XV Reunión de Trabajo en Procesamiento de la Información y Control, 16 al 20 de septiembre de 2013

Speech emotion recognition using a deep autoencoder

Neri E. Cibau[1,3,*], Enrique M. Albornoz[1,2], Hugo L. Rufiner[1,2,3]

[1] *Centro de I+D en Señales, Sistemas e Inteligencia Computacional (SINC(i)), UNL.*
[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).*
[3] *Laboratorio de Cibernética, Fac. de Ingeniería, Universidad Nacional de Entre Ríos.*
* *nericibau@gmail.com*

Abstract— **The main objective of the emotion recognition systems is to improve the human-machine interaction, giving them a more natural behavior to attend different situations and user requirement. Several works on this domain use the prosodic features and the spectrum characteristics of speech signal with classifiers based on neural networks, Gaussian mixtures and other standards classifiers. In this paper a deep autoencoder based on a Multilayer perceptron was used as a classifier. Following the deep learning paradigm, an autoencoder training strategy layer by layer was implemented, which results in a stack of perceptrons with a set of "good" weights. A final fine-tune training was applied to the whole classifier. Many configurations were evaluated in order to predict six different emotions and neutral emotional state. Performance of the classifier was over 70%, promoting better results for this novel approach.**

Key words— **Emotion Recognition, Deep Auto-encoder, Multilayer Perceptron.**

1. INTRODUCTION

Speech is the most important and efficient channel of communication in humans [1]. In addition to what is said, a lot of extra information is exchanged which is unconsciously manifested and interpreted, e.g. facial expressions, body language and prosodic features. The emotional state of a speaker can be noticed in this information and this motivates its study.

There are many theories that explain the emotions from different perspectives. For example, Ekman et al. [2] theorized the existence of six universal emotions: happiness, sadness, surprise, fear, disgust and anger. Following this idea, in this work we use a set of seven discrete emotions. Emotion recognition plays an essential role in human-machine interaction systems by providing a more natural behavior [3]. That implies not only to carry out instructions given directly by users, these systems should act according to the implicit information produced by them in the emotions. This has been implemented to support semi-automatic diagnosis of psychiatric diseases [4], detection of emotional attitudes from child through visual-acoustic interaction software [5], systems for real-life emotion detection using a corpus of agent-client spoken dialogues from a medical emergency call centre [6].

The application of neural networks to the complex problem of emotion classification is not so novel. On the other hand, there are few works with focus on deep architectures [7]. As Bengio states: "there is theoretical evidence which suggests that in order to learn complicated functions that can represent high-level abstractions (e.g. in vision, language, and other AI-level tasks), one needs deep architectures" [8].

Here we propose a method which takes the idea introduced by Hinton et al. to train deep belief networks in order to obtain better generalization results [8] [9]. Differently from the greedy layer-wise training algorithm proposed by [9] in this work we used Deep Autoencoders (DA) based on Multilayer perceptrons trained through backpropagation algorithm.

In the next section the emotional speech database used in the experiments is presented. Section 3 describes acoustic features extraction stage, and classifiers based on Multilayer perceptron and Deep Autoencoders. The method to build the Deep Classifier here proposed is also explained. In Section 4 experiments are described. Section 5 presents the obtained results. Also, performance and selection of the method are discussed. Finally, in Section 6, conclusions and future works are presented.

2. THE SPEECH CORPUS

The emotional speech signals used in this work correspond to the EMO-DB database, developed at the Communication Science Institute of Berlin Technical University [10], which consists of 535 utterances recorded in German. In order to construct the database, ten actors (5 female and 5 male) simulated the emotions, producing 10 German utterances divided into 5 short and 5 longer sentences. This corpus covers anger (A), disgust (D), boredom (B), joy (J), fear (F), sadness (S) and the neutral (N) emotional state. The corpus distribution is shown in Table 1. After recording all utterances, a perception test was carried out by 20 subjects in order to evaluate quality and naturalness of them. Only utterances with a recognition rate better than 80% and naturalness better than 60% were taken.

