces	Yale	88.70
PCNN	Yale	85.00
PCA	ORL	96.50
Arena	ORL	97.10
LBP	FERET [64]	93.00
Gabor	Yale	93.33
HOG	Yale	95.5
CT	JAFFE [62]	97.2
NN	ORL	93.84
MSSRC	AFLW [65]	80.80
DSDSA	**Yale**	**98.16**
DSDSA	**ORL**	**98.00**

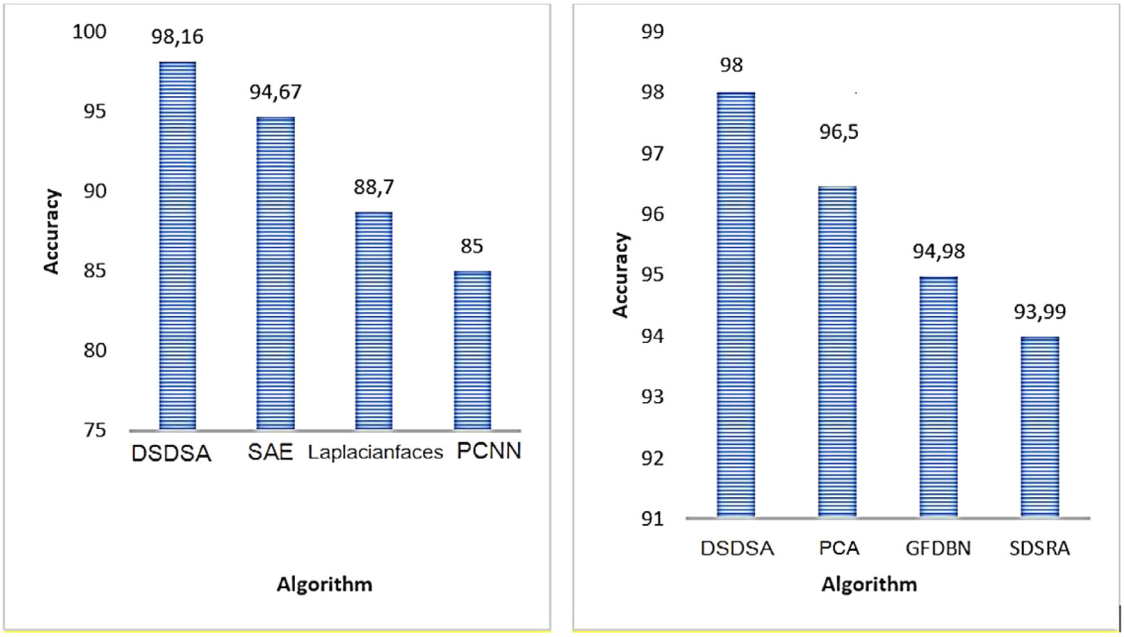


Fig. 9. Comparative analysis of different approaches on Yale (left) and ORL (right).

features from variances of face images, an optimized sparse autoencoder is built in addition to denoising process. The denoising deep sparse autoencoder which acts as a robust classifier due to powerful feature learning ability is implemented. DSDSA provides a learning algorithm investigating unlabeled data. It is simple, intuitively understandable and easy to implement. Autoencoders are more general because they specify nothing about the network topology. The additional reason for choosing autoencoders in this study is that they can be stacked on top of each other to form an effective feature extraction method while maintaining its simplicity. As listed in Table 5, FaceNet and DeepFace achieve a recognition accuracy of 98.87% and 97.35%, respectively. On the other hand our proposed DSDSA system has an accuracy of 98.16% and 98.00% with the databases Yale and ORL respectively. It can be observed that we achieve comparable results to other deep CNN and the state-of-the-art methods (FaceNet [35] and DeepFace [26]) using much less data even when the training data come from a different database. Again, this is a conclusion that may be applicable to many other tasks.

