Appling both hybrid RBM and DCNNs to FR low-resolution

By julian supardi

WORD COUNT	6993	TIME SUBMITTED	10-MAY-2023 09:56AM
		PAPER ID	99538884

Appling both hybrid restricted Boltzmann machine and deep convolution neural networks to lowresolution face image recognition

Shi-Jinn Horng, Julian Supardi, and Tianrui Li

Abstract

Due to the difficulty of finding the specific features of faces, in computer vision, low-resolution face image recognition is one of the challenging problems and the accuracy of recognition is still quite low. We were trying to solve this problem using deep learning techniques. Two major parts are used for the proposed method; first the restricted Boltzmann machine is used to preprocess the face images, then the deep convolution neural network is used to do classification. The data set was combined from the Georgia Institute of Technology, Aleix Martinez, and Robert Benavente. Based on this combined data, we conducted the training and testing processes. The proposed method is the first method that combines restricted Boltzmann machine and deep convolution neural networks to do low-resolution face image recognition. From the experimental results, compared to existing methods, the proposed method greatly improves the accuracy of recognition. The proposed method is shown in Figure 3.

Index Terms Low Resolution Face Recognition, Restricted Boltzmann Machine, Deep Learning, Deep Convolution Neural Network, Gaussian filter

1. Introduction

Face recognition is an important and difficult subject in the fields of image processing and computer vision [1]. Even though it has been studied for more than 30 years, face recognition is still not as good as it could be and is still the subject of study [2]. In fact, some methods are very accurate and have been used in a lot of different areas of science and technology. Even so, face recognition methods still needed to be improved because there are many things that can lead to false recognition, such as low lighting, a different orientation, a twisted expression, etc.

One problem with picture processing and pattern recognition is low-resolution face images. Face recognition is especially hard because the difference between pixel values in the face area is not very big. This makes it hard to find features on the face picture. Most of the time, a low-resolution image comes from video footage, an enlarged image, a picture taken with a camera that is out of focus or blurry, the result of a scan, or something similar. Figure 1 shows a picture of a face with a low quality. Some good ways to deal with this problem have been written about [4, 5, 6, 7, 8, and 9]. In general, the way the current methods work is by rebuilding the low-resolution picture into a high-resolution image. This is an easy and simple way to fix the problem. But each of these ways has its own problems that haven't solved the low-resolution face problem. Some of the problems mentioned in [10] and [11] still need to be looked into more.

This work was supported in part by the Ministry of Science and Technology under contract numbers [illegible]2221-E-011 -149 -MY2, 106-3114-E-011 -008 -, and it was also partially supported by One [illegible]dred Talents Program 2012, Sichuan Province.

[illegible]upardi is with the Department of Computer Science and Information Engineering, National Taiwan [illegible]versity of Science and Technology, Taipei, 10607, Taiwan, ROC (email: julian@unsri.ac.id)
Shi-Jinn Horng is with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, 10607, Taiwan, ROC (contact author, tel.: 886-2-27376700), and is also with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China, (e-mail:horngsj@yahoo.com.tw)
Tianrui Li is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China (e-mail: trli@swjtu.edu.cn)

Figure 1. A low resolution face image. One is sitting far away from a camera. Face area becomes so small and it is hard to find the face. Enlarge the size of the face area is not a workable solution as it will decrease the resolution of the image, and make it more difficult to be recognized. (Reprint from https://www.youtube.com/watch?v=L5xKoGj4K4U)

Face recognition is a type of recognizing patterns. Because of this, a neural network is a great tool to use [12][13][14][15][16][17][18][19]. This is because the neural network can learn to gain information so that it can solve problems the way humans do. The method that is used the most is the Multilayer Perceptron (MLP). But MLP has trouble when there are a lot of sources. LeCun Y, et al. [21] made Convolutional Neural Networks (CNN) by building on what Hubel and Wiesel found. CNN is a lot like a regular Neural Network in most ways. In other words, MLPs with a special structure are what Convolutional Neural Networks are.

Deep Convolutional Neural Networks (DCNN) is a big step forward in recognizing patterns. Since it was found, some systems built on CNN have won the ILSVRC. These include AlexNet 2012 [22], Clarifai 2013 [23], GoogLeNet 2014 [24], and ResNet 2015 [25]. In this case, a CNN was used to solve the classification problem. After that, experts turned to CNN to solve many other problems. As [26] says, however, CNN design still needs to be improved to make it more accurate and fit the case at hand. When CNN is combined with some other ideas, it will be able to do more.

Restricted Boltzmann machine (RBM) is a stochastic graph model that can learn a probability distribution over its set with n visible unit inputs and m hidden feature unit [27,28]. In general, RBM's way of teaching is unsupervised learning. In this case, learning RBM doesn't have to include the goal output as something that needs to be done. The learning process ends when each of the training data has been used a certain number of times. RBMs have been used successfully to solve many problems, such as dimension reduction [29], classification [30], joint filtering [31], feature learning [32][33], and modeling [34]. [35] and [36] are good places to learn more about RBM and deep design. RBMs can be used to improve the quality of the picture, as [37] shows.

In this study, we use Restricted Boltzmann Machine (RBM) and Deep Convolutional Neural Networks (DCNN) to come up with a way to solve low-resolution face recognition. Both unsupervised learning and guided learning are brought

together in the suggested method. Here, RBM is used to prepare the low-resolution face picture before it is recognized by DCNN. This study is important because it: 1) shows the first designed architecture of an artificial neural network using RBM and DCNN to recognize low-resolution face images; 2) improves the quality of low-resolution face images; and 3) raises the rate at which faces are recognized.

2. Hybrid Restricted Boltzmann Machine and Deep Convolution Neural Networks

In this study, we come up with a method that combines RBM and DCNN to solve low-resolution face recognition issues. This method has two steps: the learning step and the testing step. Both RBM and DCNN algorithms are used in each step. In both steps, low-resolution images are first put through the RBM algorithm. In the learning phase, however, the RBM algorithm's job is to figure out the weights on the RBM network so that a single image can be reconstructed. This image will be used as input for DCNN learning. In the testing phase, however, the RBM algorithm's job is to turn low-resolution input into an image that will be used as input for DCNN classification.

The proposed method consists of five steps as shown in Figure 3.