3. METHODS

2.1. Acoustic Features Extraction

For every emotional utterance two sets of characteristics were extracted: MFCC (Mel Frequency Cepstral

Coefficients) and prosodic features.

The MFCC parameterizations were calculated using a Hamming windows of 25 ms with a 10 ms frame shift and the first 12 MFCC were used here (extracted using the Hidden Markov Toolkit [11]). The mean of the log-spectrum (MLS) on each frequency band along the frames was calculated for every utterance, as was introduced in [12].

Table 1: Number of utterances corresponding to each emotion class.

Emotion class	A	B	D	F	J	S	N
No. of Utterances	127	81	46	69	71	62	79

Many works used the prosodic features in emotion recognition, while the classic methods were used to calculate energy (E) and voice fundamental frequency (F_0) along signals [13]. Many useful parameters can be extracted from prosodic features; usually the minimum, mean, maximum and standard deviation over the whole utterances is selected. This set of parameters has already been studied and some works reported an important information gain to discriminate emotions [14].

In this work, the selected feature vector is which obtained best results in our previous experiments reported in [12]. It has 46 elements: 12 mean MFCC + 30 MLS + F_0 mean + E mean + the standard deviation for F_0 and E.

2.2. Multilayer Perceptron

Multilayer Perceptron (MLP) is a class of artificial neural network that consists of a set of simple perceptrons arranged in layers. In the MLP, the perceptrons are fully connected between layers without connections between them in the same layer. The input vector feeds into each of the first layer perceptrons, the outputs of this layer feed into each of the second layer perceptrons, and so on [15]. The output of the neuron is the weighted sum of the inputs plus a bias term, passed through an activation function. Backpropagation training algorithm was generally used in order to find the optimal values for all the parameters of the MLP. In this work the Stuttgart Neural Network Simulator[1] (SNNS) was used for this purpose. SNNS offers the possibility to interact with users through a graphical interface; however, **batchman**[2] program was used here because many hours of CPU time and relatively complex networks manipulations were required. Batchman uses a specific language that allows to generate, train, test and perform other tasks with neural networks in background execution.

2.3. Deep Autoencoders

We can describe a general classification task as a process in which, given a certain classes, we have to pick from a set of features those which better fit each of these classes. When that set consists of too many features the task might not be so easy and it could be better to have a smaller set, but whose elements are still representative of the first. Applying this concept to neural networks, we want to find a low dimensional representation of a high dimensional data and that is what autoencoders can do [16]. This process of dimensionality reduction is performed by such autoencoder which consists of a multilayer neural network with a small central layer and whose aim is to reconstruct its high-dimensional input vectors. Since the input data is forced through that "bottleneck" and then is should be reconstructed, a low-dimensional code is learned by the network (for example see the autoencoder in Fig. 2). To train the whole autoencoder we could initialize weights randomly and then apply the backpropagation algorithm. However, following this way in the case of nonlinear deep autoencoders (i.e. autoencoders with 2, 3 or 4 hidden layers) we only reach poor local minima since gradients are very small in the early layers. Due to local minimum problems, backpropagation need to be close to a good solution before training the weights of the whole deep autoencoder [16] [17]. Finding such initial weights requires a very different type of algorithm that learns one layer of features at a time. This way of initializing weights, so called "pre-training", is accomplished by applying a serie of steps as explained below. The addition of a last layer converts the deep autoencoder (DA) into a deep classifier (DC).

2.4. Building the Deep Classifier

The whole process to obtain a deep classifier can be split into 8 steps. To better understand the previous ones, we will first take a look to the final classifier that will be obtained at the final step. Figure 1 shows the topology of that classifier which is a feed forward, fully connected network with an input layer **i**, two hidden layers $\mathbf{h_1}$, $\mathbf{h_2}$ and an output layer **o**. Similarly, the corresponding number of units is represented by n_i, n_{h1}, n_{h2} and n_o for each layer, respectively. All units from this and other networks used in this work have the same characteristics: the activation corresponds to the logistic function, and outputs to the identity function. Since the input vector has 46 elements and there are 7 classes to discriminate, the number of units in the input and output layer are fixed, i.e. n_i=46, n_o=7.

