1-1-2012

Modular dynamic RBF neural network for face recognition

Sue Inn Ch'Ng

Kah Phooi Seng

Li-minn Ang
Edith Cowan University

10.1109/ICOS.2012.6417629

Modular Dynamic RBF Neural Network for Face Recognition

Sue Inn Ch'ng, Kah Phooi Seng
Department of Computer Science & Networked Systems
Sunway University
Selangor, Malaysia

Li-Minn Ang
School of Engineering
Edith Cowan University
Australia

Abstract—**Over the years, we have seen an increase in the use of RBF neural networks for the task of face recognition. However, the use of second order algorithms as the learning algorithm for all the adjustable parameters in such networks are rare due to the high computational complexity of the calculation of the Jacobian and Hessian matrix. Hence, in this paper, we propose a modular structural training architecture to adapt the Levenberg-Marquardt based RBF neural network for the application of face recognition. In addition to the proposal of the modular structural training architecture, we have also investigated the use of different front-end processors to reduce the dimension size of the feature vectors prior to its application to the LM-based RBF neural network. The investigative study was done on three standard face databases; ORL, Yale and AR databases.**

Keywords- modular structure, RBF neural networks, Levenberg-Marquardt algorithm, face recognition

I. INTRODUCTION

The use of radial basis function (RBF) neural network for face recognition has been widely explored over the past decade with promising results [1-8]. Most of these works are motivated by the findings of the work by Joo et.al in [2] that study the use and adaptation of the RBF neural network for the small sample high dimensional problem of face recognition. The authors in [2] used Principal Component Analysis (PCA)[9] and Fisher Linear Discriminant (FLD) [10] to reduce the dimension of the face images and extract the discriminant features. Subsequently, the patterns are classified using an RBF neural network learned using a hybrid algorithm; Linear Least Squares (LLS) and Gradient Descent (GD). LLS is used to learn the weights of the network while GD is used to learn the adjustable parameters; width and centers, of the RBF units. To-date, the method of structure determination employed by these authors is still widely practiced in the RBF design of some of the recently developed face recognition systems [1, 3, 6-8]. For example, the work by [7], uses the structure determination method by [2], with regularized orthogonal least square (ROLS) to create an RBF neural network with incremental learning capabilities for the task of face recognition. On the other hand, the work by [1, 3, 6, 8] uses different feature extraction/pre-processing techniques prior to applying the data to the RBF neural network designed using the same concept specified in [2].

Compared to first order algorithms, second order algorithms are advantageous as they are much faster [11, 12] and can be an effective solution for problems with up to hundreds of parameters [13]. Thus, instead of using two separate algorithms e.g. LLS for weights and GD for width and centers, the second-order algorithm can be theoretically used to learn all the adjustable parameters in the RBF neural network. This is evident in the recent work by [11, 14] which uses different improved methods of computation to determine the Jacobian matrices in the Levenberg-Marquardt (LM) algorithm and subsequently use the improved LM algorithm to determine all the adjustable parameters in a single feed-forward neural network. Although the above-mentioned approaches were tested to be effective on the two commonly used benchmark tests; function approximation and two-spiral classification, the application of such methods to high dimensional multi-class learning has not been explored. To the best of the authors' knowledge, it is noted that the use LM algorithm for the learning of the RBF neural networks designed for the task of face recognition is scarce. This could be due to the fact that the high computational complexity incurred by the learning algorithm itself eventually far outweighs any advantages offered by the algorithm resulting in most researchers opting for a less computationally demanding algorithm.

In this paper, the modified LM algorithm to update all the adjustable parameters in the RBF neural network is first presented. This is followed by the proposal of the modular dynamic LM-based RBF neural network, which is based on the network construction method presented in [11]. Then, the proposed modular dynamic LM-based RBF (MD-RBF) is coupled with several front-end processing methods to determine which method complements the performance of the proposed MD-RBF neural network. The contribution of this paper is two-fold; 1) the proposal of the MD-RBF neural network for high dimensional multi-class learning and 2) performance analysis of a combination of different front-end processing methods with the MD-RBF neural network .

