A Comparison of Neural Networks for Real-Time Emotion Recognition from Speech Signals

MEHMET S. UNLUTURK, KAYA OGUZ, COSKUN ATAY
Department of Software Engineering
Izmir University of Economics
Sakarya Cad No.156, Balcova, Izmir 35330
TURKEY
suleyman.unluturk@ieu.edu.tr kaya.oguz@ieu.edu.tr coskun.atay@ieu.edu.tr

Abstract: Speech and emotion recognition improve the quality of human computer interaction and allow easier to use interfaces for every level of user in software applications. In this study, we have developed two different neural networks called emotion recognition neural network (ERNN) and Gram-Charlier emotion recognition neural network (GERNN) to classify the voice signals for emotion recognition. The ERNN has 128 input nodes, 20 hidden neurons, and three summing output nodes. A set of 97920 training sets is used to train the ERNN. A new set of 24480 testing sets is utilized to test the ERNN performance. The samples tested for voice recognition are acquired from the movies “Anger Management” and “Pick of Destiny”. ERNN achieves an average recognition performance of 100%. This high level of recognition suggests that the ERNN is a promising method for emotion recognition in computer applications. Furthermore, the GERNN has four input nodes, 20 hidden neurons, and three output nodes. The GERNN achieves an average recognition performance of 33%. This shows us that we cannot use Gram-Charlier coefficients to discriminate emotion signals. In addition, Hinton diagrams were utilized to display the optimality of ERNN weights.

Key-Words: Back propagation learning algorithm, Neural network, Emotion, Speech, Power Spectrum, Fast-Fourier Transform (FFT), Bayes optimal decision rule.

1 Introduction

Speech is one of the oldest tools humans use for interaction among each other. It is therefore one of the most natural ways to interact with the computers as well. Although speech recognition is now good enough to allow speech to text engines, emotion recognition can increase the over all efficiency of interaction and may provide the users a more comfortable user interface.

In most cases humans are able to determine the emotion of the speaker and adjust their behavior accordingly. Similarly, emotion recognition will give the programmer a chance to develop an artificial intelligence to meet the speaker's feelings in a way they can be used in many scenarios from computer games to virtual sales-programs. Getting to know how emotion is encoded in speech signals may also improve the reverse process, where the speech is generated by the computer.

Since a formula cannot be devised for the problem, using neural networks is considered to be an appropriate choice. It is an advantage of neural networks that a trained one is quick to respond since in most applications the emotion needs to be determined instantly. The training may take a long time but is irrelevant as it is mostly done off-line.

In the experiments, three base emotions, angry, happy and neutral are taken into account. Various speech sets that belong to these emotion groups are extracted from different movies and used for training and testing. Phrases from movies are selected instead of using amateur actors to read given sentences because trained actors do a lot better job in controlling their voice and convincing the audience. Therefore they are the closest available recorded samples to original emotions.

Two neural networks are designed using two different techniques. In the emotion recognition neural network (ERNN) the wave data is transformed from time domain to frequency domain and their power spectrums are generated. In the Gram-Charlier emotion recognition neural network (GERNN) the Gram-Charlier coefficients of the wave data are presented as input. Among

these two neural networks, the ERNN is capable of distinguishing the samples successfully.

For both training and testing, MATLAB environment is used. MATLAB has extensive tools for both reading wave files and creating and testing neural network simulations.

Emotion recognition is not a new topic and both research and applications exist using various methods most of which require extracting certain features from the speech [1,2,3,4,5,6]. These features can be related to the signal, pitch and formants. The methods used for emotion recognition range from artificial neural networks to hidden Markov models and Radial Basis Function Networks.

This paper is organized as follows; in part 2 previous and similar works are summarized, part 3 is about ERNN, part 4 is about GERNN, part 5 talks about results and discussion, part 6 includes conclusion and future work.

2 Existing Approaches

As mentioned earlier in the introduction, emotion recognition is not a new topic by any means. In this brief chapter, it would be appropriate to cover some of the approaches and compare them to this research.

Dellaert, Polzin and Waibel [3] use several statistical pattern recognition techniques to recognize emotion in speech. Taking four categories, namely happy, sad, anger and fear, into consideration, they used methods such as Maximum Likelihood Bayes Classifier, Kernel Regression and K-nearest neighbors using utterances from amateur actors. Dellaert et al., are focused on their approach on extracting features from speech.

Lee et al. [4], focused on a negative emotion, anger, for call centers. With the data that is generated by amateur actors, they have used Back-propagation Network with acoustic features and compared it to Decision Tree C5.0.

Sharma et al. [5], unlike the previous one mentioned and like this study, used utterances from movie sequences. They took three emotions, anger, happiness and sorrow into consideration. Using Bayesian Classifier and Radial Basis Function approaches, they have satisfactory results as well.