Regards the hidden layers, the number of units in each $-n_{h1}$, $n_{h2}-$ are degrees of freedom to modify during experiments in order to find the best net architecture. A restriction imposed is that the number of units in a layer should be less than in the previous one, hence the size of rectangles in the diagrams. This restriction was made in order to force the net to progressively perform a dimensionality reduction of the input features. There are several previous steps that were made to pre-train the classifier described above, they are detailed below.

[1] The latest version 4.3, available in http://www.ra.cs.uni-tuebingen.de/SNNS/

[2] A command interpreter for batch jobs, included in SNNS.

Step 1. The aim of this step is to initialize the weights corresponding to links between the input layer **i** and the first hidden layer $\mathbf{h}_1$ of the classifier. To achieve that, an autoencoder was constructed which was formed by a single hidden layer $\mathbf{h}_1$ with n_{h1} units and identical input and output layers with 46 units each, fully connected with $\mathbf{h}_1$ (Fig. 2). Different values of n_{h1} were experimen-

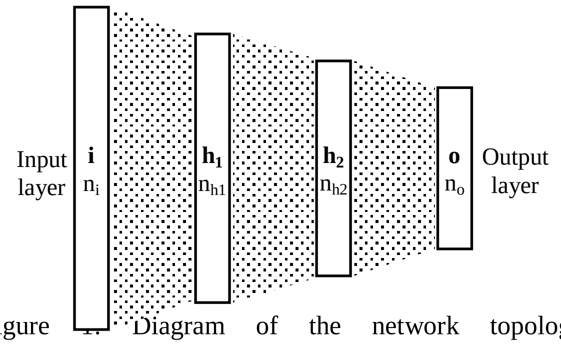

Figure 1: Diagram of the network topology corresponding to the final classifier. Layers are represented by rectangles, and dot pattern indicates the fully connection between them. The network has an input layer **i**, two hidden layers $\mathbf{h}_1$, $\mathbf{h}_2$ and an output layer **o**. The number of units in each layer is indicated by n_i, n_{h1}, n_{h2} and no, respectively.

ted. During training, the Sum of Squared Errors (SSE) in each epoch was calculated for training and test sets respectively. For each configuration, the network corresponding to the epoch that yields the minimum test SSE was chosen. Then, SSE for validation set was additionally computed.

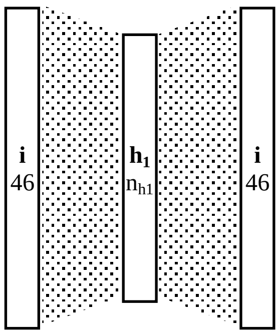

Figure 2: Diagram of the autoencoder corresponding to **step 1**. It was formed by a single hidden layer $\mathbf{h}_1$ with n_{h1} units and identical input and output layers with 46 units (coder-decoder).

Step 2. It was taken the best autoencoder configuration -minimum test SSE- from the previous step and the output layer was removed (decoder), obtaining a network (coder) like the one shown in Fig. 3. We called it as $\mathbf{i+h}_1$ because these indexes corresponds to the layers that form the network. All training, test and validation patterns were passed through this network in order to generate outputs that will be the input patterns used in the following step.

Step 3. Similarly to **step 1**, a new autoencoder was constructed (Fig. 4), but in this case the objective was to initialize the weights corresponding to links between the first and second hidden layer, i.e. $\mathbf{h}_1$ and $\mathbf{h}_2$. To train this autoencoder we used patterns obtained in **step 2** and the best autoencoder configuration was selected according the same rules applied in **step 1**.