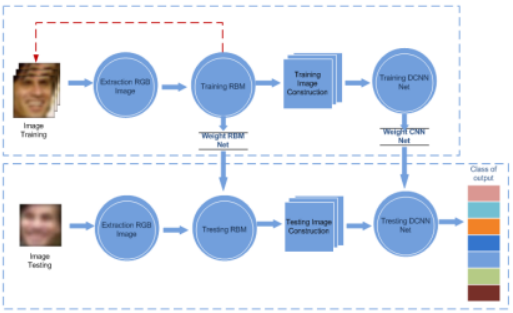


Figure 2. The scheme of the proposed method.

In summary the working mechanism of the proposed method is listed in the following. It begins with learning of RBM Net. This is done by unsupervised learning. There are two types of the output of this RBM, namely the weight and the result of image reconstruction. After the learning of RBM is complete, it then goes to the DCNN learning. In this case the input of DCNN is the reconstructed image from the output of RBM. This learning is done by supervised learning. The outputs of this CNN learning are the weight of CNN Net, feature maps layer convolution and the weight of the fully connectivity layer. After the learning phase is completed, the next step is the stage of testing or the stage of classification of low resolution image input. The detailed explanation is shown as follows:

A. Training Restricted Boltzmann Machine

Furthermore, step by step, training RBM described in [38-50]. Suppose I is a low resolution face image of size $m\,x\,n$. Let $I_r, I_g,$ and I_B, each of size $m\,x\,n$ be the three color channels (red, green, and blue) of I. Each of them is represented as:

$$I_{r(m,n)} = \begin{vmatrix} I_{r(11)} & \cdots & I_{r(1n)} \\ \cdots & \cdots & \cdots \\ I_{r(m1)} & \cdots & I_{r(mn)} \end{vmatrix}$$

$$I_{g(m,n)} = \begin{vmatrix} I_{g(11)} & \cdots & I_{g(1n)} \\ \cdots & \cdots & \cdots \\ I_{g(m1)} & \cdots & I_{g(mn)} \end{vmatrix}$$

$$I_{b(m,n)} = \begin{vmatrix} I_{b(11)} & \cdots & I_{b(1n)} \\ \cdots & \cdots & \cdots \\ I_{b(m1)} & \cdots & I_{b(mn)} \end{vmatrix}$$

Let I' be the image that is reconstructed from image I and it is shown in Figure 4. Let $I'_r, I'_g,$ and I'_b, each of size $m\,x\,n$, be the three color channels reconstructed images. Each of them is represented as:

$$I'_{r(m,n)} = \begin{vmatrix} I'_{r(11)} & \cdots & I'_{r(1n)} \\ \cdots & \cdots & \cdots \\ I'_{r(m1)} & \cdots & I'_{r(mn)} \end{vmatrix}$$

$$I'_{g(m,n)} = \begin{vmatrix} I'_{g(11)} & \cdots & I'_{g(1n)} \\ \cdots & \cdots & \cdots \\ I'_{g(m1)} & \cdots & I'_{g(mn)} \end{vmatrix}$$

$$I'_{b(m,n)} = \begin{vmatrix} I'_{b(11)} & \cdots & I'_{b(1n)} \\ \cdots & \cdots & \cdots \\ I'_{b(m1)} & \cdots & I'_{b(mn)} \end{vmatrix}$$

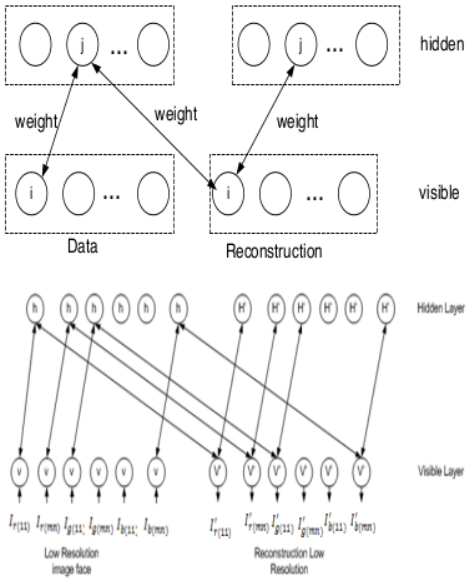


Figure 3. The process of reconstructing a low resolution face image using RBM.

The training rules of RBM is as follows:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} + \eta^{(k)}\epsilon\left(\left(v_i h_j\right)_{data} - \left(v_i h_j\right)_{reconstructon}\right) \quad (1)$$

$$b_i = b_i + \eta[\langle v_i\rangle_{data} - \langle v_i\rangle_{reconstruction}] \quad (2)$$

$$c_j = c_j + \eta\left[\langle h_j\rangle_{data} - \langle h_j\rangle_{reconstruction}\right] \quad (3)$$

B. Training Deep Convolution Neural Networks

B.1. Architecture CNN

Convolutional Neural Networks (CNN) is special case of neural networks [50]. CNN have one or more convolution layers, sub sampling layers, and fully connectivity layer.
In general, the CNN architecture for low resolution face image is shown in Figure 6. Each layer receives input from a set of features that reside in a small environment in the previous layer called the receptive field. With local receptive fields, it can extract basic visual features, which is then merged by a higher layer.

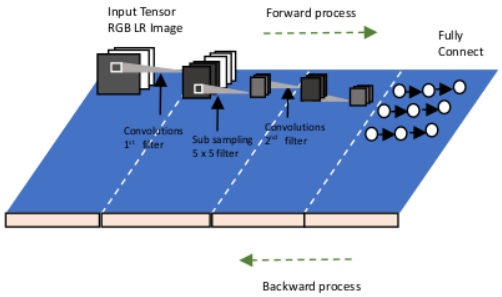


Figure 4. The architecture of DCNN [51].