Fig. 9 demonstrates a comparative analysis of different approaches on Yale and ORL respectively. In SAE the system is implemented using two hidden layers with [200,100] and 1024 hidden units. The proposed DSDSA achieved the mentioned

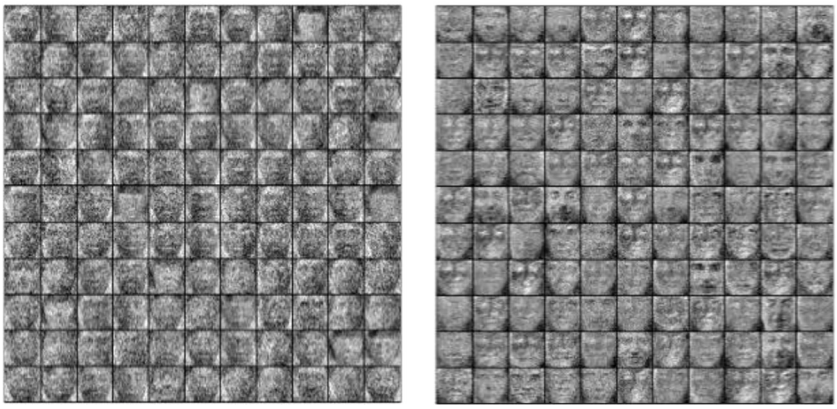


Fig. 10. The learnt filters after training the first layer/autoencoder (Yale-left, ORL-right).

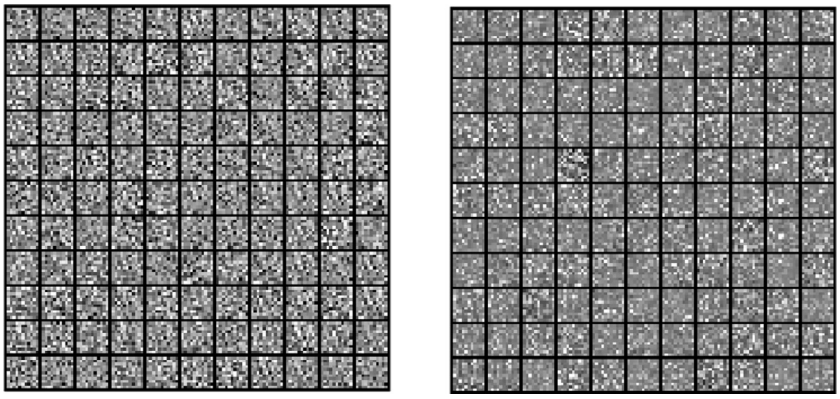


Fig. 11. The result after training the second layer/autoencoder (Yale-left, ORL-right).

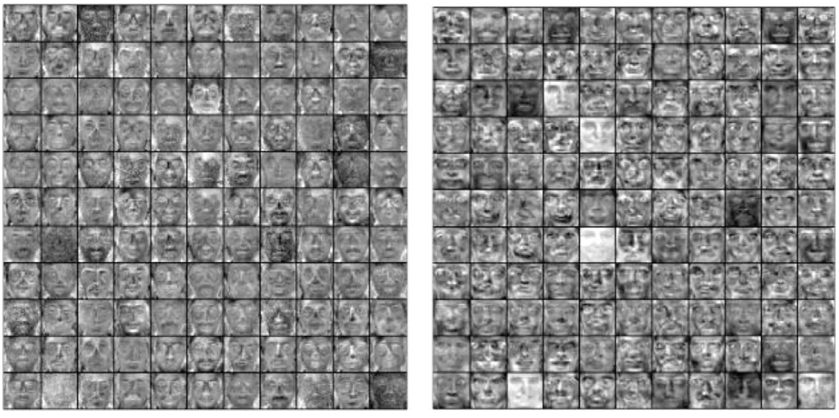


Fig. 12. The DSDSA reconstruction results (Yale-left, ORL-right).

results with the input size of 1024 and two hidden layers with [121,121] hidden units for Yale and [529,529] hidden units for ORL dataset.

Figs. 10 and 11 show the learnt filter after pre-training the first and second denoising sparse autoencoder with high corruption (0.5) using Yale and ORL respectively. Figs. 12 and 13 show the stacked autoencoder's reconstruction result and the confusion matrix for PubFig15 dataset, respectively.