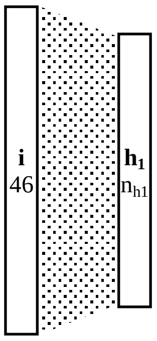

Figure 3: The resulting $\mathbf{i+h}_1$ net of **step 2** (coder), after removing the output layer (decoder) of the autoencoder from **step 1**.

Step 4. From the best autoencoder obtained in **step 3**, the output layer was removed resulting in a single layer network, $\mathbf{h}_1\mathbf{+h}_2$, as shown in Fig. 5.

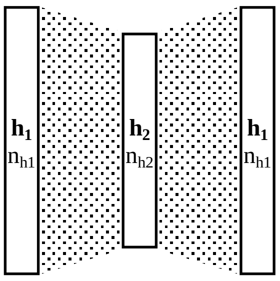

Figure 4: Diagram of the autoencoder corresponding to **step 3**. It was formed by a single hidden layer $\mathbf{h}_2$ with n_{h2} units and identical input and output layers with n_{h1} units.

Step 5. Resulting networks from **steps 2** and **4** were combined to build a DA that was again constructed as described below. Firstly, $\mathbf{i+h}_1$ and $\mathbf{h}_1\mathbf{+h}_2$ networks were

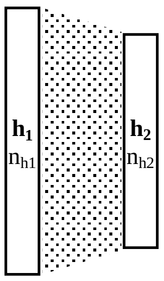

Figure 5: Resulting $\mathbf{h}_1\mathbf{+h}_2$ net of **step 4**, after removing the output layer of the autoencoder from **step 3**.

linked through one-to-one connections between units of **h1** layers of both into a new $\mathbf{i+h}_1\mathbf{+h}_2$ net. Note that we write $\mathbf{i+h}_1\mathbf{+h}_2$ instead of $\mathbf{i+h}_1\mathbf{+h}_1\mathbf{+h}_2$ since the one-to-one connections had fixed weights equal to one and they were not susceptible to being modified during training (Henceforth, we assume this kind of connections when linking nets). Secondly, a copy of that network was inverted resulting in another net **h_2'+h_1'+i'**, leaving weights unmodified (we use apostrophes to differentiate layers with identical characteristics but that are in different locations in the network). Finally, the DA **i+h_1+h_2+h_1'+i'** (see Fig. 6) was created by linking **i+h_1+h_2** and **h_2'+h_1'+i'** between units of **h_2** layers (note

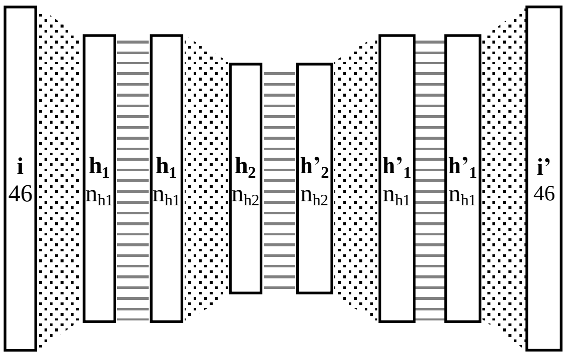

Figure 6: The **i+h_1+h_2+h_1'+i'** deep autoencoder corresponding to **step 5**. One-to-one connections between layers are represented by parallel lines.

that the **h_2-h_2'** junction is referenced only as **h_2** due to it act as a single layer). After training, the best DA was selected.

Step 6. The best DA obtained previously was converted into **i+h_1+h_2** net by eliminating the decoder part, i.e. **h_1'** and **i'** layers (Fig. 7). Next, the same train, test and validation pattern sets using during training in **step 5** were passed through **i+h_1+h_2** to obtain a new sets of patterns. They will be the input patterns for the following step.

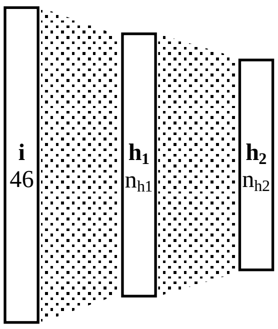

Figure 7: The **i+h_1+h_2** network after **step 5** was applied.