The architecture in Figure 6 can be avowed using abstract description as stated in Equation (31) [52]:

$$I'^{(1)} \rightarrow K^{(1)} \rightarrow I'^{(2)} \rightarrow \cdots \rightarrow K^{(l-1)} \rightarrow I'^{(l)} \rightarrow K^{(l)} \rightarrow z \quad (1)$$

Equation (31) illustrates layer by layer of the CNN in a forward pass. For the first layer, the input of CNN is $I'^{(1)}$ and the kernel filter is $K^{(1)}$. Here, $I'^{(1)}$is a low resolution face image consisting of order 3 tensor and $K^{(1)}$is a matrix of the feature maps. If b_1 is the bias matrix on the first layer, the result of convolution $I'^{(1)}$ with $K^{(1)}$ is denoted as$I'^{(2)}$, that is, $I'^{(2)} = b_1 + I'^{(1)} * FK^{(1)}$. Now $I'^{(2)}$is the input to the second layer with another kernel filter$K^{(2)}$ and using the same mechanism as the first layer, we can determine $I'^{(3)}$, $I'^{(4)}$, …, in sequence. In other words, the first convolution layer will receive the tensor value of the input from the original image. The second layer will accept the input from the output of the first layer, and so on.
In this research, the layers and parameters of the CNN architecture used are given in Table 2in details.

Table 2. CNN Parameters setting for low resolution face image recognition

Layer	Size
Input	72x64x3
Convolution 1	5x5x3x64
Output 1	68x60x64
Convolution 2	3x3x3x128
Output 2	66x58x128
Pooling (3 x3 kernel)	22x19x128
Convolution 3	3 x3 x3x128
Output3	20x17x128
Convolution 4	3 x3 x3x128
Output	18x15x128
Pooling (3 x3 kernel)	6 x 5 x128
Convolution 5	1x1x3x512
ReLU	6x5x512
Fully Connectivity	17920

B.2. Convolution Layer

In Equation (21) the convolution is performed using a single kernel. Meanwhile, as shown in Figure 6, CNN has many layers and in each layer the convolution operation uses multiple kernels. Thus the convolution on CNN is done repeatedly in accordance with the number of layers used.
The convolution mechanism for the input elements starts from the top left corner, then the filter kernel moves right one by one until it reaches the top-right corner. After that, the filter kernel is then moved one element downward, and repeat the same process as before. This process is repeated continuously until all input elements are convoluted.
Let $K^{(l)}$ and $I^{(l)}$ be the kernel filter and the input of the l^{th} convolutionlayer, respectively, then the output of the convolution is a feature map which can be calculated using Equation (32).

$$Y_{r,s}^{(l)} = B^{(l)} + \sum_{u=-H_1}^{H_1} \sum_{v=-H_2}^{H_2} \sum_{d=0}^{D} K_{u,v}^{l} * I_{r+v,s+v}^{(l)} \quad (2)$$

Suppose the input of the first layer is $I_{MxN}^{(1)}$ (i.e. a 3-level tensorR^{MxNxD}) which is convoluted with a$K^{(l)}$kernel filter of size $H_1 \; x \; H_2 \; x \; D$ separately. Since the input is convoluted with a single kernel, it produces an output feature map which is a matrix of size $(M - H_1 + 1)x\,(N - H_2 + 1)$. Suppose it is convoluted with NK kernel filters, it will then produce NK feature maps, independently.
Equation (32) can be implemented using Algorithm 2 listed in Table 3.

Table 3. Algorithm for Convolution Layer

Algorithm 2: Convolution Layer

Input:
I: the output matrix from the previous layer
(if l=1, then I is the value of RGB image)
M: the number of rows of I
N: the number of columns of I
H1: the number of rows of a kernel
H2: the number of columns of a kernel
CNO: Channel number
NK1: the number of kernels in each layer

Output:
Matrix I'

```
for o: 0 to NK1)
  for r:1 to (M-H1+1)
    for s:1 to (N-H2+1))
      for i:0 to CNO){
        for lh1:0 to H1    //size of Kernel H1 x H2
          for lh2:0 to H2{
            Yr,s[o][r][s]=  O1[o][r][s]+ K1[o][i][lh1][lh2]
                          * I[i][r+lh1][s+lh2]
          end
        end
      end
          I'[o][r][s]= b[o] + Yrs[o][r][s]
    end
```

```
    end
  end
```

Normally, the low level layer will extract low level features like edges, lines, and angles. It then extracts the higher level features at a higher level. Figure 7 illustrates the 3D tensor convolution process used in CNN in the first layer.

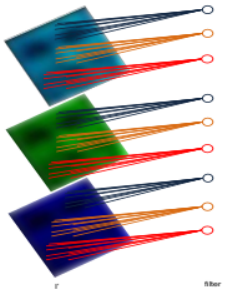


Figure 5. Illustration at convolution in $l = 1$ at DCNN.

B.3. Non-Linearity Layer

A neuron of CNNs usually uses non-linear activation function to transform the input to its output. The output of a neuron can have different responses depending on the activation functions used. Some of the most commonly used activation functions are softmax, hyperbolic tangent, sigmoid and rectified linear units (ReLU). Mathematically these functions are represented by Equations (34) (35) and (36) respectively. Graphically, they are shown in Figure 8. The activation function of ReLU is the most widely used function [51], this is because ReLU function is a linear function while the input is greater than or equal to 0 and 1 can improve the nonlinear properties of the decision function [52]. The relation between the input and the output of a neuron at layer l can be expressed as:

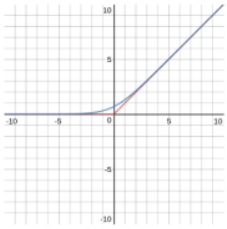
$$Y_i^{(l)} = \phi(Y_i^{(l-1)}) \quad (3)$$

where ϕ is an activation function and it can be softmax, tanh or max, respectively. Graphically, the functions are shown by Figure 8.

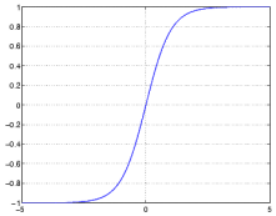
$$\phi_{softmax}(z^{(i)}) = \frac{e^{z^{(i)}}}{\sum_{j=0}^{k} e^{z_k^{(i)}}} \quad (4)$$

$$\phi(z^{(i)}) = \tanh((z^{(i)}) \quad (5)$$

$$\phi(z^{(i)}) = max\{0, z^{(i)}\} \quad (6)$$



(a)



(b)

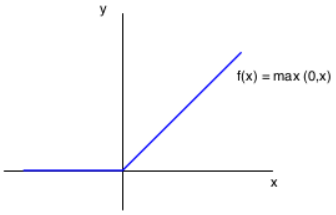


Figure 6. (a) Sofmax function, (b) tanh function, (c) ReLU function $\mathbf{y = max\{0, x\}}$.