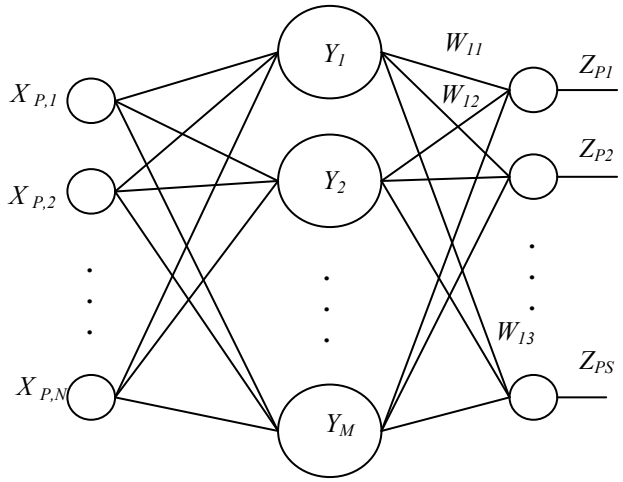

Figure1. Architecture of RBF neural network

The structure of the paper is as follows: In Section 2, a brief review of the RBF neural network architecture is presented. This is followed by the presentation of the use of LM algorithm to update all the adjustable parameters for RBF neural networks. In Section 4, a brief analysis of the issues faced by high dimensional multi-class learning for LM-based RBF neural network is presented. A description of the proposed modular structural training architecture to adapt the LM-based RBF neural network for high dimensional multi-class learning is presented in Section 5. Section 6 describes the general method of application of the proposed MD-RBF neural network for face recognition. The simulation results obtained are presented and discussed in Section 7. Finally, the paper is concluded in Section 8.

II. RBF Neural Network Architecture

The RBF neural network consists of three layers; input layer, hidden layer and output layer. Fig. 1 shows the architecture of a typical RBF neural network with N inputs, M RBF units and S outputs. The data set is denoted by X_P where P denotes the total number of patterns. The output of the RBF network with Gaussian function can be calculated as follows:

$$Z_{ps} = \sum_{m=1}^{NM} \sum Y_m\left(X_p\right)\ W_{m,s} + w_0 \qquad (1)$$

$$Y_m\left(X_p\right) = \exp\left(-\frac{\left\|X_p - C_m\right\|^2}{\sigma_m}\right) \qquad (2)$$

where $W_{m,s}$ denote the weights between the M-th RBF unit and S-th output, w_0 the bias, C_m the center and σ_m the width. Note that $\|\bullet\|$ represents the Euclidean norm.

III. Modified LM Algorithm for RBF Neural Network

The update rule of the LM algorithm commonly used to update the weights of artificial neural networks is provided in (3) [12].

$$\begin{aligned} w_{t+1} &= w_t - (J^T J + \mu I)^{-1} J^T e \\ &= w_t + \delta_t \quad where \quad \delta_t = -(J^T J + \mu I)^{-1} J^T e \end{aligned} \qquad (3)$$

where J is the Jacobian matrix, I is the identity matrix, μ is the learning rate and e is the error vector containing the output errors for each input vector used on training the network. In the case of updating the weights, the Jacobian matrix is calculated based on the first-order partial derivatives of the network error with respect to the weights. The Jacobian matrix in such cases for P number of patterns, S number of outputs and NW total number of weights results in a Jacobian matrix of size $(P \times S) \times NW$ and is mathematically represented as:

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \cdots & \frac{\partial e_{11}}{\partial w_{NW}} \\ \frac{\partial e_{12}}{\partial w_1} & \frac{\partial e_{12}}{\partial w_2} & \cdots & \frac{\partial e_{12}}{\partial w_{NW}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1S}}{\partial w_1} & \frac{\partial e_{1S}}{\partial w_2} & \cdots & \frac{\partial e_{1S}}{\partial w_{NW}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P1}}{\partial w_1} & \frac{\partial e_{P1}}{\partial w_2} & \cdots & \frac{\partial e_{P1}}{\partial w_{NW}} \\ \frac{\partial e_{P2}}{\partial w_1} & \frac{\partial e_{P2}}{\partial w_2} & \cdots & \frac{\partial e_{P2}}{\partial w_{NW}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{PS}}{\partial w_1} & \frac{\partial e_{PS}}{\partial w_2} & \cdots & \frac{\partial e_{PS}}{\partial w_{NW}} \end{bmatrix} \qquad (4)$$