Feature extraction plays an important role in determining the emotion in remaining papers, too. However, humans use other clues as well to get the emotion of the speaker. They get better results if the speaker is someone they are familiar with, or if the speaker is talking in a language they can understand. They can also take clues in the words the speaker chooses or how fast or loud he/she speaks. All of this information can be included in emotion recognition and they are likely to increase the rate of recognition.

This research, however, is focused solely on the sampled data, rather than extracting certain features. This is mostly because the basic emotions that are considered; angry, happy and neutral usually are easier to distinguish, even if they are spoken in a language unfamiliar to the listener. This approach also removes the importance of gender, speaker, language and the words uttered, leading to a more general approach to the problem.

Another important part of the approach presented in this research is that, instead of taking the whole utterance for recognition, the speech samples are trained and tested thoroughly using shorter but constant length windows, which in return give a percentage of the results. The speech is of arbitrary length and therefore in an application, instead of trying to determine how long the utterance will last, scanning it at constant lengths with constant steps removes the need to do so. It can also react to rapid changes of emotion.

It should also be noted that, this is an initial research which will need more varied testing with more varied data. However, the initial tests yield promising results.

3 ERNN

Emotion Recognition Neural Network (ERNN) uses the power spectrum data which is generated using Fast Fourier Transform algorithm from the original sample data. This data is then used to train the back-propagation neural network.

3.1 Design

To extract the emotion signatures inherent to voice signals, the back propagation-learning algorithm [7,8,9,10] is used to design the emotion recognition neural network.

The block diagram of ERNN is shown in Figure 1. Segmented data is applied to the input of the power spectrum processor utilizing the Fast Fourier Transform algorithm. The output of it is normalized and is presented to a three layer, fully interconnected neural network for classification. The output layer of the neural network is inputted by the weighted sum of outputs of the hidden and bias nodes in the hidden layer. These weighted inputs are processed by a hyperbolic tangent function. A set of desired output values is then compared to the estimated outputs of the neural network for every set of input values of the power spectrum of the voice signals. The weights are appropriately updated by back propagating the gradient of the output error through the entire

neural network.

The experimental data, used for both training and testing the ERNN, is obtained from movies in WAV format at 48000 Hz with only one channel (mono). Each experimental data segment is composed of 65536 data points.

Removing the mean and dividing it with the standard deviation normalize the segmented power spectrum data. This normalization is highly desirable because it desensitizes the neural network to the signal offset and/or signal gain. Normalization is given as

$$X_p(k\Omega) = \frac{R_p(k\Omega) - \mu}{\sigma}, k = 1, \cdots, K$$
$$\mu = \frac{1}{K}\sum_{k=1}^{K} R_p(k\Omega), \quad (1)$$
$$\sigma = \sqrt{\frac{1}{K-1}\sum_{1}^{K}(R_p(k\Omega) - \mu)^2},$$

where $R_p(k\Omega)$ (Ω is the frequency sampling interval) is the segmented power spectrum of the voice signal, μ and σ are the estimated mean (i.e., signal offset) and the estimated standard deviation (i.e., measurement scale) respectively.

The input signal to the power spectrum block (see Figure 1) is created using a sliding window. The size of the sliding window is 256 samples, and the step between two successive windows is 8 samples. The first set is taken from the beginning of the voice data. The second set is 8 samples to the right of the first set, and this is repeated until the window covers the entire 65536 samples of the voice signal. The power spectrum, $R_p(k\Omega)$, of each input set is calculated by

$$R_p(k\Omega) = FFT(x(nT)) \bullet Conj(FFT(x(nT))) \quad (2)$$

where $x(nT)$ is the sampled value of the voice signal, T is the time sampling interval, FFT is the Fast Fourier Transform, Ω is the frequency-sampling interval. The first 128 samples of $R_p(k\Omega)$ which span the entire frequency range is taken into consideration as an input to the neural network for signal classification. Hence, the neural network needs 128 inputs and 3 neurons in its output layer to classify the voice signals. The hidden layer has 20 neurons. This number was picked through experimentation and experience.

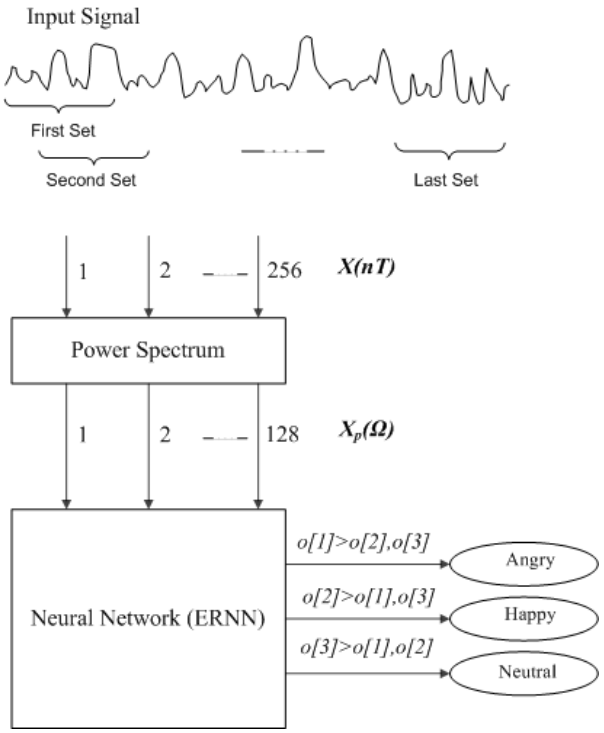

Figure 1. Block diagram of the emotion recognition neural network (ERNN) system. The output neuron that has the highest value indicates which type of emotion signal is present at the input.