B.4. Feature Pooling and Sub Sampling Layer

Pooling layer / sub sampling layer is a layer that serves to reduce the feature resolution. The purpose of Pooling is that it makes the feature more resistant to noise and distortion. There are two ways to do pooling: max pooling and average pooling. The first step in both cases is the same, i.e. the input feature is partitioned into a non-overlapping two-dimensional matrix segment. If the input is N x M size and the matrix segment iss_1 x s_2, then there will be $\frac{N}{s_1}$ x $\frac{M}{s_2}$ region each of size s_1 x s_2. The Max pooling is obtained by taking the max value of each region, while for average pooling, it takes the average value of each region. An illustration of max pooling is shown in Figure 9.

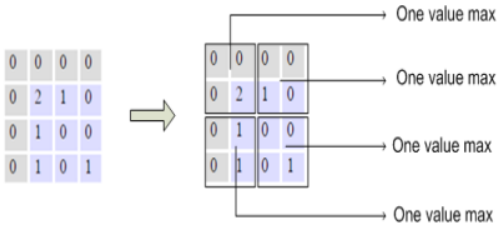


Figure 7. Illustration of Max Pooling process.

B.5. Fully Connectivity layer

A fully connected layer is often used as the last stage of the CNN for classification [52][53]. Figure 10 shows the fully connected layer.

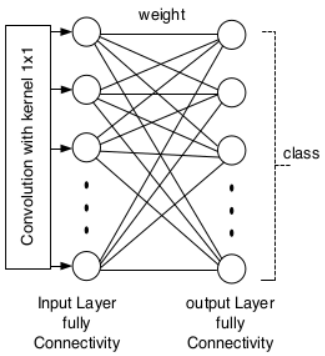


Figure 8. A fully connected layer.

Let $Y_j^{(l-1)}$ be a ReLU value of the previous convolution layer with a kernel filter of size1x1. For the fully connected layer, let $Y_j^{(l-1)}$ be theoutput of the $l-1^{th}$ layer, $w_{jk}^{(l)}$ be the weights between j and k neurons $z_k^{(l)}$ be the summation of all outputs connected to the k^{th} neuron of the l^{th} layer. Equations (37) and (38) show the relationship between these parameters.

$$z_k^{(l)} = \sum_{j=1}^{P} Y_j^{l-1} w_{jk}^l \quad (7)$$

$$Y_k^{(l)} = \phi(z_k^{(l)}) \quad (8)$$

Where ϕ is a softmax function, P is the size of the feature map with a kernel filter of size 1 x 1, $0 \le k < Q$ and Q is the number of classes.

B.6. *Feed forward Pass*

The abstraction process in Equation (31) is the feed forward pass of CNN. In the training phase, this process is repeated until reaches the target.

Suppose there are c classes and N training patterns, the squared-error loss function for the r training pattern is defined by Equation (39):

$$E^r = \frac{1}{2}\sum_{k=1}^{c}(T_k^r - Y_k^r)^2 \quad (9)$$

Here T_k^r is the $k-th$ dimension of the $r-th$ pattern's corresponding target (label), and Y_k^r is the output of the $k-th$ neuron in response to the $r-th$ pattern as counted using Equation (38). The total error is then $\sum_{r=1}^{N} E^r$.

B.7. *Backpropagation Pass*

Look again at Equation (31), now we assume that the error that occurs when propagates backward throughout the network is the sensitive response of each unit as a result of interference of the bias. Let $u^l = W^l Y^{l-1} + b^l$ be the input of a neuron at layer l, where W, Y and b are the weights, output and bias, respectively. For any layer, this gives meaning that:

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u}\frac{\partial u}{\partial b} = \delta \quad (10)$$

In this case, $\frac{\partial u}{\partial b} = 1$. While the bias sensitivity and the derivative of the error with respect to the total input unit is equivalent. Furthermore, the derivative of this backpropagation from the next layer to the previous layer uses the following recurrence relations:

$$\delta^l = (W^{l+1})^T \delta^{l+1} o\, \phi'(u^l) \quad (11)$$

where "o" denotes the element-wise multiplication. For the error function in Equation (39), especially in the fully connectivity layer L, the sensitivities for the output layer neurons will take a slightly different form

$$\delta^L = \phi'(u^L)\, o\, (y^n - t^n) \quad (12)$$

Finally, the delta rule for updating a weight assigned to a given neuron is just a copy of the inputs to that neuron, scaled by the neuron's delta. In vector form, this is computed as an outer product between the vector of inputs Y^{l-1} (which are the outputs from the previous layer) and the vector of sensitivities δ^l:

$$\frac{\partial E}{\partial W^l} = Y^{l-1}\delta^l \quad (13)$$

$$\Delta W^l = -\eta \frac{\partial E}{\partial W^l} \quad (14)$$

$$W^l(\text{new}) = W^l(old) + \Delta W^l \quad (15)$$

In practice there is often a learning rate parameter η_{ij} specific to each weight $(W)_{ij}$. Bias can be updated similarly.

B.8. *Gradient Descent*

Gradient descent is a first order iterative optimization algorithm. In this research we follow[53] to count and arrange gradient descent. An illustration of gradient descent is shown in Figure 11.

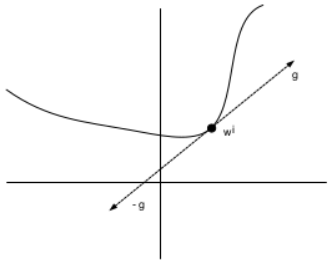


Figure 9.An illustration of gradient descent.

a. Computing Gradients in Convolution Layer

Let us see again the feed forward CNN layer by layer in Equation (31). Equation (32) then can be rewritten in Equation (46). Now we follow the description shown in [53] in the following. At a convolution layer, the previous layer's feature maps are convolved with learnable kernels and then go through the activation function f to form the output feature map. An output feature map may probably convolute with multiple input maps.

$$I_j^l = f\left(\sum_{i\in M_j} I_i^{l-1} * K_{ij}^l + b_j^l\right), \quad (16)$$

where M_j denotes a selection of input maps. In convolution operation, bias b is added to each output map; however, for some output map, the input maps are convolved using distinct kernels.