To modify the update rule in (3) to update all the parameters, we incorporate the centers and width into the update rule by concatenating all the parameters into a single vector, R, and substituting it into the original equation. The formation of vector R can be represented by (5) [11].

$$R = [W_{M,J} \mid C_M \mid \sigma_M] \tag{5}$$

Now, the new update rule is as follows:

$$R_{t+1} = R_t - (J^T J + \mu I)^{-1} J^T e_{PS} \tag{6}$$

Where R_{t+1} represents the updated parameters, J represents the Jacobian matrix, μ the learning parameter and e_{PS} the error vector defined in (8). For the original algorithm, since the update rule is used for solely determining the weights, hence the Jacobian matrix is formed by differentiating the error with respect to weights, refer to (4). In the current case, since the centers and width are incorporated into the update rule, the Jacobian matrix is formed by differentiating the error with respect to weights, $\frac{de_{PS}}{dw_{M\,s}}$, centers $\frac{de_{PS}}{dC_{N,M}}$ and width $\frac{de_{PS}}{d\sigma_M}$ respectively and concatenating the results together .

$$J_{new} = \left[\frac{de_{PS}}{dw_{M\,s}} \mid \frac{de_{PS}}{dC_{N,M}} \mid \frac{de_{PS}}{d\sigma_M} \right] \tag{7}$$

$$e_{PS} = d_{PS} - Z_{PS} \tag{8}$$

$$\frac{de_{PS}}{dw_{M,S}} = \frac{de_{PS}}{dZ_{PS}} \bullet \frac{dZ_{PS}}{dw_{M,S}} = -Y_M(X_P) \tag{9}$$

$$\frac{de_{PS}}{dC_{N,M}} = \frac{de_{PS}}{dZ_{PS}} \bullet \frac{dZ_{PS}}{dw_{M,S}} \bullet \frac{dw_{M,S}}{dC_{N,M}} = -\frac{2w_{M,S} Y_M(X_P)(X_{P,N} - C_{N,M})}{\sigma_M} \tag{10}$$

$$\frac{de_{PS}}{d\sigma_M} = \frac{de_{PS}}{dZ_{PS}} \bullet \frac{dZ_{PS}}{dw_{M,S}} \bullet \frac{dw_{M,S}}{\sigma_M} = -\frac{w_{M,S} Y_M(X_P) \left\| (X_{P,N} - C_{N,M}) \right\|^2}{\sigma_M^2} \tag{11}$$

where e_{PS} is the error vector and d_{PS} is the desired output at network output S for training pattern P. The calculation of the Jacobian elements can be calculated using the differential chain rule which results in (9)-(11).

With the addition of the differential of the center and width to the original Jacobian matrix, the size of the new Jacobian matrix is now larger. The size of the new Jacobian matrix is represented by (12), where P represents the number of patterns, S the number of outputs, M the number of neurons and N the number of inputs. To remove the need of storing the entire Jacobian matrix during each update, the concept of calculating the Hessian matrix and gradient in [15] is used whereby only the rows of the new Jacobian matrix needs to be calculated and stored. The Hessian matrix, Q, is then formed by summing the result of multiplication between the Jacobian rows, j, with its transposed, j^T; refer to (13). On the other hand, the gradient, g, is obtained by summing and multiplying the Jacobian row with the error vector, e_{PS}, refer to (14).

$$Size(J_{new}) = (P \times S) \times [(S + N + 1) \times M] \tag{12}$$

$$Q = J^T J = \sum_{P=1}^{P} j_{PS}^T j_{PS} \tag{13}$$

$$g = \sum_{P=1}^{P} j_{PS}^T e_{PS} \tag{14}$$

IV. LM Algorithm for High Dimensional Multi-Class Learning

For each iteration in the LM algorithm, the inversion of the Hessian matrix, H, is required. This becomes a problem for high dimensional multi-class learning using LM-based RBF neural networks because with the use of the LM algorithm to update all the adjustable parameters, the formulation of the Hessian matrix is now dependent on the size of the inputs and outputs. Denoting the number of inputs as NI, total number of outputs as NO and the number of neurons as M; the size of the Hessian matrix can be represented as follows.