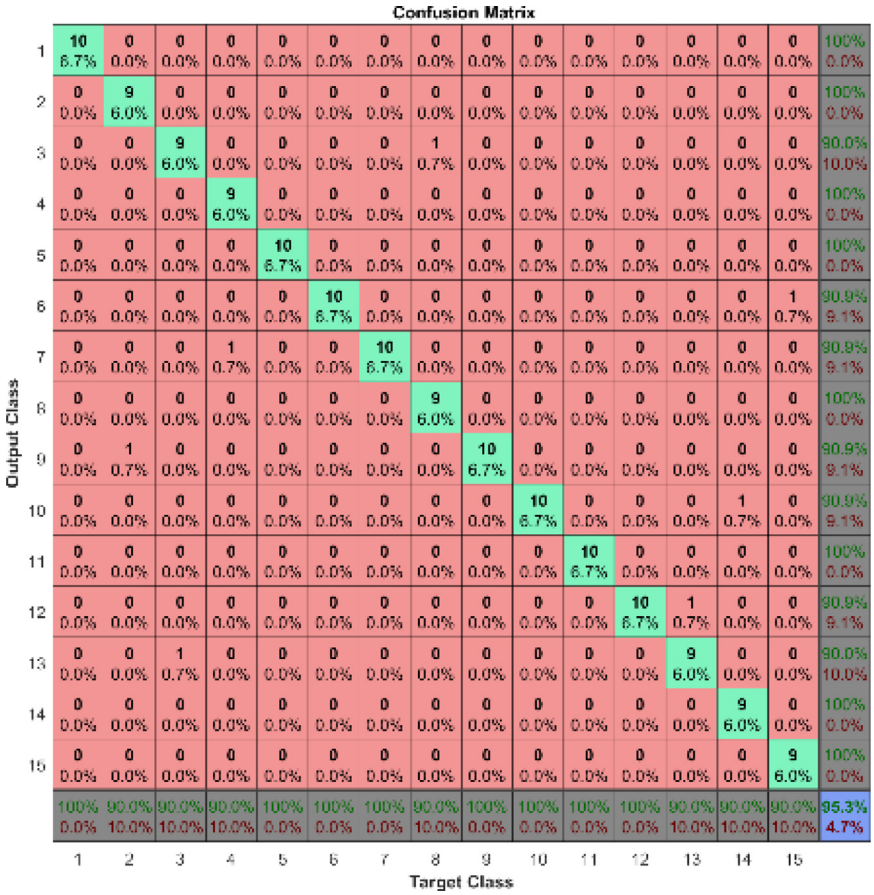


Fig. 13. Confusion matrix for PubFig15 dataset after 150 iterations with [121,121] hidden layers and neurons with 0.1 corruption and 0.5 sparsity parameter.

Table 6
The effect of the number of hidden layers on time and accuracy ['PubFig15', with neuron numbers 121 for each layer, corruption: 0.5, sparse: 0.5, iterations: 200].

Hidden layers	Softmax Acc.%	SVM Acc.%	Time
2*x* [121]	94.00	94.00	2.71
3*x* [121]	91.33	91.33	3.02
4*x* [121]	92.00	92.00	3.36
5*x* [121]	90.00	90.00	3.63
6*x* [121]	88.67	88.00	3.94
7*x* [121]	84.00	86.00	4.23
8*x* [121]	82.00	84.00	4.55
9*x* [121]	82.00	83.33	4.81
10*x* [121]	87.33	86.67	5.17

4.3. The impact of parameters

Parameters play a key role in the determination of the evaluation result. Different settings of important parameters have been experimentally evaluated in 'PubFig15' to show the impact of parameters. It should be noted that the best results are not the aim here, but only to note the effect of these parameters. Table 6 and Fig. 14 explore the effect of the number of hidden layers on performance. More hidden layers mean deeper network, this also will cost more training time and sometimes more iteration to get good results.

Table 7 explores the performance of DSDSA with various number of hidden units' sizes (neuron number). The result improves that the number of neurons has an important role on the network architecture with note that there is no steady or general rule determines the suitable number of hidden units for a specific task.

Fig. 15 draws the relation between the number of hidden units and the cost of time. Obviously, the higher the number of neurons, the greater the cost of time.

