Wavenet Based Speech Recognition System On An Embedded Board

Aniruddha Sharma, John Sahaya Rani Alex

Abstract:--- Speech Recognition is a vital and indispensable part of many modern-day systems and has subsequently become ubiquitous, finding diverse applications in fields such as automation, voice control, security, robotics etc. This paper aims to demonstrate implementation of an isolated spoken word recognition based on WAVENETs on an embedded board using an open-source numerical computing software called GNU Octave. WaveNet is an Artificial Neural Network (ANN) with wavelet function as an activation function. In this work, Gaussian wavelet is chosen as an activation function. The speech recognition involves the use of Mel-Frequency Cepstral Coefficients (MFCC) features which are calculated from the speech signal and fed as input to the NN. The Multi-Layer Perceptron (MLP) Feed Forward Neural Network is configured to process speech signal and is trained using back-propagation algorithm in MATLAB. The trained weights are then fed into and implemented using GNU Octave on Raspberry Pi. Texas Instruments' TIDIGITS Corpus is used for training and testing the neural network.

Index Terms: Mel Frequency, Neural Network, Raspberry Pi, Wavenet

I. INTRODUCTION

Speech recognition's dynamic nature has made it an alluring territory for researchers and enthusiast's alike, garnering growing interest and attention mainly due to its incredible versatility in terms of potential applications in fields such as automation, security, robotics etc., as well as the recent advances in deep learning and big data. Currently most applications in the domain of speech recognition follow a classification-based approach. Some widely used and favored models are Hidden Markov Models (HMM) and Artificial Neural Networks (ANN). Hidden Markov Models' popularity can be attributed to their treatment of observations and outcomes of a given model as a function of probability of occurence, allowing for increased flexibility and a robust modelling framework for time-varying spectral vector sequences such as speech signals [1].

However, there are some fundamental drawbacks of the Hidden Markov Models' statistical approach based on approximation from a sequence of observations. Primarily, HMMs require huge amounts of data and numerous parameters to be trained and set-up and require independence of neighbor frames. Furthermore, HMM algorithms are more expensive, in terms of both memory and compute time, both of which are critical for embedded applications. ANNs are regarded to be self-adaptive, highly data-driven, and versatile computational tools that have been slowly gaining traction and popularity over HMMs owing to their ability of capturing fundamental complex and non-linear attributes of any sustained (time-series) phenomena with a considerably high accuracy. ANNs have three basic layers: an input layer, an output layer and an intermediate hidden layer. The hidden layers consists of 'neurons', programmed with a pre-defined activation function (depending on the application), which defines the output of a particular node/neuron. The neural network is trained using specific algorithms which modify the biases and weight vectors during the process. After some iterations, the network will yield the output, which is ideally the output). These classification techniques, if implemented in reliable, compact and low-power embedded boards, can easily allow us to perform speech recognition anywhere, given the additional portability and efficiency both in terms of power and performance. When using a low perplexity corpus enumerated with a vocabulary of 571 words, the Neural Network Model's exhibited approximately 86% sentence precision and 94% word-exactness. When compared with the same case implemented using HMMs ,which yielded 88% sentence precision and 96% word exactness , the results were found to be marginally substandard. When testing without a definite language structure , the NMM yielded a 58% word exactness while the HMM delivered 49% exactness. Another set of trials made use of the TIMIT database (an aggregation of 2072 sentences spoken by 415 individuals). The observed results were poor for both cases and iterations. The results yielded by NNM were found to be better than those observed while using HMMs[2]. Identifying and isolating audio/sound waves in environments with high ambient noise is a grueling and laborious job. The results are found to be substantially better when employing Deep Neural Network (DNN) classifiers and Support Vector Machine (SVM) for the same [3]. A comparative study aimed at classifying the phonemes based on their features indicates that ANNs are better at isolated or short word recognition [4]. Using Self-Organizing Feature Maps (SOFM) to reduce dimensions of feature vectors extracted using the MFCC, techniques like Discrete Wavelet Transform (DWT) and Perceptual Linear Predictive (PLP) indicate that recognition accuracy with SOFMs is at par with conventional methods, even though the feature vector dimensions were scaled down and reduced significantly [5]. An isolated spoken word (ISW) recognition system based on WaveNets or Artificial Neural Networks (ANN) with wavelet activation function , is proposed in this research. Section 2 elaborates on the proposed work and the methodology adopted. Section 3 presents the implementation of the neural network on Raspberry Pi with Section 4 concluding the paper.

Revised Manuscript Received on March 10, 2019.

Aniruddha Sharma, School of Electronics Engineering, Vellore Institute of Technology Chennai, Chennai, India.

John Sahaya Rani Alex, School of Electronics Engineering, Vellore Institute of Technology Chennai, Chennai, India.

II. PROPOSED WORK

A. Proposed Methodology

A NN cannot be given raw unprocessed speech data. MFCC feature extraction is performed on the speech signal to extract features i.e to identify or separate linguistic content while discarding background noise and other components which are not of importance. The MFCC files/features are then fed as an input to the NN, which is trained using backpropagation method which is shown in Fig. 1.

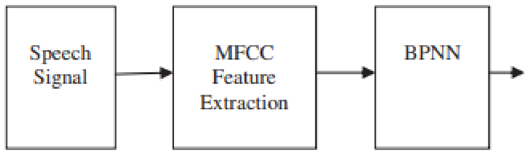


Fig.1 BPNN Neural Network Training

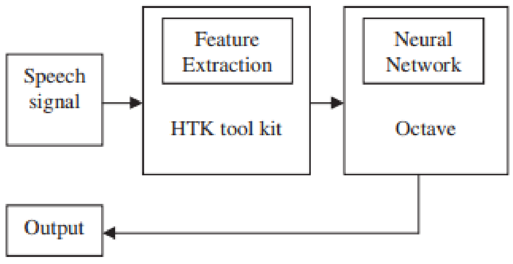


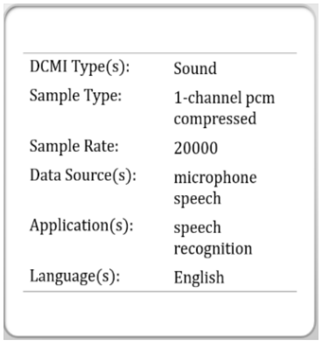
Fig.2 Testing Neural Network on Raspberry Pi using Octave

The general flow of testing of NN as shown in Fig. 2, is done by feeding a testing sample to the Raspberry Pi for feature extraction. The MFCC features are extracted using Hidden Markov Tool kit (HTK)[6]. The trained model is loaded in to R-PI. The test data which is not used for training is used for testing.

B. TIDIGITS Corpus

The TIDIGITS Corpus contains speech, originally designed and collected at Texas Instruments Inc. (TI), with specifications shown in Fig. 3, specially for designing and analyzing algorithms for the purpose of recognizing speaker-independent connected digit sequences.

The corpus consists of 326 individuals (114 women, 111 men, 51 girls and 52 boys) with each of them articulating sequences of 77 digits each. The individual sequence are divided into test and train subsets to be given as an input to the neural network.



DCMI Type(s):	Sound
Sample Type:	1-channel pcm compressed
Sample Rate:	20000
Data Source(s):	microphone speech
Application(s):	speech recognition
Language(s):	English

Fig.3 TIDIGITS Corpus Specifications

C. Feature Extraction Using MFCC Method

The Mel-Frequency threshold is spaced linearly at low frequencies (below 1000Hz) logarithmically at higher frequencies (above 1000Hz) to allow for better spectral analysis of the speech signal. To avoid severe degradation of events at boundary, ensure better classification and reduce the possibility of transients affecting the signal quality, the framing duration and windowing period is set to 25ms with a 10ms overlapping period.

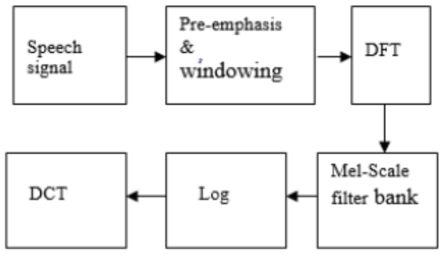


Fig.4 MFCC Feature Extraction Method

The frequency components of the framed signal are passed through 24 triangular Mel-Scale Filter Banks. Discrete Cosine Transform (DCT) technique is used to decorrelate the cepstral coefficients after which the initial 13 outputs obtained from the DCT data block are taken as static MFCC and derivatives are calculated to be used for speaker recognition, as shown in Fig.4.This is done to exploit the compaction property of DCT, where most of signal energy is concentrated in the first 10-15 blocks.

D. Backpropagation Neural Network (BPNN)

Backpropagation can be regarded as a supervised learning algorithm to train multilayer ANNs exhaustively using gradient descent. Backpropagation networks mainly consist of three layers: an input layer, intermediate hidden layer(s) and an output layer. Typically, neurons are configured in a feedforward orientation , with successive connections from input, hidden and output layer . Learning in backpropagation neural network is a two stage process.

The first step involves feeding a sample training set as an input to the input layer, after which the network begins multiple processing iterations from one layer to another. Subsequently, at the output layer we observe that a pattern is generated. In the event that the output does not agree with the target pattern, the error-term is recalculated by transmitting it back to the previous layer and eventually to the input layer. This constitutes the general error calculation and backpropagation technique using which, weights are updated.

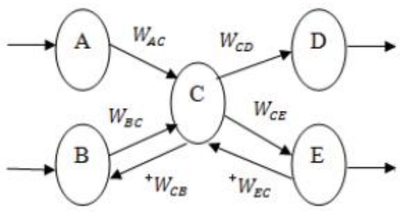


Fig.5 Backpropagation Algorithm Flow

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Fig 5. shows a backpropagation network with two inputs A, B, two output D, E and one intermediate hidden neuron C. The backpropagation algorithm involves the original weight vector W_CE and the updated weight vector +W_EC, which is "backpropagated" to the hidden layer with error term included. In a similar fashion, error is propagated from the hidden neuron C, to the input layer (A and B), after which, the weights are updated.

E. Activation Functions

An artificial neuron, as a part of any neural network calculates a "weighted sum" of data fed to it, adds a bias term and subsequently decides whether to activate it or not. An activation functions is responsible for defining the output Y of a particular node/neuron and deciding whether or not a neuron is triggered. Non-linear activation functions such as sigmoid and gaussian wavelet have been used for modelling non-linear sequences such as speech signals, given their popularity and efficiency.

The Sigmoid activation function is the most common and popular activation function used in neural networks. It is non-linear in nature with a characteristic "S" shape, as shown in Fig.6 and is bounded between 0 and 1 , allowing the output to be interpreted as a probability.

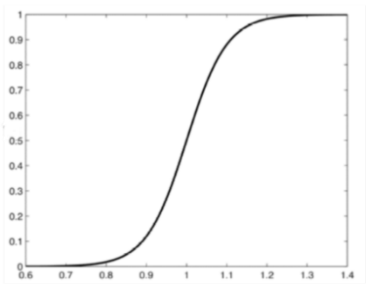


Fig.6 Sigmoid Function

It is a good choice for speech recognition, since it performs decision making on the basis of probability of word detected/matched and is cheaper in terms of computation time and complexity, both of which are critical for embedded applications.

Fig. 7 shows the Gaussian Wavelet activation function which comprises of a complex exponential carrier multiplied by a Gaussian window/envelope. This activation function is a good choice for speech recognition applications because it is closely associated with human perception of sound, promising good results without much computation complexity. Wavelets have been found to be insensitive to signal distortion with striking robustness even after multiple simulations and 20% better performance than Fourier Methods[7].

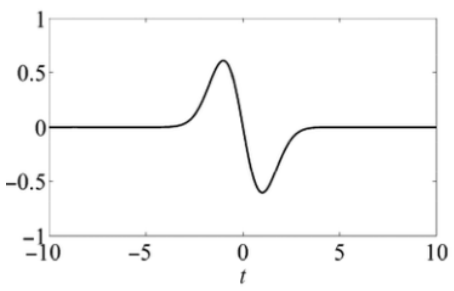
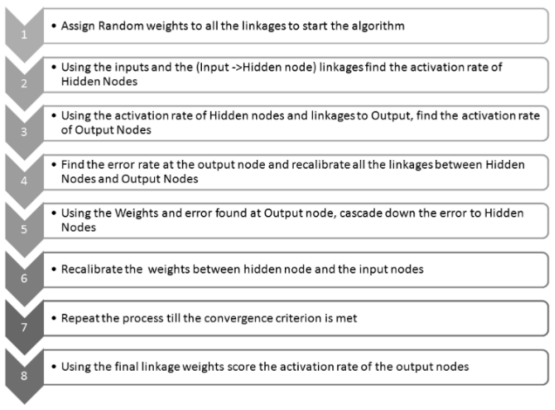


Fig.7 Gaussian Wavelet

Building a neural network is computationally intensive and expensive in terms of system resources and time. Embedded boards such as Raspberry Pi have limited computing power, and as a consequence cannot train as well as test a neural network because of the very nature of backpropagation algorithm, which requires numerous iterations to configure the weight vectors and the activation function which adds to computations per iteration.

A simple solution to allow functioning of neural network on Raspberry Pi is to train the network offline on personal computer running MATLAB, extract the updated weight vectors and other parameters from the workspace and feed it to Raspberry Pi running GNU Octave, an open-source scientific computing alternative to MATLAB.



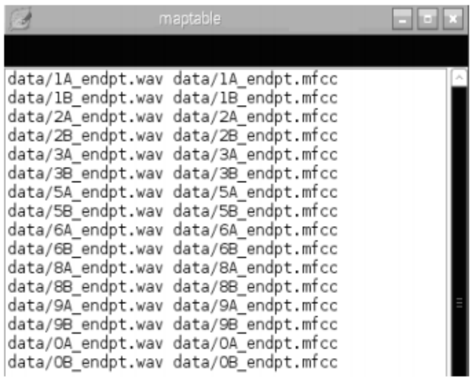
1. • Assign Random weights to all the linkages to start the algorithm
2. • Using the inputs and the (Input ->Hidden node) linkages find the activation rate of Hidden Nodes
3. • Using the activation rate of Hidden nodes and linkages to Output, find the activation rate of Output Nodes
4. • Find the error rate at the output node and recalibrate all the linkages between Hidden Nodes and Output Nodes
5. • Using the Weights and error found at Output node, cascade down the error to Hidden Nodes
6. • Recalibrate the weights between hidden node and the input nodes
7. • Repeat the process till the convergence criterion is met
8. • Using the final linkage weights score the activation rate of the output nodes

Fig.8 Neural Network Process Flow

The Raspberry Pi makes use of testing algorithms with parameters optimized for its hardware capabilities and utilizes the trained model to provide isolated spoken word recognition capability in a small package, with low power consumption and reliability.

III. IMPLEMENTATION & RESULT

MFCC Feature Extraction is done from .wav files using HTK tool kit, which is an open source software. The speech signal, which consists of two samples from a single person, are used as input data. The samples contain the uttered digits from zero (0) to nine (9). HTK toolkit installed in Ubuntu (Linux Distribution) for performing feature extraction. Fig. 9 shows MFCC files extracted from ".wav" speech files.

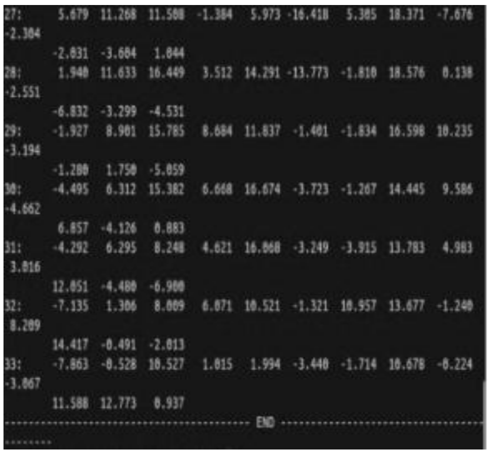


```
data/1A_endpt.wav data/1A_endpt.mfcc
data/1B_endpt.wav data/1B_endpt.mfcc
data/2A_endpt.wav data/2A_endpt.mfcc
data/2B_endpt.wav data/2B_endpt.mfcc
data/3A_endpt.wav data/3A_endpt.mfcc
data/3B_endpt.wav data/3B_endpt.mfcc
data/5A_endpt.wav data/5A_endpt.mfcc
data/5B_endpt.wav data/5B_endpt.mfcc
data/6A_endpt.wav data/6A_endpt.mfcc
data/6B_endpt.wav data/6B_endpt.mfcc
data/8A_endpt.wav data/8A_endpt.mfcc
data/8B_endpt.wav data/8B_endpt.mfcc
data/9A_endpt.wav data/9A_endpt.mfcc
data/9B_endpt.wav data/9B_endpt.mfcc
data/0A_endpt.wav data/0A_endpt.mfcc
data/0B_endpt.wav data/0B_endpt.mfcc
```

Fig.9 MFCC Files with Features Extracted from Speech Data

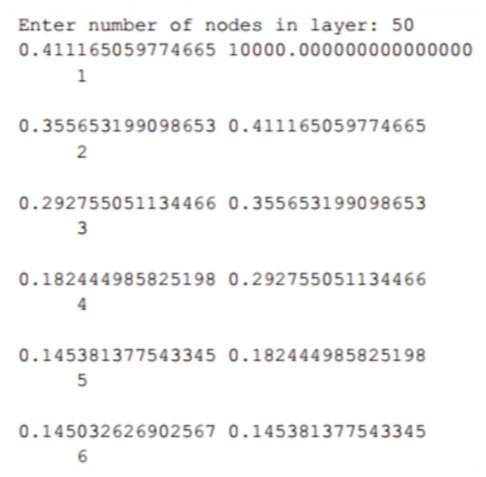
*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

Fig. 10 shows the extracted MFCC features. The training part of the project is implemented offline i.e. using a computer running MATLAB. The network is assigned 50 nodes in hidden layer and trained using the input given in the form of extracted MFCC Features using both Sigmoid and Gaussian Wavelet activation functions. The biases and weight vectors are obtained after training the network and saved as a workspace. Fig. 11 shows the biases and weight vectors of the BPNN using the Gaussian wavelet activation function.



```
27:      5.679  11.268  11.500  -1.304   5.973 -16.410   5.305  10.371  -7.676
-2.304
        -2.031  -3.604   1.044
28:      1.940  11.633  16.449   3.512  14.291 -13.773  -1.010  10.576   0.138
-2.551
        -6.032  -3.299  -4.531
29:     -1.927   8.901  15.785   0.604  11.037  -1.401  -1.034  16.598  10.235
-3.194
        -1.200   1.750  -5.059
30:     -4.495   6.312  15.382   6.668  16.674  -3.723  -1.267  14.445   9.586
-4.662
         6.057  -4.126   0.083
31:     -4.292   6.295   8.240   4.621  16.060  -3.249  -3.915  13.783   4.983
 3.016
        12.051  -4.480  -6.900
32:     -7.135   1.306   8.009   6.071  10.521  -1.321  10.957  13.677  -1.240
 8.209
        14.417  -0.491  -2.013
33:     -7.063  -0.528  10.527   1.015   1.994  -3.440  -1.714  10.678  -0.224
-3.067
        11.588  12.773   0.937
------------------------------------ END ------------------------------------
--------
```

Fig.10 MFCC Features

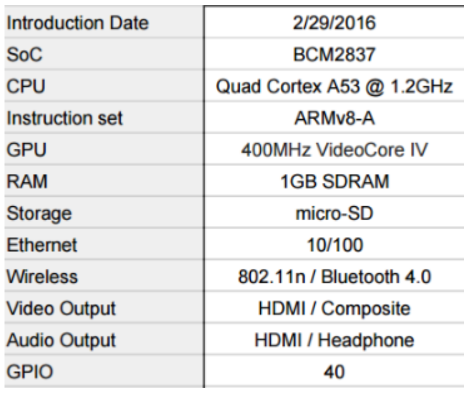


```
Enter number of nodes in layer: 50
0.411165059774665 10000.000000000000
     1

0.355653199098653 0.411165059774665
     2

0.292755051134466 0.355653199098653
     3

0.182444985825198 0.292755051134466
     4

0.145381377543345 0.182444985825198
     5

0.145032626902567 0.145381377543345
     6
```

Fig.11 Updated Weight Vectors and Error Terms (Offline MATLAB Training)

The hardware used for implementation and online testing is a Raspberry Pi 3B Board, setup and loaded with a memory card with custom Debian distribution called Raspbian installed, which is a very minimalist and lightweight OS designed for Raspberry Pi.

The Raspberry Pi has a Quad-Core 1.2 Ghz Broadcom BCM2837 64bit CPU and 1GB RAM with specifications shown in Fig.12 , which is sufficient for running testing script with the training data (updated weight vectors etc.) loaded in form of MATLAB workspaces .The Pi is also outfitted with a USB Type A connector rated at 2.5A, allowing it to be powered by any consumer-grade power bank and operate almost anywhere for long periods of time. The Raspberry Pi setup has been shown in Fig.13



Introduction Date	2/29/2016
SoC	BCM2837
CPU	Quad Cortex A53 @ 1.2GHz
Instruction set	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
Storage	micro-SD
Ethernet	10/100
Wireless	802.11n / Bluetooth 4.0
Video Output	HDMI / Composite
Audio Output	HDMI / Headphone
GPIO	40

Fig.12 Raspberry Pi 3B Specifications

Raspberry Pi was setup headless i.e. without any physical display monitor attached to it. Instead, a Virtual Network Computing (VNC) Server was configured for display mirroring through a Static IP Address, allowing for added portability and mobility in terms of interfacing. The Raspberry Pi can be controlled and interfaced on-the-go using a smartphone on the same network. GNU Octave is chosen for testing on Raspberry Pi mainly because it is a popular open-source alternative for MATLAB, with full code and syntax compatibility, allowing for streamlined workflow between training on personal computer using MATLAB, exporting extracted parameters as workspace and using the training data on the Raspberry Pi locally to achieve the same.

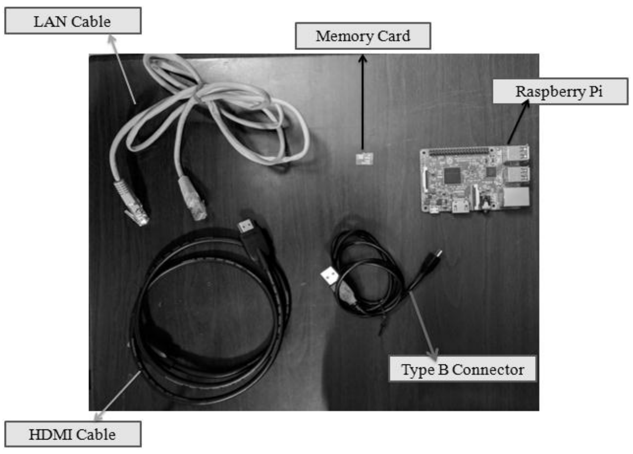
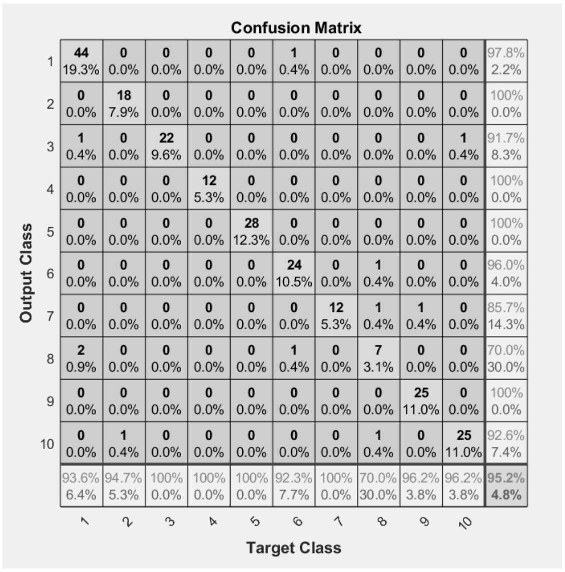


Fig.13 Raspberry Pi Setup

The speech data set was split into various train-test portions to test robustness in isolated spoken word recognition accuracy. Confusion matrices for each of these cases, for training and testing offline on MATLAB yielded ≈90-95% accuracy for isolated spoken word recognition, as shown in Fig. 14



	1	2	3	4	5	6	7	8	9	10	
1	44 19.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	97.8% 2.2%
2	0 0.0%	18 7.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	1 0.4%	0 0.0%	22 9.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.4%	91.7% 8.3%
4	0 0.0%	0 0.0%	0 0.0%	12 5.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	28 12.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	24 10.5%	0 0.0%	1 0.4%	0 0.0%	0 0.0%	96.0% 4.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	12 5.3%	1 0.4%	1 0.4%	0 0.0%	85.7% 14.3%
8	2 0.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.4%	0 0.0%	7 3.1%	0 0.0%	0 0.0%	70.0% 30.0%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	25 11.0%	0 0.0%	100% 0.0%
10	0 0.0%	1 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.4%	0 0.0%	25 11.0%	92.6% 7.4%
	93.6% 6.4%	94.7% 5.3%	100% 0.0%	100% 0.0%	100% 0.0%	92.3% 7.7%	100% 0.0%	70.0% 30.0%	96.2% 3.8%	96.2% 3.8%	**95.2% 4.8%**

Fig.13 Confusion Matrix (95.2% Accuracy) for OfflineTraining and Testing

IV. CONCLUSION

When the Neural Network was trained and tested offline i.e on MATLAB, fairly accurate results with 92% accuracy for the standard 80/20 training-to-testing ratio (80% of data used to train the neural network and 20% the remaining data to test it), reaching a fairly good 95.2% accuracy for 95/5 training-to-testing ratio. On the Raspberry Pi, using extracted weight vectors and biases, we got a maximum of 50% accuracy, while testing on 8 datapoints, keeping in mind the limited processing capability and memory of Raspberry Pi. For faster processing, a much-optimized encoding and bit-feeding scheme for 'stream analytics' can be used for testing on individual data points coming in for live processing purposes. The reduced accuracy can be attributed to the unavailability of an optimized configuration for Octave on the Raspberry Pi, being roughly configured for an 'arm-unknown-linux-gnueabihf' model. The neural network and it's accuracy have been tested using several activation functions such as i) Sigmoid ii) Hyperbolic Tangent(tan-h) and iii) Gaussian Wavelet, out of which the Gaussian Wavelet activation function has given us the highest accuracy. Furthermore, as future work, the model can be trained and tested using more promising activation functions such as the Morlet Transform and Fast Morlet Transform and also work towards optimizing the configuration for Octave environment on Raspberry Pi to exploit the hardware capabilities more efficiently and speed up computation.

REFERENCES

1. Chandralika Chakraborty, P.H.Talukdar, "Issues and Limitations of HMM in Speech Processing: A Survey", International Journal of Computer Applications(0975-8887), Volume 141 – No.7, May 2016
2. Steve Renals,David McKelvie and Fergus McInnes,"Comparative Study of Continuous Pattern Recognition Using Neural Networks and Hidden Marcov Model," IEEE 1991
3. "A Ian McLoughlin; Haomin Zhang; Zhipeng Xie; Yan Song; Wei Xiao,"Robust Sound Event Classification Using Deep Neural Networks," IEEE/ACM Transactions on Audio, Speech, and Language Processing,vol-23,pp. 540 - 552,2015.
4. Xian Tang. Hybrid Hidden Markov Model and Artificial Neural Network for Automatic Speech Recognition, Circuits,
5. Communications and Systems, IEEE 2009
6. John Sahaya Rani Alex, Ajinkya Sunil Mukhedkar, Nithya Venkatesan, "Performance Analysis of SOFM based Reduced
7. Complexity Feature Extraction Methods with back Propagation Neural Network for Multilingual Digit Recognition" Networks",Indian Journal of Science and Technology, Vol 8(19), IPL098, August 2015
8. Pranjali P. Patange, John Sahaya Rani Alex "Implementation of ANN Based Speech Recognition System on an Embedded Board"
9. G.Ososkov, A. Shitov "Gaussian Wavelet Features and their Applications for Analysis of Discretized Signals", Elsevier Science, October 2005

AUTHORS PROFILE

Aniruddha Sharma is pursuing his B.Tech in Electronics and Communication Engineering from Vellore Institute of Technology, Chennai

Dr. John Sahaya Rani Alex is an associate professor working in School of Electronics Engineering, VIT Chennai. Her research interests are Speech Processing, Signal Processing, implementation of DSP algorithms on embedded boards and ANN

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*