$$\text{Size}(H) = (NI + 1 + NO) \cdot M \tag{15}$$

Hence, for the high dimensional multi-class learning problem, NI and NO which depends on the size of feature vector and number of classes to be classified respectively, will inevitably increase resulting in the inversion of a larger matrix during each update of the learning process. To reduce the computational complexity related to the inversion of the Hessian matrix during each iteration, steps can be taken to reduce both the size of feature vectors and the number of outputs. The latter is a much trickier problem because unlike the size of the feature vector which can easily be reduced through the use of pre-processing or dimension reduction techniques, the number of classes to be classified cannot be arbitrarily reduced just to reduce the number of outputs. In this paper, we propose the use of a modular training structure to mitigate the problem of reducing the number of outputs used by the network structure. The proposed modular structural training architecture is presented in the following section.

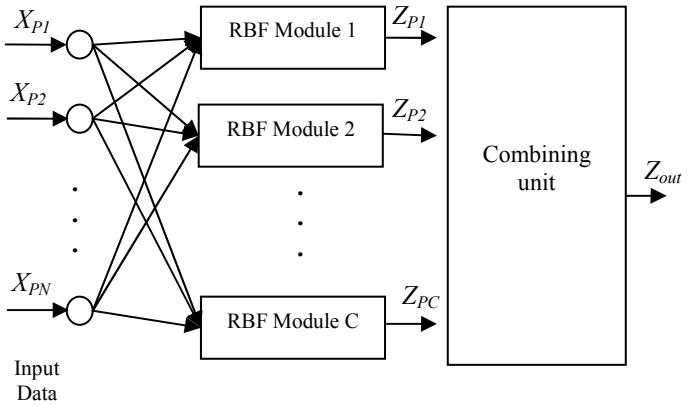

Figure 2. MD-RBF network architecture

V. Proposed Modular Dynamic RBF Neural Network (MD-RBF)

The architecture of the proposed modular training structure for the dynamic LM-based RBF neural network (MD-RBF) is depicted in Fig. 2. The structure of the proposed MD-RBF neural network is similar to that of the conventional RBF neural network shown in Fig. 1 except that the training of each class is done in modules resulting in the classification of only one class at a time. Then, the individual outputs of each module are concatenated together to form the final output of the MD-RBF neural network, Z_{out}.

Thus, for the training of an arbitrary number of classes, the number of inputs for each RBF module is equals to that of the number of features (i.e. the dimension of the input space) but each module only has one output and the total number of modules in the network is equals to the total number of classes to be classified. For example, denoting the input data as X_{PN} for P number of patterns and N number of inputs, the classification of C classes using MD-RBF neural network will require C number of modules. The output of the C-th module, Z_{PC}, is given as follows:

$$Z_{PC} = \sum_{m=1}^{M} \sum Y_m(X_P) \; W_m + w_0 \tag{16}$$

$$Y_m(X_P) = \exp\left(- \frac{\left\| X_{P,N} - C_m \right\|^2}{\sigma_m} \right) \tag{17}$$

$$Z_{out} = [Z_{P1} \mid Z_{P2} \mid Z_{P3} \ldots Z_{PC}] \tag{18}$$

where M the total number of neurons, W_M the weights between the M-th neuron and the single output, w_0 the bias, C_M the center and σ_M the width. Note that $\|\bullet\|$ represents the Euclidean norm. The growth process of the dynamic RBF neural network presented in [11] is used for the network construction in each RBF module while the learning algorithm presented in Section III is used for the training of each of the RBF module. The details of the growth process of the network will not be described in this paper. Through the use of the proposed modular structural training, the size of the Hessian to be inverted during each iteration for each RBF module will now be only dependent on the size of the feature vectors, which subsequently translates to the number of inputs required (NI), refer to equation (19).

$$\text{Size } (H) = (NI + 2) \times M \tag{19}$$

VI. Application to Face Recognition

The use of a modular structure is able to reduce the size of the Hessian matrix and hence the overall computation of the training of such LM-based RBF networks to be independent of the number of outputs. Nonetheless, the computation of the Hessian matrix of the modular dynamic RBF neural network is still dependent on the size of feature vectors (number of inputs). This aspect can be resolved by the application of dimension reduction methods or feature selection techniques to ensure that only the most significant features are applied to the MD-RBF neural network for classification. The general block diagram of a face recognition expert using the proposed MD-RBF neural network as a classifier is depicted in Fig. 3.