Furthermore, to compute the gradient in convolution layer, we assume that each convolution layer l is followed by a down sampling layer $l+1$. The backpropagation algorithm mentions that in order to compute the sensitivity for a unit at layer l, fist the next layer's sensitivities corresponding to units that are connected to the node of interest in the current layer l, are

multiplied by the associated weights one by one defined at layer $l+1$ under these connections and then these values are then summed up. This result is then multiplied by the derivative of the activation function evaluated at the current layer's pre-activation inputs, u^l . As you can see, a convolutional layer is followed by a downsampling layer and a block of pixels in the convolutional layer's output map correspond to one pixel in the next layer's associated sensitivity map δ. Thus each unit in a map at layer l connects to only one unit in the corresponding map at layer $l+1$. An easy way to compute the sensitivities at layer l is to upsample the downsampling layer's sensitivity map to the same size as the previous convolutional layer's map and then just multiply the upsampled sensitivity map from layer $l+1$ with the activation derivative map at layer l element-wise. Let β (a constant) be the weights defined at a downsampling layer map. δ^l can then be obtained by just scaling the previous step's result by β. This process can repeat for each map j in the convolutional layer, pairing it with the corresponding map in the subsampling layer:

$$\delta_j^l = \beta_j^{l+1}(f'(u_j^l) o\, up(\delta_j^{l+1}), \qquad (17)$$

where up(.) denotes an upsampling operation.One possible way to implement this function efficiently is to use the Kronecker product:

$$\mathrm{up}(x) \equiv x \otimes 1_{nxn} \qquad (18)$$

The bias gradient can be immediately computed by summing over all the entries in $\delta_j^l$from the sensitivities for a given map:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{uv} \qquad (19)$$

At last, using backpropagation the gradients for the kernel weights are calculated, except that the same weights are shared across many connections. Similarly to the bias term, the gradients for a given weight can be summed up over all the connections that mention this weight:

$$\frac{\partial E}{\partial k_{ij}^l} = \sum_{u,v} (\delta_i^l)_{uv} (p_i^{l-1})_{uv}, \qquad (20)$$

where $(p_i^{l-1})_{uv}$ is the patch in I'^{l-1}_i that was multiplied elementwise by K_{ij}^l during convolution for computing the element at (u,v) in the output feature mapI'^l_j as mentioned in Equation (46).

b. Computing Gradients in Subsampling Layer

In [53], a subsampling layer (pooling layer) produces downsampled versions of the input maps with the smaller size. More formally,

$$I_j'^l = f(\beta_j^l\, down(I_j'^{l-1}) + b_j^l), \qquad (21)$$

where down(·)denotes a sub-sampling function as mentioned in Sec. 3.B.4.

As you can see only learnable parameters β and b should be updated. Usually, the subsampling layers are surrounded above and below by convolution layers. Suppose a fully connected layer follows the subsampling layer, the sensitivity maps for the subsampling layer can be computed with Equation (42).

Furthermore, to compute the gradient of a kernel in kernel convolution, one has to figure out which patch in the current layer's sensitivity map corresponds to a given pixel in the next layer's sensitivity map in order to apply a delta recursion looking like Equation (41).

The gradient for b is again just the sum over the elements of the sensitivity map

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{uv} \qquad (22)$$

During the feedforward pass, the multiplicative bias β will certainly involve the original down-sampled map computed at the current layer. Let

$$d_j^l = down(x_j^{l-1})$$

The gradient for β is then computed by

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} (\delta_j^l\, o\, d_j^l)_{uv} \qquad (23)$$

Now a brief description step by step for the proposed method is shown in Figure 12.

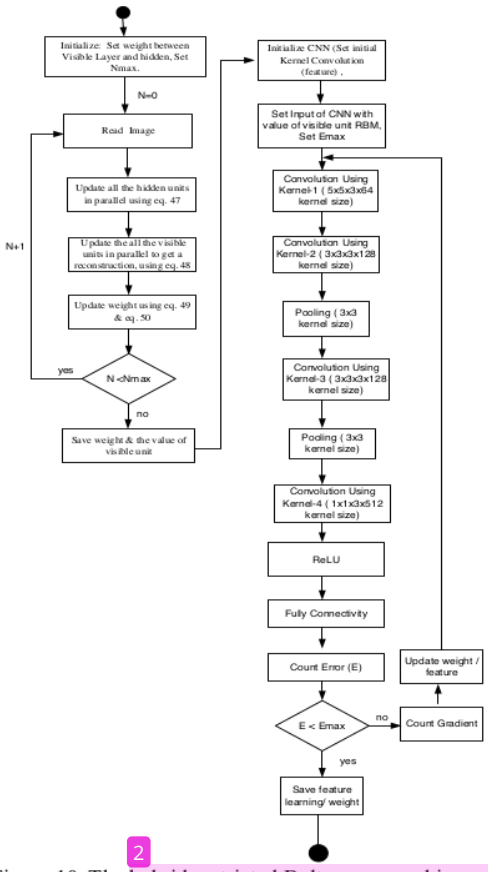


Figure 10. The hybrid restricted Boltzmann machine and deep convolution neural network.

3. Results and Discussion

To evaluate the performance of the proposed method, we use the data as done in [1], i.e. the combination of data contained in [54] and [55] and the Lfw database [56][57]. We chose random data from the data set of the Georgia Institute of Technology [58], Aleix Martinez and Robert Benavente [59], and a database of facial recognition technologies [60]. The numbers of persons of each data set are 40, 70, and 500 people, respectively. Each person has 5 different expressions and

positions. So the number of facial images used is 610 x 5 = 3050 images. Our face data is cut to 120 x 165 pixels and then the data is converted into low resolution images using a Gaussian filter.

For the sake of the same data size as in [1], scaling operation is needed. That is, for face verification, the face image is scaled to 72 x 64 pixels, while for the identification process, the face image is scaled to 18 x 16 pixels.

For each person we randomly take face images to train the RBM-CNN network. Subsequent to convergent training, we do two-stage testing. The first step is to use training data and this test is to verify the knowledge gained as a result of the learning process. Normally, the accuracy of this test should be 100%. Furthermore, the second stage of testing uses untrained data and this test is performed in order to see the performance of the system.