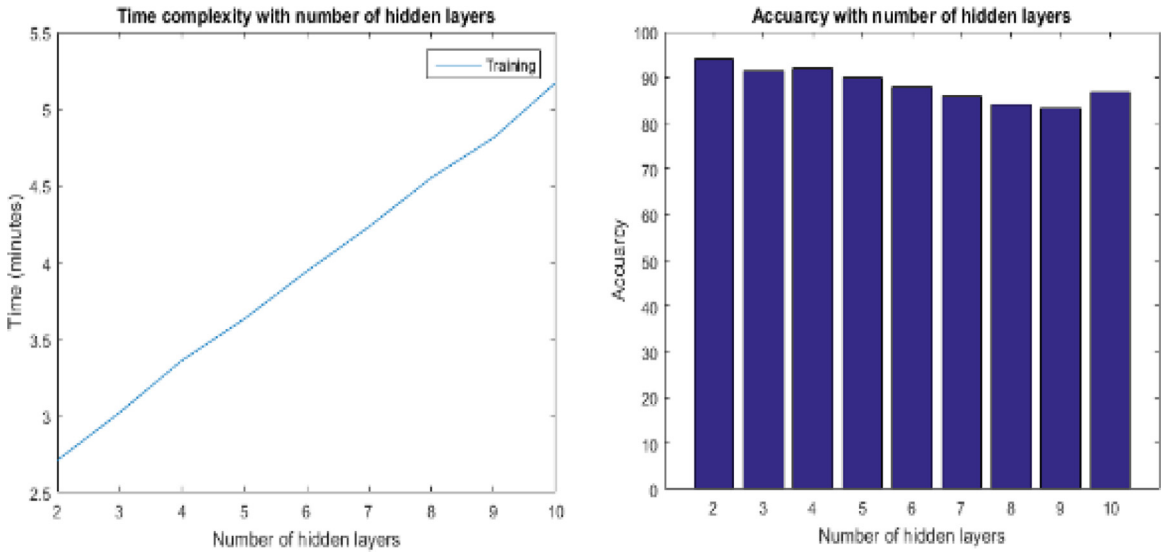


Fig. 14. The effect of the number of hidden layers on time and accuracy.

Table 7
The performance of the proposed DSDSA under different number of neurons.

Hidden units' sizes	Softmax Acc.%	SVM Acc.%	Time
[64,64]	91.33	92.67	0.89
[121,121]	94.67	95.33	1.66
[200,200]	96.00	94.67	2.86
[441,441]	94.67	93.33	7.35
[667,667]	92.00	91.33	13.27

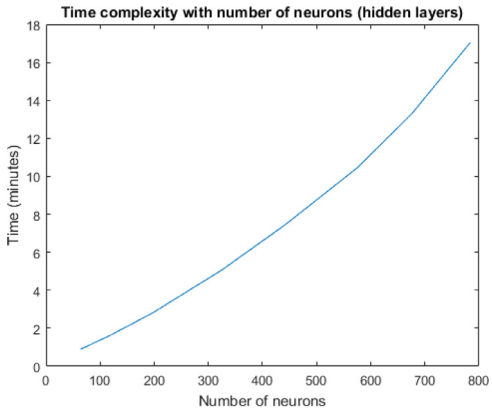


Fig. 15. Time complexity with number of neurons.

Denoising term is determined by corruption level, which is usually estimated between 0 and 0.5. Table 8 shows the impacts of denoising level on the performance. The results show that the autoencoder has the ability to restore a good reconstruction from corrupted input version even with a high corruption level.

Finally, the sparsity was determined by two parameters, the sparsity penalty β and the sparsity parameter. Table 9 shows the effect of changing the sparsity parameter while beta was chosen to as 3. The results show how the selection of the correct sparsity value effects the performance.

4.4. Discussion

In this study we propose a complete and robust face recognition system that also includes face detection and landmark localization and can deal with various face datasets containing unconstrained faces with variability in pose, expression,

Table 8
The effect of denoising level on performance.

Corruption levels	Softmax Acc.%	SVM Acc.%
[0.0]	93.33	93.33
[0.1]	95.33	95.33
[0.2]	92.67	92.00
[0.3]	94.67	95.33
[0.5]	94.67	95.33

Table 9
The effect of sparsity parameter on accuracy.