Step 7. In order to initialize the output layer weights of the classifier, a fully connected network was created (Fig. 8) with an input layer **h_2''** of n_{h2} units (the same number of units as the **i+h_1+h_2** output layer) and an output layer of 7 units. It will be, in turn, the output layer of the classifier. Training patterns were composed of inputs from the previous step and the desired outputs according to the labeled data provided in the speech corpus. For each partition, the preferred network was the one that yield the minimum test error percentage, in other words, the highest percentage of correct classifications for test patterns. The classification rule applied in the output layer is "winner takes all", that means the class given by the network will correspond to the unit with the highest output. And, of course, it was classified correctly if that agrees with the teaching output.

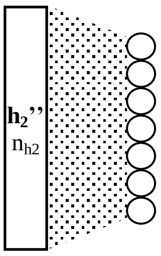

Figure 8: Network of step 7 whose aim is to initialize the final layer weights of the classifier.

Step 8. The classifier (Fig. 9) is constructed by joining networks from **step 6** and **7**. The pre-training procedure is complete now. After a final "fine-tuning" training, the best classifier was selected by the same criteria of **step 7**.

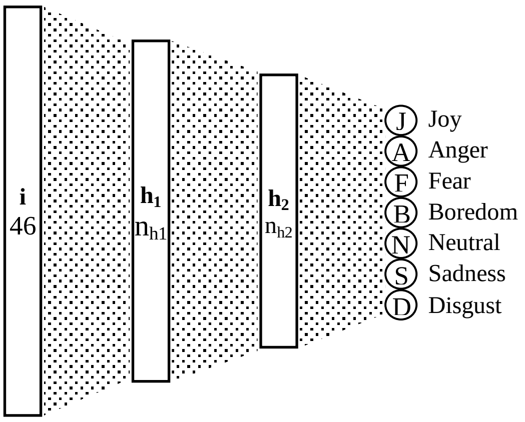

Figure 9: Resulting classifier at the end of the whole process. In the output layer, emotions classes are schematically labeled.

4. CLASSIFICATION EXPERIMENTS

In order to estimate the performance of the classifier, ten partitions were generated which were extracted from the whole corpus at random. As can be seen in Table 1, the distribution of emotions is quite unbalanced. This can bias the results obtained with several classifiers. To avoid this problem, the dataset was balanced by equalizing the size of the classes which was performed by selecting randomly the same number of samples for all classes in each partition (46 x 7 = 322 utterances), so it consisted in a train set (196 patterns), test set (63 patterns) and validation set (63 patterns) [12].

Experiments were performed using six different network structures and two combinations of backpropagation algorithm parameters, leading to twelve configurations. The network structures refer to diverse configurations with different number of units in

N. Cibau, E. M. Albornoz & H. L. Rufiner; "Speech emotion recognition using a deep autoencoder"

hidden layers, i.e. n_{h1} and n_{h2}. Some parameters of the backpropagation algorithm were evaluated: Learning Rate (LR), Momentum (M), Flat Spot Elimination value[3] (FSE) and Tolerance[4] (T). The two combinations of them were called Parameters A (PA) and Parameters B (PB). They were different in training process corresponding to **steps 1, 3** and **5** (Table 2) but the same in **steps 7** and **8** (Table 3)**.**

Table 2: Training parameters of **steps 1**,**3** and **5**

	Step 1				Step 3				Step 5			
	LR	M	FSE	T	LR	M	FSE	T	LR	M	FSE	T
PA	0.08	0.1	0.1	0.1	0.08	0.1	0.1	0.1	0.07	0.08	0.1	0.1
PB	0.08	0.08	0.07	0.07	0.08	0.1	0.07	0.07	0.05	0.05	0.07	0.07

Table 3: Training parameters of **steps 7** and **8**

Step 7				Step 8			
LR	M	FSE	T	LR	M	FSE	T
0.3	0.3	0.15	0.1	0.01	0.05	0.1	0.1

PA and PB configurations were selected from extensive previous experimentations (not reported here) with initial typical values, graphics of SSE were analyzed, and then parameters were set in order to get smoother graphics. This allowed networks to easily reach a deeper local minimum and to obtain a better performance.

5. RESULTS AND DISCUSSION

The results obtained for the deep classifiers for all twelve configurations are shown in Table 4. The classification rates are the average percentage over the ten partitions corresponding to validation (training classification rates are omitted).

Table 4: Performance of deep classifiers. Classification rate in [%].

Network structure	PA	PB
46+27+17+7	64.60	68.73
46+30+20+7	66.83	69.21
46+33+23+7	65.87	68.73
46+36+26+7	64.13	68.89
46+39+23+7	63.33	70.95
46+42+23+7	61.11	69.36

The best performance was 70.95% corresponding to the 46+39+23+7 classifier using PB. Nevertheless, the same network structure yield only 63.33% with PA, the second lowest value. Comparatively, a MLP with one hidden layer of 90 units gave a 66.83% performance using identical partitions [12]. With respect to network configurations, there is no significant difference among the structures that use PB (0.8 standard deviation). In contrast, results using PA were scattered between 61.11% and 66.83% (2.0 standard deviation).

It is clear that using PA yield worse results than PB for all networks structures. Although these parameters only modified **steps 1, 3** and **5**, their influence on the classifier performance is significant. To present that more clearly, SSEs for validation in these steps are shown in Table 5 and Table 6 using PA and PB, respectively. Also, the classification rate of **step 7** is included.

Table 5: Results of steps previous to the final classifier training with PA

Network Structure	PA			
	Step 1 (SSE)	Step 3 (SSE)	Step 5 (SSE)	Step 7 (%)
46 27 17	8.18	1.54	9.66	42.54
46 30 20	11.35	1.83	6.04	46.51
46 33 23	7.08	1.59	6.05	43.81
46 36 26	8.16	1.87	7.51	46.35
46 39 23	9.25	1.70	8.04	42.54
46 42 23	9.90	2.74	11.87	47.62

All average values of SSE for **steps 1, 3** and **5** using PA (8.99, 1.88, and 8.20) are higher than PB case (6.91, 1.37, and 3.34) but the biggest difference happens for **step 5** (40% smaller). In this step, weights of the DA are adjusted to obtain a good representation of the input in its output, thus, a better dimensionality reduction in its central layer is obtained.

Classification rate of **step 7** do not show significant differences between PA and PB.

Table 6: Results of steps previous to the final classifier training with PB

Network Structure	PB			
	Step 1 (SSE)	Step 3 (SSE)	Step 5 (SSE)	Step 7 (%)
46 27 17	4.08	0.54	3.65	48.41
46 30 20	7.28	1.45	3.23	45.87
46 33 23	5.15	0.96	3.39	44.76
46 36 26	7.13	1.54	3.26	47.94
46 39 23	7.47	1.45	3.26	48.10
46 42 23	10.35	2.25	3.23	47.78

6. CONCLUSIONS AND FUTURE WORK

In this work we evaluated the application of a Deep Autoencoder to the task of emotional speech recognition. We proved the importance of doing a pre-training as a way to initialize weights. The results represent a good starting point for this novel method, they are promising and encourage us to accomplish

[3] A constant value which is added to the derivative of the activation function to enable the network to pass flat spots of the error surface [19].

[4] A tolerance for the maximum difference between a teaching value and an output of an output unit, i.e. which is propagated back as zero [19].

more experiments using different structure networks and parameters in order to improve the classifier performance. Other aspect to consider is the possibility of including some sparsity constraints when training the DA. Moreover, the proposed method could be applied in other tasks such as digit and speech recognition [18].