The sample data set and the corresponding low resolution data set are shown in Figures 14(a) and 14(b), respectively.

A. Experiments for face verification

Face verification is a matching activity of two face images, wether they are of the same person or not. In this study the number of verified persons is 610. The way used to perform verification is by using a system trained with one image from a person until reaching convergence. After that, the remaining four face images with different expressions and positions in the data set are taken one by one to be matched. This is repeated 610 times for all persons to be verified.

Furthermore, to measure the effectiveness of the proposed method, we use the benchmark as used by [1]. This benchmark has been used in PCA, S^2R^2, DSR, CKE, respectively. The overall comparisons of the existing results and those from the proposed method are shown in Figure 13.

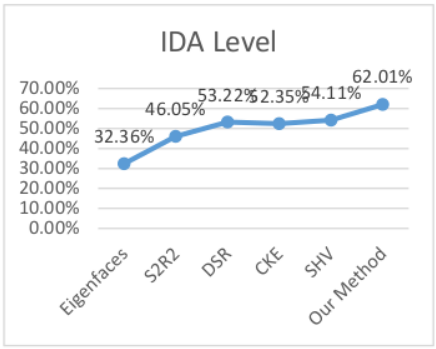


Figure 11. Identification Accuracy (IDA) levels for different methods.

As shown in Figure 13, the IDA level obtained from our method is 62.01%, which is better than any existing methods.

From the test results, we find that many verification errors on expression no. 4 from AR database in Figure 14.This is because this expression is different from the facial expression in general, i.e. the mouth is open and the eyes are closed. However, when the training phase of the system has been trained with this expression, then the system can recognize it well. And also if one's expression is not too different from ordinary expression, then the system can also recognize it well.

B. Experiments for face identification

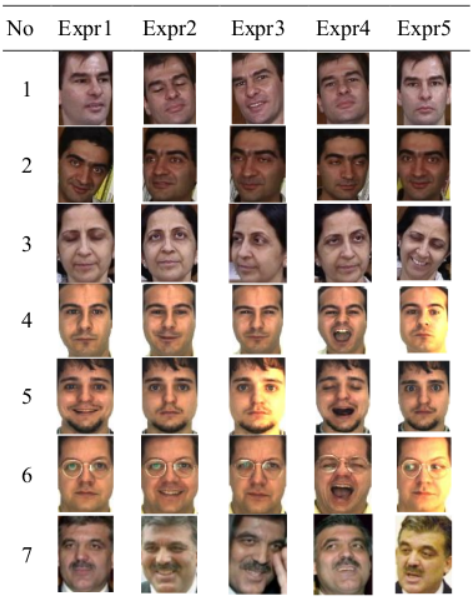
Face identification is a process to recognize a person's identity based on matching a face image to a set of facial images in the database. To do this, in this study the system was trained by using three faces taken at random for each person. Training is done repeatedly until convergence.

Furthermore, we use the benchmark provided by [1] to verify the proposed method. For this purpose, we divide the test mechanism into two ways. For the first way, a mixed data set is generated from the training dataset and the blind dataset, where each person has three face images from the training dataset and two face images from the blind dataset. For the second way, the test dataset has the trained dataset and blind dataset (never included in the training phase) separately. The results for the test under the mixed dataset yielded an IDA level 91.06%. Meanwhile, the results under the test with the trained data set and blind dataset generated an IDA level 100% and an IDA level 78.33%, respectively. Under the blind dataset testing, the results obtained from our method are compared to those of existing methods and they are shown in Table 4.

Table 1. Comparison of results using IDA level

Method	IDA Level
Eigenfaces [62 in 1]	39.44%
LBP [65 in 1]	42.58%
Gabor [66 in 1]	43.71%
SR[61]+Eigenfaces [62 in 1]	41.19%
SR[61]-LBP [65 in 1]	44.19%
SR[61]-Gabor [66 in 1]	45.20%
S^2R^2 [61 In the 1]	55.70%
DSR [63 in 1]	71.66%
CKE [64 in 1]	71.24%
SHV[1]	72.15%
Our Method	**78.33%**

From Table 4, it shows that the IDA level of our method is better than that of any existing methods.



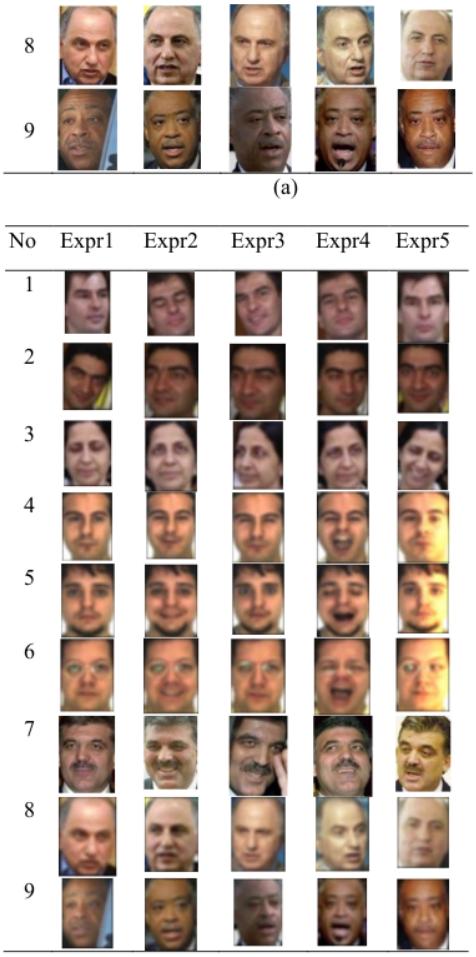


Figure 12. Sample data set each with 5 expressions. (a) The original dataset. (b)The corresponding low resolution dataset based on Gaussian Filtering. Data is combination from GT database (no. 1-3), AR database (No. 4-6), and LFW database (No. 7-9).