Sparsity target	Softmax Acc.%	SVM Acc.%
[0.1]	94.67	94.00
[0.2]	94.00	93.33
[0.5]	95.33	95.33

illumination and scene. In this study a sparsity constraint is specified to make the trained model lie on the balance point where the model is neither underfitting nor overfitting. The sparsity term also dramatically reduces the computational cost.

Experimental results show that the proposed DSDSA has achieved satisfactory results on face recognition after examination on different databases. DSDSA proved its ability to generalize and demonstrated the usefulness of sparsity and denoising to overcome the overfitting problem and improve performance efficiency. One of the additional pros of the proposed system is that pre-training that is provided in the autoencoders allows the usage of huge amounts of available input data. One key advantage of the proposed DSDSA is its ability to learn from fewer training data unlike existing systems that rely on a huge amount of data [26,35].

In this study we implement DSDSA instead of using deep CNN. CNN is designed to handle supervised learning which depends on labeled data. It mostly needs large data for training, for example DeepFace [26] was trained with 4 million images and FaceNet [35] was trained with 200 million images. DSDSA uses a different network architecture. To extract more robust and discriminative features from variances of face images, an optimized sparse autoencoder is built in addition to denoising process. The denoising deep sparse autoencoder which acts as a robust classifier due to powerful feature learning ability is implemented. DSDSA provides a learning algorithm which is simple, intuitively understandable and easy to implement. Autoencoders are more general because they specify nothing about the network topology. The additional reason for choosing autoencoders in this study is that they can be stacked on top of each other to form an effective feature extraction method while maintaining its simplicity.

It can be observed that we achieve comparable results to other deep CNN and the state-of-the art methods (FaceNet [35] and DeepFace [26]) using much less data even when the training data come from a different database. This is a conclusion that the proposed method may be applicable to many other tasks.

However, adjusting of network parameters needs further improvement to achieve higher results. Since the system uses the whole entire image (after normalizing) in the training, the image size can be considered as a limitation. Regarding to the face recognition issue, the system still needs more effort in the normalization and aligning steps to address the problem of pose variation. The development of system performance should take into account the effect of using alternative activation functions and optimization methods such as stochastic gradient descent. Also, regularization techniques such as dropout should be investigated.

5. Conclusions

In this paper a face recognition system based on deep stacked denoising and sparse autoencoders called (DSDSA) is proposed. The algorithm extracts robust features by supervised learning through an optimized and denoised deep sparse autoencoder. The experimental results on four well-known face image databases (ORL, Yale, Caltech and Pubfig) show that the proposed DSDSA is efficient to deal with face recognition tasks and achieves good results. Many aspects such as denoising, sparsity, weights initialization and more details are still the subject of research and development area to reach more robust results.

References

[1] A.K. Jain, S.Z. Li, Handbook of Face Recognition, Springer, New York, 2011.
[2] S. Liao, A.K. Jain, S.Z. Li, A fast and accurate unconstrained face detector, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2016) 211–223.
[3] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 711–720.
[4] M.A. Turk, A.P. Pentland, Face recognition using eigenfaces, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR'91, 1991, pp. 586–591.
[5] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[6] B. Schölkopf, A. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.