During testing, ERNN accomplished a recognition performance of 100% with that many hidden neurons. And also, if the network has trouble classifying, then additional neurons can be added to the hidden layer. Figure 2 shows a three layer neural network that is designed to classify the voice signal power spectrum. The hidden neuron's output of this neural network, y_j, is given as

$$y_j = \phi(\sum_{i=1}^{N} X_{Pi} w_{ji}^{\ h} + \theta^j) \quad (3)$$

where the activation function for the hidden and output layer, $\phi()$, is a tangent hyperbolic function defined as

$$\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

The term X_p is the normalized input vector, w_{ji}^{h} is the weight from j^{th} hidden neuron to the i^{th} input neuron.

$$X_p = [X_p(\Omega), \cdots, X_p(i\Omega), \cdots, X_p(N\Omega)] \quad (5)$$

Back propagation algorithm is used to estimate the hidden layer and output layer weights and biases for the optimal design of ERNN. Then, the output vector *o*, is applied to the decision block in order to classify the voice signal,

$$\begin{aligned} o[1] > o[2], o[3] &\longrightarrow Angry \\ o[2] > o[1], o[3] &\longrightarrow Happy \\ o[3] > o[1], o[2] &\longrightarrow Neutral \end{aligned} \quad (6)$$

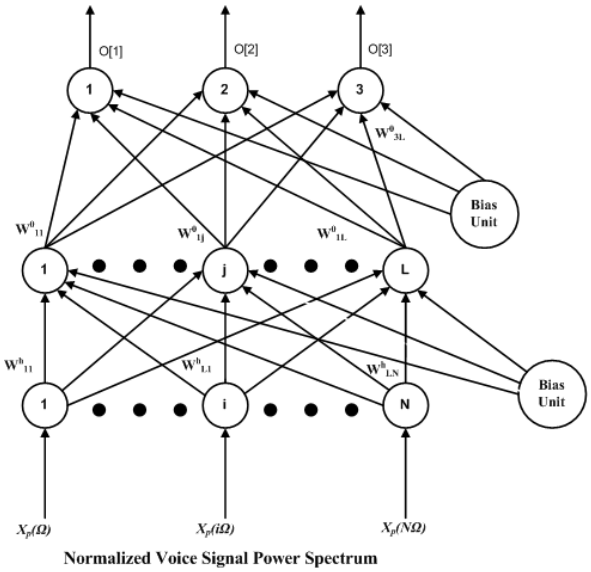

Figure 2. A three-layer model of emotion recognition neural network (N=128 and L=20).

Before the creation or training of the neural network, the samples and their outputs are prepared. As each file is read, they are positioned in the input matrix, as the correct outputs are positioned in the output matrix.

When the files are read through built-in MATLAB functions, they are put in arrays that have values between -1 and 1. This array is then scanned by a window, which is 256 bytes long, with a step of 8 bytes. Each file is scanned thoroughly, and they are read up to the 16th power of 2, 65536; as a result of this process, each file gives 8160 samples.

$$sampleCount = \frac{fileLength - windowSize}{step} \quad (7)$$

The input matrix is generated in a way that the columns follow the angry, happy and neutral samples with the output matrix corresponding to the related values to help the back propagation algorithm to give better results.

Before placing the samples into the matrix, they are transformed from time domain to frequency domain using the Fast Fourier Transform. After the transformation, their power spectrums are generated. This power spectrum is symmetric so the first half is placed into the matrix. With four WAV files for each emotion that is scanned to give more than 8K samples, the input matrix is made up of almost 100K columns and 128 rows. The output matrix has 3 rows.

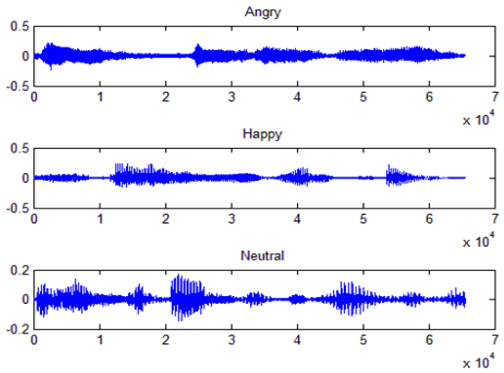

Figure 3. Samples from each emotion

Once the matrices are ready, they are fed into the neural network for training. After the training is completed, the samples are simulated on the neural network. The samples are also taken through the same procedure and transformed to frequency domain. As there are almost 8K simulations in one test, the number of correct answers as percentage is taken as a final result (see Equation 6).