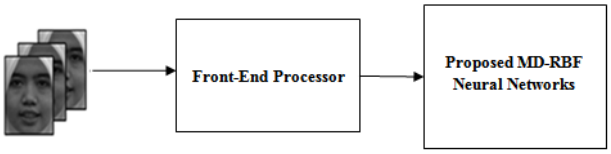

Figure 3. General block diagram of a face recognition expert using the proposed MD-RBF neural network

The front-end processor block set can be replaced with any feature extractor or pre-processing techniques.

VII. Results and Discussion

In this section, the use of multiband curvelet technique [16], PCA+LDA[2], deep features[17] and block-based pre-processing [18] as the front-end processor is investigated to determine the type of front-end processor that complements the performance of the proposed MD-RBF neural network. The recognition performance for each of the face recognition expert using the different type of front-end processor and the feature vector dimension is used as the measurement index. Our investigative study is done on three standard face databases namely the ORL database, Yale database and AR database.

The Yale database consists 15 individuals where for each individual, 11 images containing varying illumination, facial expressions and glasses were taken. The illumination variation contained in the Yale database is limited to either from the left, from the right or centre. From these 11 face images, three images with neutral expression and even lighting were used for training while the remaining images were for testing.

The ORL database contains a set of face images of males and females taken between April 1992 and April 1994 at Olivetti Research Laboratory in Cambridge, UK. There are a total of ten different images of 40 subjects in the database. These images contain facial expression variations and perspective variations. In our experiments, five images were randomly selected for training and the remaining five images were used for testing.

The AR database consists of face images of 126 people taken in two sessions on two different days. For each subject, the frontal view faces featuring different facial expressions, mild illumination effect and occlusions were taken. The same pictures were taken in both sessions. However, in our experiments, a subset of 100 people (50 males and 50 females) out of the entire database was used. For the evaluation of expression variation using this database, two images showcasing neutral expression for each subject were used for training whereas three images with varying facial expressions were used for testing. On the other hand, for the evaluation of illumination variation using this database, three images with even illumination per subject were used for training whereas four images with varying illumination were used for testing. In the following text, the test set that contains expression variation is denoted as AR-Expression whereas the test set that contains illumination variation is denoted as AR-Illumination.

All the images used in the simulation were scaled to 32×32 pixels before training. A summary of the total number of subjects, number of training images and number of testing images that were used for each database in the following experiments is tabulated in Table I. For the simulation of the face expert system using deep features, the layer sizes of the deep belief networks were set to: (8×8)-60-60-60-(no. of subjects). The weighted sum fusion rule is used to fuse the individual results of all blocks in both the block-based MD-RBF face expert and deep features face expert. The performance evaluation of the proposed face experts on all three databases are shown in Table II. A comparison of the feature sizes of the different front-end processor is tabulated in Table III.

TABLE I: Summary of database settings used in the experiments

Database	**AR Expression**	**AR Illumination**	**Yale**	**ORL**
Number of subjects	100	100	15	40
Number of training images	200	200	45	200
Number of testing images	300	400	120	200

TABLE II: Recognition performance (%) for each of the front-end processor with the proposed MD-RBF neural network

	Database			
Front-End Processor	***Yale***	***ORL***	***AR_Expression***	***AR_Illumination***
Deep features	96.7	94.0	87.0	86.8
Block-based	99.2	94.5	97.0	92.3
Multiband Curvelet	94.2	94.0	91.3	90.8
PCA+LDA	93.3	96.5	88.0	84.5

TABLE III: Comparison of dimensions of feature vector for different front-end processors

Front-End Processor	**Feature Vector Dimension**
Deep features	60
Block-based	64
Multiband Curvelet	562+420
PCA+LDA	no. of subjects-1

From the results obtained in Table II, the front-end processor using the block-based technique yields the best performance for all three databases except for the ORL database. This could be due to the fact that the ORL database contains slight pose variations which are not present in the other two databases. In addition to the superior performance exhibited by the block-based MD-RBF face expert, by dividing the face images into smaller block prior to training, the feature size input to the proposed MD-RBF neural network is greatly reduced compared to the multiband curvelet technique and PCA+LDA with the latter's feature size highly dependent on the size of total subjects, refer to Table III. This can be disadvantageous when the number of subjects to be trained is large e.g. large-scale population recognition, as the size of the features will increase exponentially and this will affect the performance and the training duration of the proposed MD-RBF neural network.

