Turkish Phoneme Classification with Mel Frequency Cepstral Coefficients using Artificial Neural Networks

Göktuğ Kayacan, Remzi Orak, Sefa Alp

***Abstract*— Phonemes are the smallest part of the speech, by designing metrics such as Mel frequency cepstral coefficients for the different kinds of phonemes, it becomes possible to differentiate between these sounds by using a parametric classification model. By classifying different sounds, words can be constructed, creating a highly dynamic statistical automated speech recognition system.**

***Index Terms*— MATLAB, Automatic Speech Recognition, Digital Signal Processing, Artificial Neural Networks, Pattern Recognition**

I. Introduction

This project's main goal is to implement a phoneme recognition system using Mel Frequency Coefficients (MFCC) to distinguish between different phoneme using a parametric classification model such as a feedforward neural network or a recurrent neural network. The system should be able to produce outputs for a given set of MFCC features of a frame with the size of the frames that are used to train the model. The result from the network should be a list of label possibilities for the given features.

II. Problem specification

Project is consistent of constructing a system to analyze and frame the given file and label each of these frames accordance to the alignments and segments of the file. By doing this a features and labels vectors can be constructed. Features vectors can be made up of a variety of data, in most cases MFCC and frame energy is used however, delta and delta-delta features which are the differential and acceleration coefficients are not used. This project makes use of both MFCC and frame energy in all test cases and a single use of delta (differential feature) in one case. The resulting feature and label sets are given to an artificial neural network (ANN) built by using MATLAB's ANN Toolkit. The results are tested in terms of both phonemes matches and word matches. The primary performance measure in this case is the number of word matches using the primary parametric output and the secondary parametric output (first and second outputs from the competitive layer of ANN).

III. Data

The data that is used for feature extraction is 30 minutes long 3 different audio files, sampled at 1.6kHz. Each of these audio recordings are taken from a Turkish TV News show and shows similar specifications in terms audio attributes.

Data is processed by chaining each audio file into a 90 minutes long audio signal. Signal is then framed into 25 milliseconds long frames with each frame being shifted by 10 milliseconds. This results all built frames to be processed in feature and label extraction.

Feature extraction uses MFCC, frame energy and delta calculation methods. The label is found by finding the temporal location of the frame and finding if the frame is within a given range of a phoneme [2].

The given raw data is made up of segments of speech which indicates the start time and end time of speech, alignments which describes the phonemes within a given segment in terms of start of the phoneme, middle of the phoneme, end of the phoneme and "individual" phoneme that does not exist within a word but is a word on its own.

The data segments have been matched with their alignment counterparts to create a time-series data that is non-linear as not every time instance is labeled in data. Processed data's phonemes and start-end time matches are tested using a software capable of labeling audio files such as *Wavesurfer*.

Once the raw data is processed into a time-series data and tested to make sure phonemes are matching the time ranges, data can be processed and analyzed.

While picking data to analyze for the model, thirteen MFCC is usually enough, note that due to the high rate of change of the first MFCC, its excluded [2].

IV. Evaluation criteria

Evaluation is going to be based primarily on the word matches per sentence, secondary evaluation is going to be based on the number of letters that are correct within a word. Secondary output of the model is also going to be considered in the evaluation, in a case where two different output possibilities are very close it might be necessary to look at both outputs to deduce if the model came close to correctly calculated the label of that frame.

.

Evaluation is going to be done on both RNN and FFNN to show a measure of performance and how model memory effects the outcome of the model.

V. Approach

Implementation of all the algorithms are done on MATLAB including MFCC calculation, Mel filterbank calculation, data separation, data analysis and model. Python scripts are also utilized for data processing and testing. This section details implementation and theoretical information for each of these modules and describes how they are used.

A. *Data Separation*

Data separation involves aligning phonemes in respective segments. Each segment contains several alignments of phoneme with start and end times as offsets in time. By building an accumulator for start and end time each phoneme can be listed in terms of start times in seconds and end time in seconds. All three audio files are also chained together with each chaining audio file being offset by the total runtime of the previous audio files. This produces one large audio file.

Also due to the memory limitations of MATLAB it is impossible to create a model for the total of 120 labels that is in the data. Therefore, data has been rasterized to 30 total labels which 29 letters plus Silence. This damage the integrity of the dataset and precision of the system but does not prevent the system from producing relative results.

B. *Data Analysis*

Data analysis involves framing and windowing of the audio file and attaching phonemes to each individual frame. This is one of the more computationally challenging parts due to the sheer amount of data that is being processed. It should also be noted MATLAB is incapable of containing arrays larger than a certain size and direct implementation of a function to frame and label the whole audio file is going to result in a system error insufficient memory on most systems. As a workaround the audio file can be separated into chunks of equal size, issue with this is that there are going to be incomplete chunks as there is not going to be enough samples to fill the last frame and the frame needs to be padded with zeros or discarded. In this project case, recording has been split into 153 equal size chunks, with the frame length that is calculated, it is possible minimize the amount of padding down to a single sample per chunk. By doing this each frame is have same size, no sample is discarded and 153 new samples as 0 are added which equates to the less than 0.1% of the total amount of samples. Following *Figure-1* Shows the process of framing a signal.

During phoneme matching of the data it should also be noted that the total segmentations lengths does not directly correlate to the length of voiced area within the audio recording. This results in certain parts of the audio file becoming non-labeled. This is one the reasons for picking a correct separation that results in the minimal amount of appended empty samples or removed frames as the result is going to be ever so slightly skewed depending on the ratio of the added samples or the removed samples. Although data analysis and data separation may be a small part of this project it's the most important part as every other module in this project is built on correct implementation of these methods.

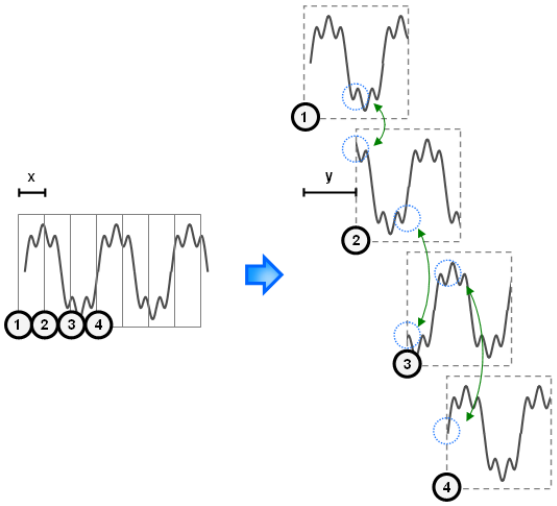


Figure-1: Signal framing

C. *Mel Filter Bank*

Mel filter bank is a set of triangular filters with differing frequency ranges. These filters are shaped to simulate hearing ranges of humans. The scale that these shapes are based on is called the *Mel Scale* which relates to perceived frequency or pitch [4]. Humans are better at hearing lower-frequencies and composition of *Mel Scale* reflects this. *Figure-2* is plot of our implementation of a *Mel Filter Bank* calculator.

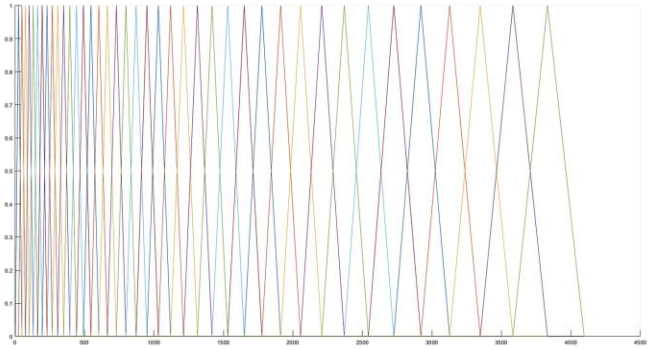


Figure-2: Mel Filter Bank 0hz to 4kHz

Figure given here is drawn using 8192-point fft, and only the first positive half is shown.

Calculating this filter bank involves calculating the *Mel Scale* conversion of hertz and vice versa. The conversion equations are given as *Equation-1* and *Equation-2*.

$$M(f) = 1125 * \ln(1 + \frac{f}{700}) \quad \textit{Equation-1}$$

$$M^{-1}(m) = 700 * (e^{\frac{m}{1125}} - 1) \quad \textit{Equation-2}$$

Computation of these frequencies involves calculating a lower and higher binding frequency values. A linear space with N number of elements is computed between these values. After which *Equation-2* is applied to this space to convert the space back to hertz domain. This results in a sequence of frequency values determining the peak points of the N number of filters. This is followed by rounding the produced frequency values and turning them into fft-bin values within the range of zero and fft points divided by two. These values can be utilized in *Equation-3* to produce the triangle filters [3] [4].

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \ \ or \ \ k > f(m+1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \end{cases}$$

Equation-3

By applying *Equaiton-3* for each filter peak value, a series of overlapping frame can be built. By using this filterbank it becomes possible to calculate MFCC Features.

D. MFCC calculation

Calculation of MFCC features is a more involved in terms of computational complexity and different methods of normalization and optimization. Due to these reasons its not good to solely use raw MFCC. Most common method of normalization is to compute the mean and the standard deviation of each coefficient and normalize the sequence of vectors from an utterance by subtracting the mean and dividing by the standard deviation. There are more advanced statistical modelling techniques such as Gaussian model that learns from the mean and covariance matrix of the coefficients without any pre-emphasis [4] [3] but for this case an implementation of a more advanced normalization model is out of scope.

The calculation of MFCC requires few steps, including the ones that have already been discussed in this report. Steps to calculating MFCC at a glance follows as;

1. Apply Pre-emphasis filter
2. Frame the signal and applying windowing.
3. Calculate Power of each frame (Periodogram estimate)
4. Apply Mel Filterbank to power spectra, sum energy in filters
5. Take logarithm of all filter banks.
6. Take discrete cosine transform of log F.B. energies.
7. Add Frame energy, discard first MFCC.

Calculating MFCC requires use of DFT (Discrete Fourier Transform) in the form of fft and calculation of the power spectral estimate for a given instance of speech. This can be calculated using the equation given as *Equation-4.*

$$P_i(k) = \frac{1}{N} * |S_i(k)|^2 \quad \textit{Equation-4}$$

Multiplying this value by the filterbank that was discussed in the previous section, an estimate of total amount of energy in each frequency range can be found by summing up the multiplied energies. For a total of M amount of filterbank filters, M number of energies are calculated. After taking discrete cosine transform (DCT) of these energies, a new set of energy values are made that emphasizes certain energies. These values are the MFCC features [1].

Once the MFCC values are calculated, they should be normalized by one of the aforementioned methods which should lift certain features and suppress others to provide a more meaningful relationship between these features, creating a dataset that can be used for statistical and parametric classification for the individual labels of the frames. The usual labels are for four different forms of each phoneme. This makes differentiating the features harder and may require calculation of Delta and Delta-Delta features for the frames. However, this is out of scope for this project.

Figure-3 shows difference between a non-normalized frame MFCC and normalized frame MFCC.

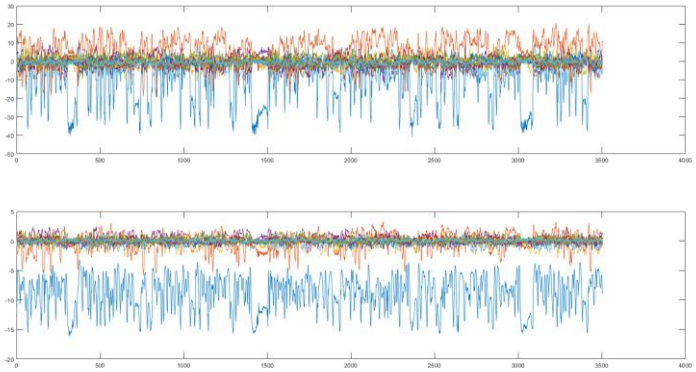


Figure-3: Raw and Normalized MFCC values over frames

Normalization in *Figure-3* is done using lifting and mean-variance method which was explained earlier in this section.

E. Model

Model is the classifier that distinguishes between the labels, given the features. This process involves training the model by giving both features and their corresponding labels, the method of training is out of the scope of this project. Model is constructed using MATLAB's ANN-Toolkit. Two different models are constructed, one using Feed-Forward-ANN and one using Recurrent-ANN. Recurrent ANN is like a conventional shallow ANN but has a memory of its output. This memory can be utilized to chain individual inputs, in this case chaining individual phonemes. This allows for dynamic between different phonemes to be learned by the model. *Figure-4* and *Figure-5* shows two different network views, built using this toolkit.

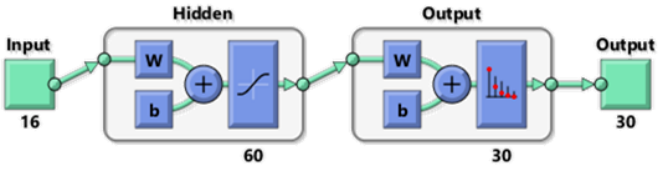


Figure-4: Feed Forward ANN

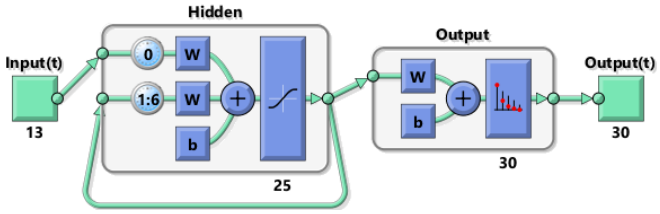


Figure-5: Recurrent ANN with memory of 6 inputs

The model is consistent of two layers, the hidden layer which utilizes sigmoid function to parameterize the input values and the output layer which is a competitive layer that does classification by using softmax function. This function converts the output into a series of probabilities for any of the possible labels occurring. This allows for comparisons between different possible outputs if the output possibilities are close to each other. Note that the ANN structures given in

the figures are not the only networks that are implemented in this project, a variety of different combinations of parameters relating to the number of hidden neurons, transfer function of the hidden layer, number of labels and number of MFCC features have also been tested. Due to the large amount of data and lack of a strong machine to train more complicated networks some of these networks have only been tested on the surface. Therefore, does not have conclusive results that can be included in the report.

VI. Results, Analysis and Evaluation

Due to the problems that have not been considered before, time restriction and lack of proper device to train a network with the large amount of data, the results that are produced are sub-par and requires observation to determine the performance of the overall system and word success rate.

The framing process creates a total of **536,265** frames which are all labeled either **NaN** if frame falls to an unvoiced region, or a phoneme if the frame falls in a voiced region. The number of voiced region frames is **337,234** frames. Each frame is 400 samples in size. Calculated power of each frame looks similar to the figure given as *Figure-6.*

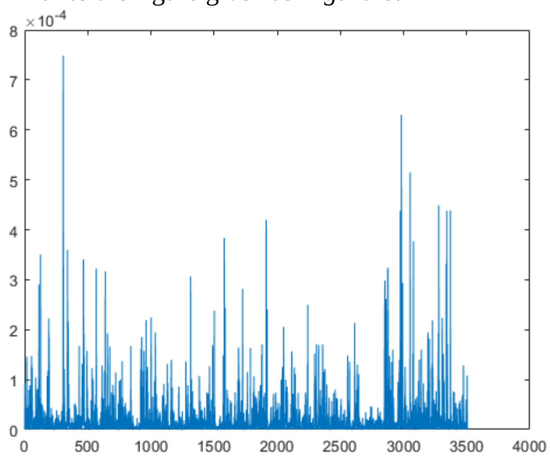


Figure-6: Sample frame power spectra

After calculating the power spectrum, these values are multiplied with filterbank given in the previous sections. This results in set of MFCC feature vector that can be represented in the form given in *Figure-7*.