ACKNOWLEDGMENTS

The authors wish to thank: the *Agencia Nacional de Promoción Científica y Tecnológica*, the *Universidad Nacional de Litoral* (with PAE 37122, PACT 2011 \#58, CAI+D 2011 \#58-511), the *Universidad Nacional de Entre Ríos* and the *Consejo Nacional de Investigaciones Científicas y Técnicas* (CONICET), for their support.

REFERENCES

[1] L. Rabiner, B-H Juang, Fundamentals of Speech Recognition, Prentice Hall PTR, 1993.

[2] P. Ekman, E. R. Sorenson, W. V. Friesen, "Pan-cultural elements in facial displays of emotions," *Science*, vol. 164, no. 3875, pp. 86-88, 1969.

[3] B. Schuller, S. Steidl, A. Batliner, "The INTERSPEECH 2009 Emotion Challenge," *INTERSPEECH 2009, 10th Annual Conference of the International Speech Communication Association*, pp. 312-315, 2009.

[4] D. Tacconi, O. Mayora, P. Lukowicz, B. Arnrich, C. Setz, G. Troster, C. Haring, "Activity and emotion recognition to support early diagnosis of psychiatric diseases," *Proceedings of the 2nd International Conference on Pervasive Computing Technologies for Healthcare 2008, Tampere, Finland,* pp. 100-102, 2008.

[5] Serdar Yildirim, Shrikanth Narayananb, Alexandros Potamianosc, "Detecting emotional state of a child in a conversational computer game," *Computer Speech & Language*, vol. 25. Issue 1, pp. 29–44, 2011.

[6] Laurence Devillers, Laurence Vidrascu, "Real-Life Emotion Recognition in Speech," *Speaker Classification II: Selected Projects. Lecture Notes in Computer Science*, vol. 4441, pp. 34-42, 2007.

[7] A. Stuhlsatz, C. Meyer, F. Eyben, T. Zielke, G. Meier, B. Schuller, "Deep Neural Networks for Acoustic Emotion Recognition: Raising the Benchmarks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.

[8] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[9] G. E. Hinton, S. Osindero, Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[10] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, B. Weiss, "A Database of German Emotional Speech," *9th European Conference on Speech Communication and Technology - Interspeech 2005,* pp. 1517–1520, 2005.

[11] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, P.Woodland, The HTK Book (for HTK Version 3.1), Cambridge University Engineering Department, England, (Dec. 2001).

[12] E. M. Albornoz, D. H. Milone, H. L. Rufiner, "Spoken Emotion Recognition using Hierarchical Classifiers," *Computer Speech and Language,* vol. 25, pp. 556-570, 2011.

[13] J. R. Deller, Jr., J. G. Proakis, J. H. Hansen, Discrete-Time Processing of Speech Signals, Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993.

[14] J. Adell Mercado, A. Bonafonte Cávez, D. Escudero Mancebo,, "Analysis of prosodic features: towards modelling of emotional and pragmatic attributes of speech," *SEPLN 2005,* no. 35, pp. 277–284, 2005.

[15] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice Hall, 1998.

[16] G. E. Hinton, R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science,* no. 28, pp. 504-507, 2006.

[17] R. Brueckner, B. Schuller, "Likability Classification – A not so Deep Neural Network Approach," in *Proceedings INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association*, Portland, Oregon, USA, pp. 1-4, 2012.

[18] G. E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 82-97, 2012.

[19] Andreas Zell et. al., "Stuttgart Neural Network Simulator. User Manual, version 4.2," *Institute for Paralell and Distributed High Performance Systems (University of Stuttgart), Wilhelm-Schickard-Institute for Computer Science (University of Tubingen),* 2000.

[20] A. Mohamed, T. Sainath, G. E. Dahl, B. Ramabhadran, G. E. Hinton, M. Picheny, "Deep Belief Networks using Discriminative Features for Phone Recognition," in *ICASSP-2011, ISCA*, Portland, OR, USA, pp. 5060-5063, 2012.

[21] K. R. Scherer, A blueprint for affective computing: a sourcebook, Oxford: Oxford University Press, 2010.