3.2 Implementation Details

As mentioned earlier, MATLAB software is used to read in the data from WAV files. MATLAB has a built-in function, wavread, which can easily read and sample the data. Equation (7) is used to walk through the data and apply the Fast Fourier Transform, using the fft function, to get the power spectrum of the data. This data is then copied to the matrix which will be used to train the neural network. A piece of sample code is shown below.

```
piece_angry = wave_file(start: end);
ff_a = fft(piece_angry);
pw_a = ff_a .* conj(ff_a);
res_a = pw_a(1:window/2);
P(:,counter) = res_a';
T(:,counter) = TT(:,1);
```

Here, the matrix P holds the data which will be used to train the network and T holds the output values. Since the power spectrum is symmetric, the first half of the data is taken into account.

Figure 4 and Figure 5 display training and state graphs respectively for ERNN.

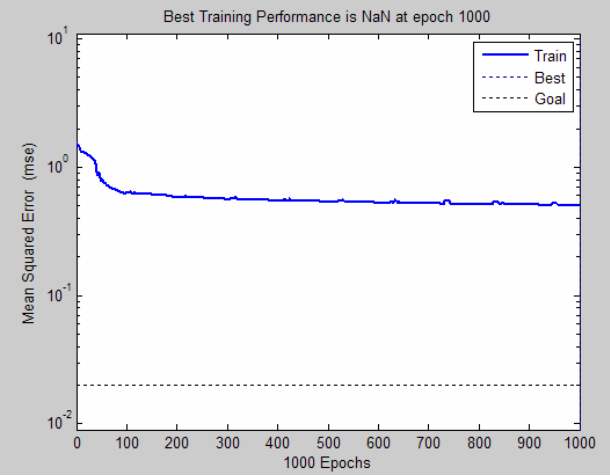

Figure 4. Training graph for ERNN at 1000 epochs using MATLAB.

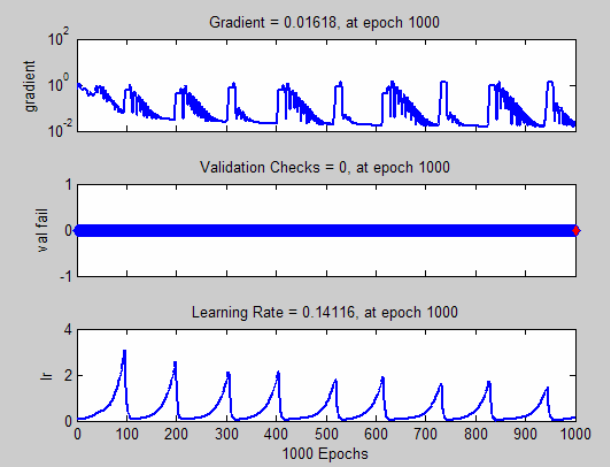

Figure 5. Training state graphs for ERNN at 1000 epochs using MATLAB

The success of ERNN may be obscured if there is no understanding of the optimality of the neural network. In this study Hinton diagrams are used to explain the optimal characteristics of the neural network. Hinton diagrams are named for Geoffrey Hinton who used this method to display neural network weights. In the Hinton diagrams, the size of each square is proportional to the strength of each weight. The green color indicates a positive magnitude weight, and red color indicates a negative magnitude weight. Instead of displaying the weights between 128 input nodes and 20 hidden nodes, we chose to display the input nodes whose locations are between 53 and 72 for clarity purposes. Figure 6 depicts the Hinton diagram of the ERNN. The first row shows the weights from the input nodes whose locations are between 53 and 72 to the first hidden neuron, the second row depicts the weights from the same range of input nodes to the second hidden neuron, and so on. Note that the number of rows in Hinton diagrams is equal to the number of hidden neurons which is 20 in ERNN.