[7] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K.R. Mullers, Fisher discriminant analysis with kernels, in: Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop, 1999, pp. 41–48.
[8] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2037–2041.
[9] W.R. Boukabou, L. Ghouti, A. Bouridane, Face recognition using a Gabor filter bank approach, in: First NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, 2006, pp. 465–468.
[10] O. Déniz, G. Bueno, J. Salido, F. De la Torre, Face recognition using histograms of oriented gradients, Pattern Recognit. Lett. 32 (2011) 1598–1603.
[11] S. Biswas, J. Sil, An efficient face recognition method using contourlet and curvelet transform, J.King Saud Univ. Comput. Inf. Sci. (2017), doi:10.1016/j.jksuci.2017.10.010.
[12] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[13] D. Ren, M. Hui, N. Hu, T. Zhan, A weighted sparse neighbor representation based on Gaussian kernel function to face recognition, Optik 167 (2018) 7–14.
[14] J. Ho, M.H. Yang, J. Lim, K.C. Lee, D. Kriegman, Clustering appearances of objects under varying illumination conditions, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, pp. 1–11.
[15] M. Guillaumin, J. Verbeek, C. Schmid, Is that you? Metric learning approaches for face identification, in: Proceedings of the IEEE Twelfth International Conference on Computer Vision, 2009, September, pp. 498–505.
[16] C. Huang, S. Zhu, K. Yu, 2012, Large scale strongly supervised ensemble metric learning with applications to face verification and retrieval. arXiv:1212.6094.
[17] R.G. Cinbis, J. Verbeek, C. Schmid, Unsupervised metric learning for face identification in TV video, in: Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV), 2011, pp. 1559–1566.
[18] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[19] L. Zhang, M. Yang, X. Feng, Sparse representation or collaborative representation: which helps face recognition? in: Proceedings of the IEEE in Computer vision (ICCV), 2011, pp. 471–478.
[20] S. Gao, Y. Zhang, K. Jia, J. Lu, Y. Zhang, Single sample face recognition via learning deep supervised autoencoders, IEEE Trans. Inf. Forensics and Secur. 10 (10) (2015) 2108–2118.
[21] M. Yang, L. Zhang, Gabor feature based sparse representation for face recognition with Gabor occlusion dictionary, in: Proceedings of the European Conference on Computer Vision, Berlin, Heidelberg, Springer, 2010, pp. 448–461.
[22] C.Y. Lu, H. Min, J. Gui, L. Zhu, Y.K. Lei, Face recognition via weighted sparse representation, J. Vis. Commun. Image Represent. 24 (2) (2013) 111–116.
[23] R. Min, J.L. Dugelay, Improved combination of LBP and sparse representation based classification (SRC) for face recognition, in: Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), 2011, pp. 1–6.
[24] J. Yin, Z. Liu, Z. Jin, W. Yang, Kernel sparse representation based classification, Neurocomputing 77 (1) (2012) 120–128.
[25] G.B. Huang, H. Lee, E. Learned-Miller, Learning hierarchical representations for face verification with convolutional deep belief networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2012, pp. 2518–2525.
[26] Y. Taigman, M. Yang, M.A. Ranzato, L. Wolf, Deepface: closing the gap to human-level performance in face verification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1701–1708.
[27] Y. Sun, X. Wang, X. Tang, Hybrid deep learning for face verification, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1489–1496.
[28] Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1891–1898.
[29] N. Zeng, Z. Wang, H. Zhang, W. Liu, F.E. Alsaadi, Deep belief networks for quantitative analysis of a gold immunochromatographic strip, Cogn. Comput. 8 (4) (2016) 684–692.
[30] Y. Sun, X. Wang, X. Tang, 2015, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2892–2900.
[31] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[32] Y. Sun, D. Liang, X. Wang, X. Tang, 2015, Deepid3: face recognition with very deep neural networks. arXiv:1502.00873.
[33] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 05), 1, 2005, pp. 539–546.
[34] Z. Zhu, P. Luo, X. Wang, X. Tang, 2014, Recover canonical-view faces in the wild with deep neural networks. arXiv:1404.3543.
[35] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
[36] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, in: Proceedings of the BMVC, 2015, September, p. 6.
[37] Z. Zhang, J. Li, R. Zhu, Deep neural network for face recognition based on sparse autoencoder, in: Proceedings of the Eighth International Congress on Image and Signal Processing (CISP), 2015, pp. 594–598.
[38] S. Biswas, J. Sil, S.P. Maity, On prediction error compressive sensing image reconstruction for face recognition, Comput. Electr. Eng. (2017), doi:10.1016/j.compeleceng.2017.11.009.
[39] Y. Chen, T. Huang, H. Liu, D. Zhana, Multi-pose face ensemble classification aided by Gabor features and deep belief nets, Optik - Int. J. Light Electron Opt. 127 (2) (2016) 946–954.
[40] N. Jain, K. Shishir, A. Kumar, P. Shamsolmoali, M. Zareapoor, Hybrid deep neural networks for face emotion recognition, Pattern Recognit. Lett. (2018), doi:10.1016/j.patrec.2018.04.010.
[41] A. Vinay, A. Gupta, A. Bharadwaj, A. Srinivasan, K.N.B. Murthy, S. Natarajan, Deep learning on binary patterns for face recognition, Proc. Comput. Sci. 132 (2018) 76–83.
[42] X. Luo, R. Shen, J. Hu, J. Deng, L. Hu, Q. Guan, A Deep convolution neural network model for vehicle recognition and face recognition, Proc. Comput. Sci. 107 (2017) 715–720.
[43] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[44] M. Kan, S. Shan, H. Chang, X. Chen, Stacked progressive auto-encoders for face recognition across poses, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1883–1890.
[45] J. Zhang, S. Shan, M. Kan, X. Chen, Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment, in: Proceedings of the European Conference on Computer Vision, Cham, Springer, 2014, pp. 1–16.
[46] S. Gao, I.W.H. Tsang, L.T. Chia, Kernel sparse representation for image classification and face recognition, in: Proceedings of the European Conference on Computer Vision, Berlin, 2010, pp. 1–14.
[47] C. Ding, D. Tao, Robust face recognition via multimodal deep face representation, IEEE Trans. Multimed. 17 (11) (2015) 2049–2058.
[48] F. Li, X. Gao, L. Wang, Face recognition based on deep autoencoder networks with dropout, Adv. Intell. Syst. Res. 132 (2017) 243–246.
[49] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, A.M. Dobaie, Facial expression recognition via learning deep sparse autoencoders, Neurocomputing 273 (2018) 643–649.
[50] Y. Bengio, Learning deep architectures for AI, Found. Trends® Mach. Learn. 2 (1) (2009) 1–127.
[51] C. Gravelines, Deep learning via stacked sparse autoencoders for automated voxel-wise brain parcellation based on functional connectivity, in: Proceedings of the Electronic Thesis and Dissertation Repository, 2014.
[52] P. Viola, M.J. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2) (2004) 137–154.

[53] X. Yu, J. Huang, S. Zhang, W. Yan, D.N. Metaxas, Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model, in: Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1944–1951.
[54] T. Baltrusaitis, P. Robinson, L.P. Morency, Constrained local neural fields for robust facial landmark detection in the wild, in: Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops (ICCVW), Sydney, 2013, pp. 354–361.
[55] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[56] N. Kumar, C.A. Berg, P.N. Belhumeur, S.K. Nayar, Attribute and simile classifiers for face verification, in: Proceedings of the International Conference on Computer Vision (ICCV), 2009.
[57] X. He, S. Yan, Y. Hu, P. Niyogi, H.J. Zhang, Face recognition using laplacianfaces, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (2005) 328–340.
[58] N. Sudha, A.R. Mohan, P.K. Meher, A self-configurable systolic architecture for face recognition system based on principal component neural network, IEEE Trans. Circuits Syst. Video Technol. 21 (8) (2011) 1071–1084.
[59] T. Sim, R. Sukthankar, M. Mullin, S. Baluja, 1970, High-Performance Memory-based Face Recognition for Visitor Identification.
[60] E.G. Ortiz, A. Wright, M. Shah, , Face recognition in movie trailers via mean sequence sparse representation-based classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2013, pp. 3531–3538.
[61] G.B. Huang, M. Ramesh, T. Berg, E. Learned-miller, Labeled faces in the wild: a database for studying face recognition in unconstrained environments, in: Proceedings of the ECCV Workshop on Faces in Real-life Images, 2008, 2008.
[62] M.J. Lyons, J. Budynek, S. Akamatsu, Automatic classification of single facial images, IEEE Trans. Pattern Anal. Mach. Intell. 21 (12) (1999) 1357–1362.
[63] N.C. Ebner, M. Riediger, U. Lindenberger, FACES—A database of facial expressions in young, middle-aged, and older women and men: development and validation, Behav. Res. Methods 42 (1) (2010) 351–362.
[64] J.R. Beveridge, D. Bolme, B.A. Draper, M. Teixeira, The CSU face identification evaluation system: its purpose features and structure, Mach. Vis. Appl. 16 (2) (2005) 128–138.
[65] M. Koestinger, P. Wohlhart, P.M. Roth, H. Bischof, Annotated facial landmarks in the wild: a large-scale, real-world database for facial landmark localization, in: Proceedings of the IEEE International Workshop on Benchmarking Facial Image Analysis Technologies, 2011, pp. 2144–2151.