VIII. CONCULSIONS

In this paper, we have presented a modular structural training architecture to adapt the LM-based RBF neural network, to the application of face recognition. This results in the proposal of the MD-RBF neural network which uses the growth process presented by [11] to create a compact neural network in each of the RBF modules. For the application of face recognition, the computational complexity of the LM-based RBF neural network training is further reduced through the use of front-end processors to extract and reduce the dimension of the feature vectors to be applied to the MD-RBF neural network. Thus, we have investigated the use of different pre-processing and dimension reduction methods in this paper. Our simulation results shows that the front-end processor using block-based processing yields the best recognition performance majority of the test sets and has the second smallest feature vector dimension compared to the other front-end processors.

REFERENCES

[1] B.-J. Oh, "Face Recognition using Radial Basis Function Network based on LDA," *World Academy of Science Engineering and Technology,* vol. 7, pp. 255-259, 2005.

[2] E. Meng Joo*, et al.*, "Face recognition with radial basis function (RBF) neural networks," *Neural Networks, IEEE Transactions on,* vol. 13, pp. 697-710, 2002.

[3] J. Haddadnia*, et al.*, "A hybrid learning RBF neural network for human face recognition with pseudo Zernike moment invariant," in *Neural*

Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on, 2002, pp. 11-16.

[4] S. Thakur, *et al.*, "Face Recognition by Combination of RBF Neural Networks Using Dempster-Shafer Theory," presented at the Proceedings of the International Conference on Computing: Theory and Applications, 2007.

[5] V. Radha and N. Nallammal, "Neural Network Based Face Recognition Using RBFN Classifier," presented at the World Congress on Engineering and Computer Science (WCECS 2011), San Francisco, USA, 2011.

[6] W. Wang, "Face Recognition Based on Radial Basis Function Neural Networks," presented at the Proceedings of the 2008 International Seminar on Future Information Technology and Management Engineering, 2008.

[7] Y. W. Wong, *et al.*, "Radial Basis Function Neural Network With Incremental Learning for Face Recognition," *IEEE Transactions on Systems, Man, and Cybernetics-Part B Cybernetics,* vol. 4, pp. 940 - 949, 2011.

[8] Y. M. Wu, *et al.*, "Research of Face Recognition Based on LLE and RBF Neural Network," in *Electrical and Control Engineering (ICECE), 2010 International Conference on*, 2010, pp. 1605-1608.

[9] M. Turk and A. Pentland, "Eigenfaces for face recognition," *Journal of Cognitive Neuroscience,* vol. 3, pp. 71-86, 1991.

[10] P. N. Belhumeur, *et al.*, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 19, pp. 711-720, 1997.

[11] Y. Hao, *et al.*, "Advantages of Radial Basis Function Networks for Dynamic System Design," *Industrial Electronics, IEEE Transactions on,* vol. 58, pp. 5438-5450, 2011.

[12] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *Neural Networks, IEEE Transactions on,* vol. 5, pp. 989-993, 1994.

[13] N. Ampazis and S. J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," *Neural Networks, IEEE Transactions on,* vol. 13, pp. 1064-1074, 2002.

[14] P. Jian-Xun, *et al.*, "A New Jacobian Matrix for Optimal Learning of Single-Layer Neural Networks," *Neural Networks, IEEE Transactions on,* vol. 19, pp. 119-129, 2008.

[15] B. M. Wilamowski and Y. Hao, "Improved Computation for Levenberg-Marquardt Training," *Neural Networks, IEEE Transactions on,* vol. 21, pp. 930-937, 2010.

[16] S. I. Ch'ng, *et al.*, "Multiband Curvelet-based technique for Audio-visual Recognition over Internet Protocol," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST),* vol. 1, pp. 53-60, 2011.

[17] T. Liu, "A novel text classification approach based on deep belief network," presented at the Proceedings of the 17th international conference on Neural information processing: theory and algorithms - Volume Part I, Sydney, Australia, 2010.

[18] S. I. Ch'ng, *et al.*, "Block-based Deep Belief Networks for Face Recognition," *International Journal of Biometrics,* vol. 4, pp. 130-143, 2012.