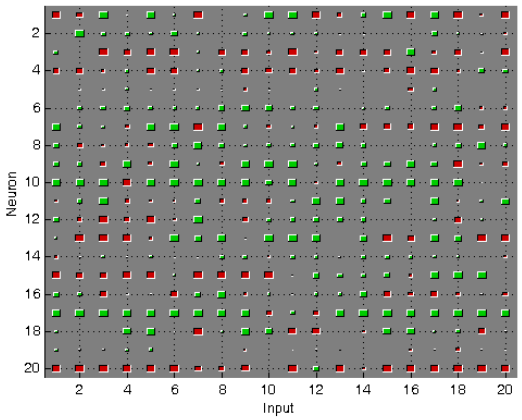

Figure 6. Hinton Diagram for Hidden Layer Weights

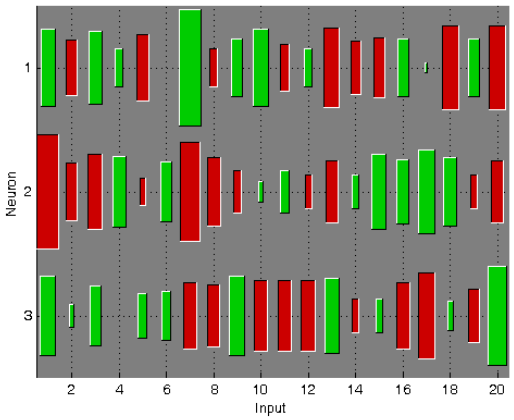

Figure 7. Hinton Diagram for Ouput Layer Weights

Figure 7 depicts the weights from the hidden neurons to the output nodes. Since there are three output nodes, the number of rows in this Hinton diagram is 3. Again, the first column depicts the weights between the output nodes and the first hidden neuron. Inspection of these Hinton diagrams reveals that all hidden nodes contribute to the decision process and that the optimal design of the neural network has been realized. Furthermore, there is no situation suggesting that a set of weights yields very large values, very low values, or predictable values.

ERNN has the following properties:

1. the response is real-time once it is trained,
2. all hidden neurons have the same structure which makes its hardware implementation easy using VLSI design techniques [11, 12].

Based on experimental observations, ERNN can be efficiently used in discrimination process of emotions inside the voice signals.

Next section deals with another neural network approach to our emotion recognition problem. With this new neural network model, we will try to achieve the ERNN's recognition performance goal while reducing the number of input nodes from 128 to four.

4 GERNN

Gram-Charlier Emotion Recognition Neural Network (GERNN) uses Gram-Charlier coefficients of the speech signal to determine its emotion.

4.1 Design

The wave files that are used in the trainings can be considered as a row vector, and thus can be represented as

$$\vec{X}=[x(1), x(2),...x(m)] \quad (8)$$

where m is 65536. Each element in the vector corresponds to a value between -1 and 1. Then, there are three emotion classes from which our input wave can belong to;

H_0 = Angry,
H_1 = Happy,
H_2 = Neutral.

The prior probability of an unknown wave file belonging to emotion k is h_k (k=0,1,2). The cost of making a wrong decision for emotion k is υ_k. Note that the prior probability h_k and the cost probability υ_k are taken as being equal, and hence can be ignored. The problem is to find a neural network model for determining the class from which an unknown emotion signal is generated. If we know the probability density functions $f_k(\vec{X})$ for all classes, the Bayes optimal decision rule[13] can be used to classify $\vec{X}$ into class k if

$$h_k \vartheta_k f_k(\vec{X}) > h_m \vartheta_m f_m(\vec{X}) \quad (9)$$

where $k \neq m$.

A major problem with the Equation (9) is that the probability density functions of the classes are unknown. We can use Gram-Charlier series expansion to estimate these unknown probability density functions. In this case, the training set for the neural network consists of Gram-Charlier coefficients. Our objective is to use these coefficients for classification. If the neural network is trained with known Gram-Charlier coefficients, then to determine the emotion of a signal, we need only to feed its Gram-Charlier coefficients.

The Gram-Charlier series expansion of the probability density function of a random variable with a mean μ and variance σ^2 can be represented as

$$\rho(x) = \frac{1}{\sigma}\sum_{i=0}^{\infty} c_i \phi^{(i)}\left(\frac{x-\mu}{\sigma}\right), \quad (10)$$

where $\phi(x)$ is a Gaussian probability density function and $\phi^{(i)}(x)$ represents the *i*-th derivative of $\phi(x)$. For normalized data where $(\mu = 0, \sigma^2 = 1, c_0 = 1)$ the above equation can be simplified to

$$\rho(x) = (\phi(x) + c_3\phi^{(3)}(x) + c_4\phi^{(4)}(x) \quad (11)$$

where c_i coefficients are related to the central moments of $\phi(x)$. In a sense, derivatives of the Gaussian function in Equation (11) provide us with

the class type information for the emotion signal. Furthermore, $c_i\phi^{(i)}$ are orthogonal functions that present unique information about the emotion signal class distribution. Let's define $\beta(x) as$

$$\beta(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \qquad (12)$$

Then, the probability density function becomes

$$\rho(x) = \beta(x)\left\{1 + \frac{1}{6}\Lambda_3 A_3 + \frac{1}{24}(\Lambda_4 - 3)A_4 + \cdots\right\} \qquad (13)$$

where

$$A_i(x) = x^i - \frac{i^{[2]}}{2.1!}x^{i-2} + \frac{i^{[4]}}{2^2.2!}x^{i-4} - \frac{i^{[6]}}{2^3.3!}x^{i-6} + \cdots$$

$$i^{[k]} = \binom{i}{k} = \frac{i!}{(i-k)!}$$

$$m_k = \frac{1}{n}\sum_{i=1}^{n}[x(i)]^k \qquad (14)$$

$$\Lambda_3 = -\frac{m_3 - 3m_2m_1 + 2{m_1}^3}{3!},$$

$$\Lambda_4 = \frac{m_4 - 4m_3m_1 + 6m_2{m_1}^2 - 3{m_1}^4}{4!},$$

$$\Lambda_5 = -\frac{m_5 - 5m_4m_1 + 10m_3{m_1}^3 - 10m_2{m_1}^3 + 4{m_1}^5}{5!},$$

$$\Lambda_6 = \frac{m_6 - 6m_5m_1 + 15m_4{m_1}^2 - 20m_3{m_1}^3 + 15m_2{m_1}^4 - 5{m_1}^6}{6!}$$

Equation (14) is the so-called Gram-Charlier series [13] and the polynomial $A_i(x)$ is called the Tchebycheff-Hermite polynomial.

GERRN has a single processing stage (Figure 8). This single stage reads the wave file into an array of 2^{16}. Then, it estimates Gram-Charlier coefficients [13] corresponding to the vector.

These parameters are applied to a back propogation neural network to classify the signal (Figure 9). This neural network is fully connected feed forward neural network. Number of hidden neruons, *L,* is 20, and number of input neurons is 4. Gram-Charlier coefficients are presented as input to the neural network. The training matrix was prepared using these Gram-Charlier coefficients. Each column in the training matrix represented one set of these Gram-Charlier coefficients with a length of 4. Note that the amount of columns in the training set could be any number.

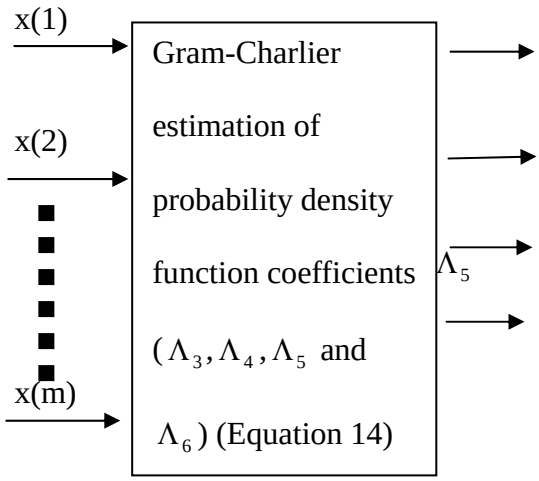

Figure 8. Each emotion signal ($x(i), i = 1 \cdots m$ where m is 65,536) is normalized (mean value is removed and standard deviation is made 1). Then, Gram-Charlier coefficients of the emotion signal are calculated.

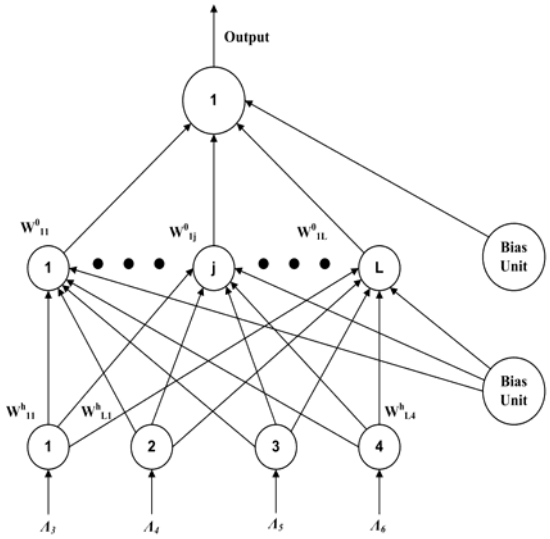

Figure 9. A three-layer model of Gram-Charlier emotion recognition neural network (GERNN) where L is 20.

4.2 Implementation

GERRN uses same routines as ERNN to read the files. However it generates the Gram-Charlier coefficients instead of generating the power spectrum of the wave file.

The coefficients are extracted as follows, using the tiny get_moment function, also displayed below. The get_coeffs function uses Equation 14, which is called Gram-Charlier series.

```
function c = get_moment(x,i)
  y = x.^i;
```

```
    c = sum(y) / size(x,1);

function c = get_coeffs(bw)
    x = bw;
    [xn,meanx,stdx] = prestd(x);
    m1 = get_moment(xn,1);
    m2 = get_moment(xn,2);
    m3 = get_moment(xn,3);
    m4 = get_moment(xn,4);
    m5 = get_moment(xn,5);
    m6 = get_moment(xn,6);

    d2 = 1;
    d3 = m3-3*m2*m1+2*m1^3;
    d4 = m4-4*m3*m1+6*m2*m1^2-3*m1^4;
    d5 = m5-5*m4*m1+10*m3*m1^2-
10*m2*m1^3+4*m1^5;
    d6 = m6-6*m5*m1+15*m4*m1^2-
20*m3*m1^3+15*m2*m1^4-5*m1^6;

    c3 = d3 / 6;
    c4 = (d4-6*d2+3) / 24;
    c5 = (d5-10*d3) / 120;
    c6 = (d6-15*d4+45*d2-15) / 720;
    c = [c3 c4 c5 c6];
```

The training and state graphs can be seen in Figure 10 and Figure 11.