4. Conclusion

As you can see, currently there are many commercial face recognition systems. Owing to specific cases such as low illumination, different orientation, twisted expression, etc., the false recognition rate of face recognition system can still be high. Hence, it still needs to be improved by newer approaches. In this research, we propose a new approach by combing RBM and DCNN together to do the face recognition. The major contributions are listed in the following.

a. The proposed RBM-DCNN network architecture can improve the IDA level of low resolution face image recognition.
b. The considerable difference in facial expression between training data and testing data reduces the accuracy of the recognition. In other words, facial expressions that are too much different between the test data and the training data affect IDA levels.

References

[1] Jian, M., &Lam, K. M. (2015). Simultaneous hallucination and recognition of low-resolution faces based on singular value decomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, *25*(11), 1761–1772.

[2] Li, J., Zhao, B., &Zhang, H. (2009). Face recognition based on PCA and LDA combination feature extraction. *2009 First International Conference on Information Science and Engineering*,(20042013), 1240–1243.

[3] Zou, W. W. W., &Yuen, P. C. (2012). Very low resolution face recognition problem. *IEEE Transactions on Image Processing*, *21*(1), 327–340.

[4] Yang, R., Wang, Y., Yang, D., Xu, T., &Zhou, J. (2011). Face hallucination via using the graph-optimal locality preserving projections. *2011 10th IEEE/ACIS International Conference on Computer and Information Science*, 189–193.

[5] Yang, C. Y., Liu, S., &Yang, M. H. (2013). Structured face hallucination. *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 1099–1106.

[6] An, L., &Bhanu, B. (2014). Face image super-resolution using 2D CCA. *Signal Processing*, *103*, 184–194.

[7] Biswas, S., Aggarwal, G., Flynn, P. J., &Bowyer, K. W. (2013). Pose-robust recognition of low-resolution face images, *35*(12), 3037–3049.

[8] Park, J. S., &Lee, S. W. (2008). An example-based face hallucination method for single-frame, low-resolution facial images. *IEEE Transactions on Image Processing*, *17*(10), 1806–1816.

[9] Ren, C. X., Dai, D. Q., &Yan, H. (2012). Coupled kernel embedding for low-resolution face image recognition. *IEEE Transactions on Image Processing*, *21*(8), 3770–3783.

[10] Zhao, W., Chellappa, R., Phillips, P. J., &Rosenfeld, a. (2003). Face recognition: A literature survey. *Acm Computing Surveys*, *35*(4), 399–458.

[11] Wang, Z., Miao, Z., Jonathan Wu, Q. M., Wan, Y., &Tang, Z. (2014). Low-resolution face recognition: A review. *Visual Computer*, *30*(4), 359–386.

[12] Supardi, J., & Utami, A. S. (2014). Development of artificial neural network architecture for face recognition in real time. *International Journal of Machine Learning and Computing*, *4*(1), 110–113.

[13] Aitkenhead, M. J., &McDonald, A. J. S. (2003). A neural network face recognition system. *Engineering Applications of Artificial Intelligence*, *16*(3), 167–176.

[14] Tisan, A., &Chin, J. (2016). An end-user platform for FPGA-based design and rapid prototyping of feedforward artificial neural networks with on-chip backpropagation Learning. IEEE Transactions on Industrial Informatics, 12(3), 1124–1133.

[15] Riggan, B. S., Reale, C., &Nasrabadi, N. M. (2015). Coupled auto-associative neural networks for heterogeneous face recognition. *IEEE Access*, *3*, 1620–1632.

[16] Boughrara, H., BenAmar, C., Chtourou, M., &Chen, L. (2014). Face recognition based on perceived facial images and multilayer perceptron neural network using constructive training algorithm. *IET Computer Vision*, *8*(6), 729–739.

[17] Zhang, M., &Fulcher, J. (1996). Face recognition using artificial neural network group-based adaptive tolerance (GAT) trees. *IEEE Transactions on Neural Networks*, *7*(3), 555–567.

[18] Ranganath, s., Arun, K. (1997). Face recognition using transform features and neural networks. *Pattern Recognition,* vol. 30, no. 10, 1615-1622

[19] Lin, J., Ming, J., &Crookes, D. (2009). Robust face recognition using posterior union model based neural networks. *Iet Computer Vision*, *3*(3), 130–142.

[20] Inoue, T., Miyakawa, H., Ito, K., Mikoshiba, K., &Kato, H. (1992). A hyperpolarizing response induced by glutamate in mouse cerebellar Purkinje cells. *Neuroscience Research*, *15*(4), 265–271.

[21]LeCunn, Yann., Bottou, L., Bengio, Y., Haffner,P.(1998). Gradient-based learning applied to document recognition. Proceedingsofthe IEEE, vol. 86, issue, 11, Nov. 1998, 2278-2324.

[22] Krizhevsky, A., Sutskever, I., &Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1–9.

[23] Zeiler, M. D., &Fergus, R. (2014). Visualizing and understanding convolutional networks. *Computer Vision–ECCV 2014*, *8689*, 818–833.

[24] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., …Rabinovich, A. (2015). Going deeper with convolutions. *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *7-12-NaN-2015*, 1–9.

[25] He,K., Zhang,X., Ren, S., Sun,J. (2015). Deep residual learning for image recognition. CVPR 2015.

[26] Shankar, S., Robertson, D., Ioannou, Y., Criminisi, A., &Cipolla, R. (2016). Refining architectures of deep convolutional neural networks. arXiv:1604.06832 [cs.CV]

[27] Fischer, A., &Igel, C. (2014). Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, *47*(1), 25–39.