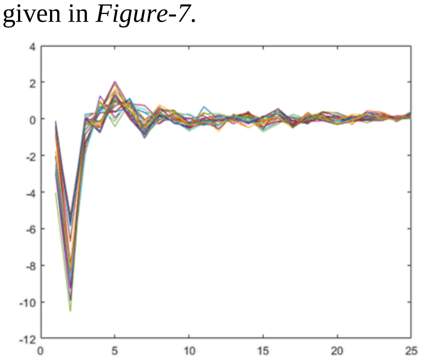


Figure-7: MFCC feature representation

By doing this process for all of the frames, a representation for all of the MFCC features can be made. This representation has been given in *Figure-8* and *Figure-9* as the normalized and non-normalized MFCC feature representations of the whole audio file respectively.

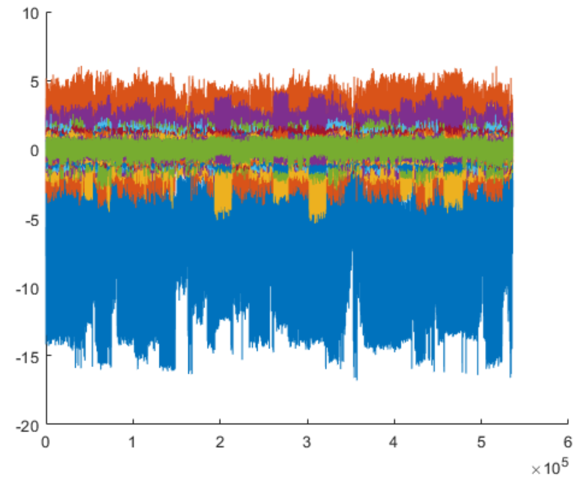


Figure-8: Normalized MFCC features for recording

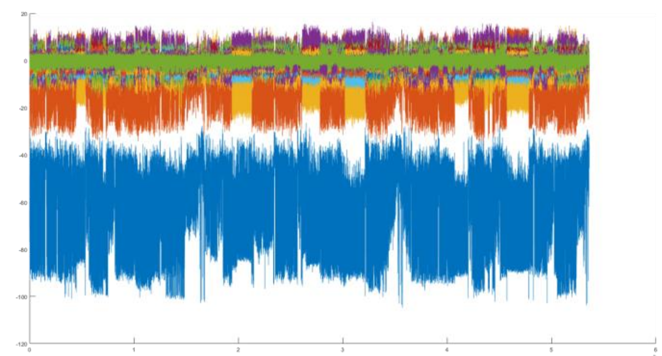


Figure-9: Non-Normalized MFCC features for recording

Running a recurrent ANN model on the normalized MFCC features results in an overall success rate of **88.8%** success rate on the validation set. Due to the rasterized labels, network is unable to recognize the following phoneme features: b, c, f, g, ğ, h, i, j, ö, p, v, y. Network instead puts the maximum value of these to surrounding phonemes. This makes it impossible to compute word rate computationally. For a given sentence:

"polis bağdat'ta düzenlenen diğer saldırlarda da ondan fazla kişinin öldürüldüğünü bildirdi"

Network's output frame-wise is:

"rrrrroooolllllliiiisssss__aaçaaçaaaaaiiidddaaaaatttttttaaaadddduuuyyyyyyyeeeeeennnlleeennnnneeeeeeee ennnnnnddiiiiiiiiieeeeeerrrrrsssssssssaaaaaaaallllldiiirriiiiillllaaaaaaaarrrrddddaaadddddaaaaaaaaaaaa___ ______________________oooooooooooooonnnnnddaaaaaaaannnnneeeeeeaaaaaaayyyllakkkkkkkkiiiisssss siiiiinnnniiiiiiiinnnnnnnnnnnnssssssllllllldduuurruuuuullllldyiiunnuuuuçiiiilllldiiiiirrrddiiiiiiiiiii_______ ____aaaaaaaaayyyrriiiççççaaaaarrrrrooooolllllliiiiis sssssssdddddyyuuuuuuuuuyunnnnnnnnnnnnnaççaaaaaaiiidddaaaatttttttiiiiinnnnnnnnnnççççeeeessssssiiii ttlliikkkkkeeeeesssssssiimmmmllleeeeerineeeeaaaaaattttttiiilmmmmmmmmmiiiiiiissssssssss_____________ __________________oooooooooootttttttuuuuuuyyyyyaaaaaaaaaaaaaaaasssss"

Which can be put through a rasterizer to eliminate repeating frame. This method is usually time-warping method, the method used here simply removes repeating adjacent letter-phonemes. The output is given below.

"_rolis_açaçaidataduyenlenendier**saldirilardada_ondan**eayla **kisinin**slduruldyiunuçildirdi_"

From this output, its seen that normalized MFCC features in combination with a short-memory, shallow RNN model is not enough to make a proper classification. Other reasons that this model might have failed is the lack of more advanced statistical method when normalizing the data, making no use of delta feature or delta-delta features or improper/insufficient model architecture due to use of a single hidden layer shallow ANN or use of short memory on RNN.

It should also be noted that other Architectures and other feature and label sets have been tested in a variety of formations and settings, majority of the resolutions either led to model being too big and slow or results were very similar to the one that is described above.

Further test have been done using different sentences, any sentence that contains the letters that are aforementioned gives bad results and majority of "silent" areas are not recognized by the network and instead act like an expansion of the previous frame labels, this is assumed to be because of the RNN memory and short silences between the speaker's words. Although evaluation has to be done per word correction basis, it is impossible to do with current results.

VII. Development and Future Work

Due to the incompleteness of the project, future work should involve re-analysis of the dataset and inclusion of all the possible labels. MFCC features should be coupled with delta features and delta-delta features to provide acceleration and differentiation statistics of each frame. This is going to result in feature sets containing more information which is going to lead to building a more proper model such as an LSTM model (long short time memory). Although like RNN, LSTM models are more apt at dealing with large amounts of data. Constructed model may also be built following the architecture patterns of deep neural networks by coupling multiple hidden layers. This will result in exponentially increasing system requirement. By using different services such as Google's cloud training service this kind of network can be trained and tested, although this would require implementation to be expanded to different scripting languages such as Python.

Although there is no conclusive result, working on this project has involved studying problems involving a variety of different areas. This is one reason that the research and amount of experimentation done with different methods and models is relatively high. Given more time, this project could reach maturity with the implementation and addition of the methods and features given above.

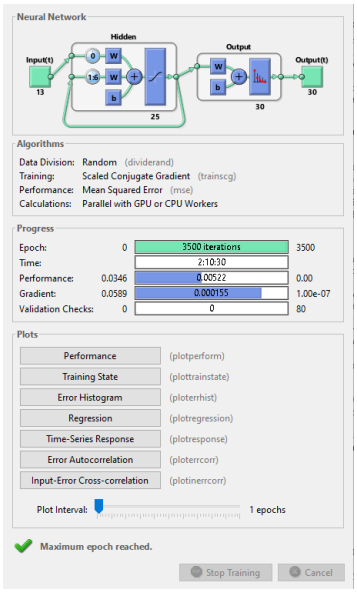
VIII. Conclusion

Although the project is incomplete, this study showed how automatic speech recognition systems function on a fundamental basis. The research done for this study has been encompassing on all parts of this process, from analysis and separation of the data, calculation of features, type of features and what features are used in which cases, how features correlate with signal aspects such as energy and frequency and how these specifications relate to aspects of human hearing abilities and study of psychoacoustics. The different categorization and classification models used to differentiate between phoneme utterances and how these models are tested and validated using number of words matching.

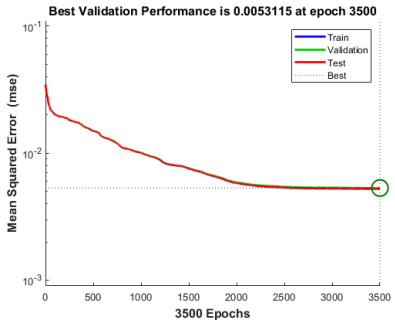
Unfortunately, project haven't come to fruition, but the amount of information synthesized during this study could very much produce a tangible result given more time is spent on the implementation of the model and addition of delta and delta-delta features to the dataset.

Appendix

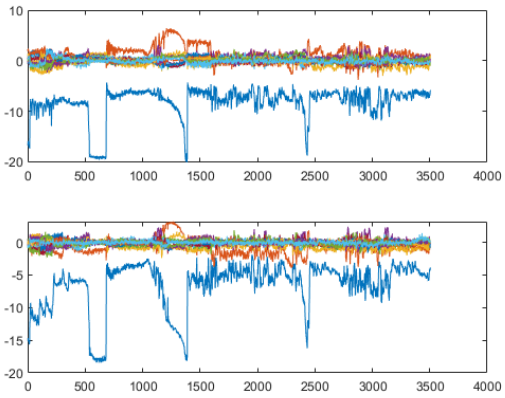
Due to large amounts of code and the size of the data, all related files are attached to this report, including a pre-trained network function which can be used by calling "customNetworkFunction_RNN_35K" function with data generated from attached file "MainAlternate.m".



Appendix-1: MATLAB ANN training results for RNN

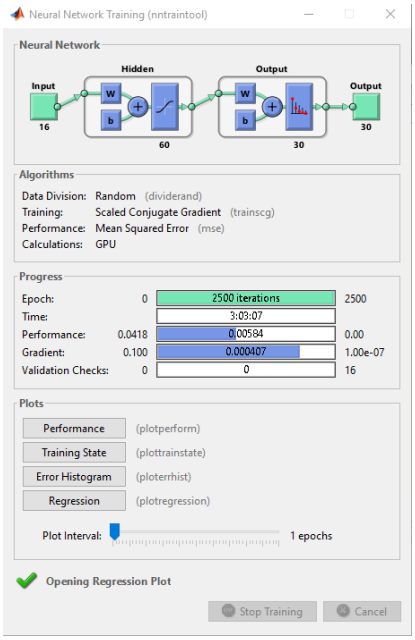


Appendix-2: Performance of RNN over epochs during training



Appendix-2

Appendix-2 is a comparison between high-variance normalized and non-normalized MFCC features for over 3.5k frames.



Appendix-3: MATLAB ANN training results for FFNN

This is the model that the system was originally tested on, it produced results similar to the RNN with less accuracy. This model also was completely incapable of recognizing silences while still being unable to recognize certain phonemes.

ACKNOWLEDGMENT

We would like to express our very great appreciation to Dr. Ebru Arısoy Saraçlar for her valuable insights and help on this project. Her willingness to help us during the extend of this project has been very much appreciated.

REFERENCES

[1] Welch, P. (1967). "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms". IEEE Transactions on Audio and Electroacoustics. ***15*** *(2): 70–73.*

[2] Davis, S. Mermelstein, P. (1980) Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *In IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28 No. 4, pp. 357-366*

[3] X. Huang, A. Acero, and H. Hon. Spoken Language Processing: A guide to theory, algorithm, and system development. *Prentice Hall, 2001.*

[4] *Tyagi and C. Wellekens (2005), On desensitizing the Mel-Cepstrum to spurious spectral components for Robust Speech Recognition, in Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on, vol. 1, pp. 529–532.*