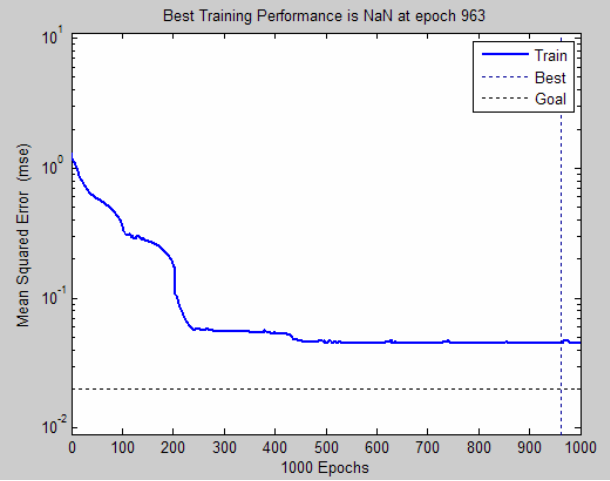

Figure 10. Training graph for GERNN at 1000 epochs using MATLAB.

5 Results and Discussion

The voice data, applied for both training and testing the ERNN, is obtained from the movies "Anger Management" and "Pick of Destiny". Figure 12 shows three types of emotion voice signals.

To reach the performance goal, a sliding window is used to segment the data in order to arrange as many sets of data as possible for training and testing the neural network. From voice signals, a set of 97920 training sequences, 128 samples

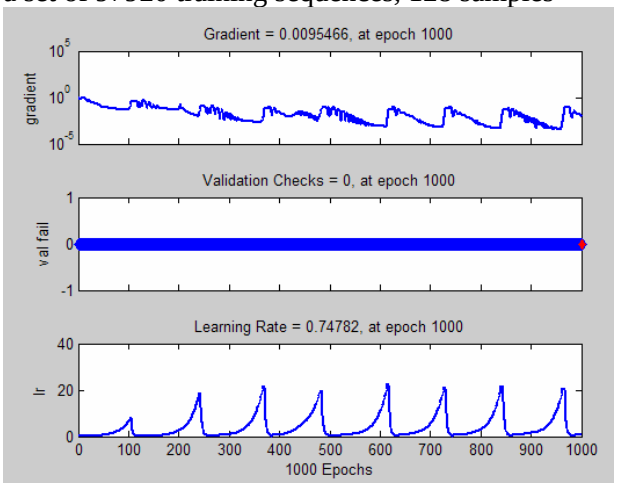

Figure 11. Training state graphs for GERNN at 1000 epochs using MATLAB.

each, was assembled to train the emotion recognition neural network. A new set of 24480 testing sequences was utilized to test the ERNN performance. Figure 6 shows a sample of 3 voice signals of angry, 3 voice signals of happy and 3 neutral voice signals. The corresponding power spectra of these voice signals are shown in Figure 13. Both the amplitude and their power spectra exhibit random pattern and a set of features that can be used for classification is not recognizable. Therefore, a trained neural network is conceivable to recognize the emotion buried into the voice signals.

		Emotions			
		Angry	**Happy**	**Neutral**	**Result**
Samples	**Angry**	**51.6848**	46.5997	1.7155	Angry
	Happy	13.0376	**70.9839**	15.9784	Happy
	Neutral	11.4569	22.0806	**66.4624**	Neutral

Table 1: Test results of ERNN from samples of three emotions, angry, happy and neutral.

As can be seen in Table 1, ERNN achieves an average recognition performance of 100%. This performance is impressive and statistically reliable since 97920 data segments are used in training the neural network.

Furthermore, the results obtained when testing the GERNN show that emotion recognition can be detected with 33% of accuracy. Out of 15 signals 12 were used for training and 3 were used for testing. The number of hidden neurons for the back

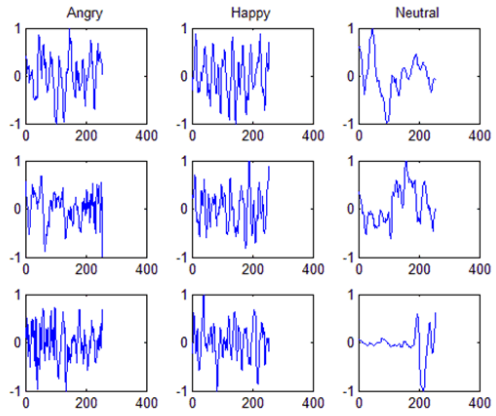

Figure 12. 256 time-samples X(nT) from each of angry, happy and neutral emotions respectively.

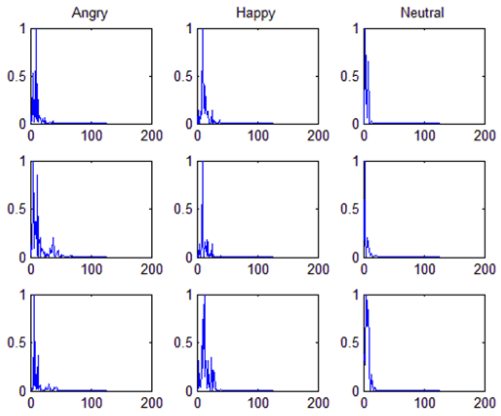

Figure 13. 128 power-samples $X_p(\Omega)$ from each of angry, happy and neutral power domain emotions respectively.

propogation neural network was chosen to be 20, based on experience. Compared to ERNN, GERNN performs poorly. This leads us that we cannot represent emotion signals with its Gram-Charlier coefficients.

This superb performance for ERNN can be attributed to adequate differences in the power spectrum of the voice signal that allows the neural network to adapt for recognition. On the contrary, the Gram-Charlier coefficients contain and display information related to the random phase spectrum. This random phase interferes with and obscures the inherent frequency characteristics of voice signals needed for properly training GERNN.

6 Conclusion

In this study we have developed two neural networks that are designed to classify the voice signals. Voice signals are random signals and are not readily quantifiable and lack uniquely recognizable features. Therefore, the neural network becomes appealing for classifying these signals because they are trainable.

The optimal values for neural networks weights are estimated using the back propagation algorithm. Experimental measurements of voice signals are utilized to train and test the emotion recognition neural network. The ERNN shows a remarkable 100% classification performance where GERNN does a performance of 33%.

The results for the ERNN are encouraging and suggest that neural networks are potentially useful for emotion recognition. Furthermore, the ERNN renders practical advantages such as real-time processing, adaptability and training capability. It is important to point out that similar neural network designs can be used in medical ultrasonic imaging for tissue characterization and diagnosis.

Another advantage is that the ERNN checks the signal in predefined windows continuously. The speech length is also arbitrary, thus continuous testing and getting the average of these tests yields better results.

As mentioned earlier, the language and the content of the speech is not tested. However, what the user is saying and how fast it is spoken may contribute to results, if they are taken into account. In future work, the first step may be taking predefined phrases for testing. The second step may be using speech recognition along with emotion recognition.

Another objective of this study is that we want this neural network to work with near real time applications, like computer games. Role-playing games can utilize the player's acting capabilities which might improve gaming experience.

References:

[1] Razak A., et al., “Comparison Between Fuzzy and NN Method for Speech Emotion Recognition”, Proceedings of the Third International Conference on Information Technology and Applications, 2005.

[2] Mingyu You, et al., “Emotion Recognition from Noisy Speech”, 2006 IEEE International Conference on Multimedia and Expo, Toronto, 2006.

[3] Dellaert F., Polzin T., Waibel A., “Recognizing Emotion In Speech”, Spoken Language, 1996, ICSLP 96, Proceedings., Fourth International Conference on, 1996.

[4] Lee, F. Li, L. Huang, R. Recognizing Low/High Anger in Speech for Call Centers, Conference on Signal Processing, Robotics and Automation, 2008. Proceedings of the 7th WSEAS International.

[5] Sharma, R. Neumann U., Kim C. Emotion Recognition In Spontaneous Emotional Utterances From Movie Sequences, WSEAS International Conference on Electronics,

Control & Signal Processing, 2002.
[6] Mao, X. Zhang, B. Luo, Y. Speech Emotion , Recognition Based on a Hybrid of HMM/ANN, Proceedings of the 7th WSEAS International Conference on Applied Informatics and Communications, 2007.
[7] J. A. Freeman and D. M. Skapura, Neural Networks: Algorithms, Applications and Programming Techniques, Addison Wesley Publishing Company, 1991.
[8] T. Masters, Practical Neural Network Recipes in C++, Academic Press Publishing Company, 1993.
[9] A. Cichocki and R. Unbehauen, Neural Networks for Optimazing and Signal Processing, John Wiley & Sons Publishing Company, 1993.
[10] Werbos, P. The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting, John Wiley & Sons, New York, 1994.
[11] Hollis P. W. and Pauolos J. J., "Artificial Neural Networks Using MOS Analog Multipliers," Int. Conf. Neural Networks, 1988.
[12] Hollis P. W. and Paulos J.J., "A Neural Network Learning Algorithm Tailored for VLSI Implementation," IEEE Transactions On Neural Networks, Vol. 5, No. 5, 1994.
[13] Kendall M. G. and Stuart A., 1963. The Advanced Theory of Statistics, Charles Griffin & Company Limited.