[28] Fischer, A., &Igel, C. (2012). An introduction to restricted Boltzmann machines. *Lecture Notes in Computer Science: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, *7441*,

14–36.
[29] Hinton, G. E., Salakhutdinov, R. R.. (2006). Reducing the dimensionality of data with neural networks. *SCIENCE*, Vol. 313, Issue 5786, 2006, 504-507.
[30] Larochelle, H., &Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. *Icml*, 536–543.
[31] Salakhutdinov, R., Mnih, A., &Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine Learning - ICML '07*, 791–798.
[32] Zheng, X., Wu, Z., Meng, H., Li, W., &Cai, L. (2013). Feature learning with Gaussian restricted Boltzmann machine for robust speech recognition. arXiv:1309.6176 [cs.CL]
[33]Tran, S. N., Wolff, D., Weyde, T., &Garcez, A. (2014). Feature preprocessing with RBMs for music similarity learning. *AES 53th*, 1–8.
[34] Ranzato, M. A., &Hinton, G. E. (2010). Factored 3-way restricted Boltzmann machines for modeling natural images. *Artificial Intelligence*, *9*, 621–628.
[35] Salakhutdinov, R., &Hinton, G. (2009). Deep Boltzmann machines. *Aistats*, *1*(3), 448–455.
[36] Bengio. Y.,(2009). Learning deep architectures for AI.*Foundations and Trends® in Machine Learning*. Vol. 2, No. 1 (2009) 1–127
[37] Sahasrabudhe, M., &Namboodiri, A. M. (2014). Fingerprint enhancement using unsupervised hierarchical feature learning. *Proc. of the 2014 Indian Conference on Computer Vision Graphics and Image Processing - ICVGIP '14*, 1–8.
[38] Hinton, G. (2014). Boltzmann machines. *Encyclopedia of Machine Learning and Data Mining*, (1), 1–7.
[39] Fischer, A., &Igel, C. (2014). Training restricted Boltzmann machines. *Pattern Recogn.*, *47*(1), 25–39.
[40] Cho, K. H. (2011). Improved learning algorithms for restricted Boltzmann machines. *Master's thesis, Aalto University School of Science*.
[42] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.
[43]Welling, M. Product of experts. Scholarpedia 2(10), 3879 (2007)
[44] LeRoux, N., &Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, *20*(6), 1631–1649.
[45] Montufar, G., &Ay, N. (2010). Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines, (2010), 1–12.
[46] Restricted Boltzmann Machines (RBM) — DeepLearning 0.1 documentation
[47] Yildirim, I. (2012). Bayesian inference: Gibbs sampling, *14627*, 1–6.
[48] Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Computer*, *9*(3), 1.
[49] Stutz, D. (2014). Understanding convolutional neural networks. *Nips 2016*, (3), 1–23.
[50] Samer, C. H., Rishi, K., &Rowen. (2015). Image recognition using convolutional neural networks. *Cadence Whitepaper*, 1–12.
[51] LeCun, Y., Kavukcuoglu, K., &Farabet, C. (2010). Convolutional networks and applications in vision. *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 253–256.
[52] Wu, J. (.2017). Introduction to convolutional neural networks.(access: June, 16, 2017)
[53] Bouvrie, J. (2006). Notes on convolutional neural networks. *In Practice*, 47–60.
[54] Hu, Y., Lam, K. M., Qiu, G., &Shen, T. (2011). From local pixel structure to global image super-resolution: A new face hallucination framework. *IEEE Transactions on Image Processing*, *20*(2), 433–445.
[55] Jian, M., Lam, K.-M., &Dong, J. (2013). A novel face-hallucination scheme based on singular value decomposition. *Pattern Recognition*, *46*(11), 3091–3102.
[56] Huang, G. B., Jain, V., &Learned-Miller, E. (2007). Unsupervised joint alignment of complex images. *Proceedings of the IEEE International Conference on Computer Vision*.
[57] Huang, G. B., Mattar, M. A., Lee, H., &Learned-Miller, E. (2012). Learning to align from scratch. *Proc. Neural Information Processing Systems*, 1–9.
[58] Georgia Inst. Technol. *GT Face Database Atlanta*. [Online]. Available: http://www.anefian.com/research/face_reco.htm
[59] A. R. Martinez and R. Benavente. The AR face database. CVC, Barcelona, Spain, Tech. Rep. 24, 1998. [Online]. Available: http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html
[60] Phillips, P. J., Wechsler, H., Huang, J., &Rauss, P. J. (1998). The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, *16*(5), 295–306.
[61] Hennings-Yeomans, P. H., Baker, S., &Kumar, B. V. K. V. (2008). Simultaneous super-resolution and feature extraction for recognition of low-resolution faces. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.
[62] Turk, M., Pentland, A.,(1991). Eigenfaces for recognition.*J. Cognit. Neurosci.*, vol. 3, no. 1,71–86
[63] Zou, W. W. W., &Yuen, P. C. (2012). Very low resolution face recognition problem. *IEEE Transactions on Image Processing*, *21*(1), 327–340.
[64] Ren, C. X., Dai, D. Q., &Yan, H. (2012). Coupled kernel embedding for low-resolution face image recognition. *IEEE Transactions on Image Processing*, *21*(8), 3770–3783.
[65] Ahonen, T., Hadid, A., & Pietikäinen, M. (2004). Face recognition with local binary patterns, ECCV 2004, 469–481.
[66] Liu, C., &Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, *11*(4), 467–476.

Julian Supardi He is a lecture in Universitas Sriwijaya, Palembang, South Sumatera, Indonesia. He is currently a Ph.D. candidate and looking forward to his degree in computer science and information engineering in National Taiwan University of Science and Technology. His research interests include image processing and deep learning.

Shi-Jinn Horngreceived the B.S. degree in electronic engineering from National Taiwan Instituteof Technology, Taiwan, the M.S.degree in information engineering from NationalCentral University, Taiwan, andthe Ph.D. degree in computer science from National Tsing Hua University,Taiwan, in 1980,1984, and 1989, respectively.Currently, he is a Chair Professor in the Department ofComputer Science and Information Engineering, National Taiwan Universityof Science and Technology. His research interestsinclude Deep Learning, Biometric Recognitions.

Tianrui Li received his B.S. degree, M.S. degree and Ph.D. degree from the Southwest Jiaotong University, China in 1992, 1995 and 2002 respectively. He is presently a Professor and the Director of the Key Lab of Cloud Computing and Intelligent Technique of Sichuan Province, Southwest Jiaotong University, China. Since 2000, he has co-edited 5 books, 10 special issues of international journals, 14 proceedings, received 1 Chinese invention patent and published over 200 research papers. He was recognized as the Top 1% Scientists in the field of Computer Science based on Thomson Reuter's Essential Science Indicators (ESI) in Sept. 2016.

Appling both hybrid RBM and DCNNs to FR low-resolution

ORIGINALITY REPORT

14%
SIMILARITY INDEX

PRIMARY SOURCES

1	docplayer.net Internet	428 words — 8%
2	www.scitechnol.com Internet	204 words — 4%
3	hdl.handle.net Internet	127 words — 2%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	< 2%
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF