Deep Learning Techniques for Speech Recognition and Audio Transcription for Content Captioning

D. G. J. B. Wimalarathne[1], S. C. M. de S Sirisuriya[2]
[1,2]General Sir John Kotelawala Defence University

Abstract

Speech recognition has taken a tremendous interest in the field of Natural Language Processing. Deep Learning has been playing an increasingly large role in speech recognition and one of the most exciting things about this field is that speech recognition is at a place right now where it is becoming good enough to enable exiting applications that end up in the hands of users. Studies have shown that efficiency of devices increases 3 times, with the availably of speech recognition on daily driving devices like mobile phones, cars etc. Content captioning or generating subtitles is very useful and important since the use of videos for communication have been grown phenomenally in the past few years. However, for this, it still need human influence to do a good captioning. But, it is possible to do this with a better efficiency and accuracy with deep learning. This paper provides an overview of how traditional ASR pipeline process speech transcription and how deep learning techniques used to replace modules in traditional ASR pipeline. Finally, describes the implemented speech recognition neural network model built on Tensor flow platform. This model is capable of recognizing English spoken digits from 0 to 9 and transcript it to text. Currently model has achieved around 70.19% accuracy over different training and testing data. This prototype model is reusable for any amount of words in any language with the availability of more data and computing power.

Keywords: *Speech recognition, Feature extraction, Acoustic model, Language model, Connectionist Temporal Classification, MFCC, Tensor flow, Librosa*

1. Introduction

Speech-to-text has achieved a motivating development in the modern world. Numerous forms of end devices like Personal Computers, hand held smart devices and even trendy cars are integrated with this feature for the convenience of the end user.

There are different components that make up a complete speech applications like speech transcription, word spotting/trigger word and speaker identification/verification. Speech transcription is essential to come up with words that represent what is being said. Applications, which need to do an action or get triggered for voice, need continuous listening. In addition, some applications may use speech recognition for authentication purposes too. For a content captioning application, it need to have a good speech transcription model.

This paper presents the basic flow of how traditional Automatic Speech Recognition systems like Hidden Markove Model process audio transcriptions, then how deep learning techniques have been influenced to traditional ASR pipeline. How the audio is preprocessing and generating simple spectrogram and how CTC (Connectionist Temporal Classification) approach do speech recognition with generated spectrogram. And finally presents the speech recognition neural network that have been developed by using Google machine learning platform Tensor flow, for the system that generates subtitles for a given audio/video.

2. Traditional ASR Pipeline

Most of the traditional ASR systems are based on probabilistic theories. [1]Traditional system break the problem of taking audio and breaking it into different parts. The raw audio has represented with some kind of feature representation.

Job of the acoustic model is to learn relationship of features representation of audio and the sequence of words W, words of the language.[2] A language model contains the structural constraints available in the language to generate the probabilities of occurrence. It induces the probability of a word occurrence after a word sequence.[3]

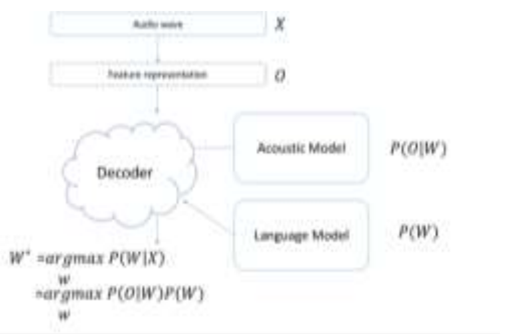


Figure 1: Modules of ASR

The job of the decoder is to find the sequence of word W that maximize the particular sequence of words W given the audio.

$$W^{*} = \underset{w}{argmax}\, P(W|X) \qquad (1)$$

That is equivalent to maximizing the product of contribution from acoustic model and language model.

$$= \underset{w}{argmax}\, P(O|W)P(W) \qquad (2)$$

However, it is not directly possible to transcribe in to characters especially in English language because way something is spelled is not always correspond to its sound. There for a way to get around this issue is to that replace this with the sort of intermediate called phonemes. So instead of trying to predict characters, it predicts the phonemes by the acoustic model.[4]

Example w_1 = “hello” = [HH AH OW] = [$q_1q_2q_3q_4$]

In the above example, if the word the word that need to be predicted should be hello, instead of characters, acoustic model predicts the corresponding phonemes. This in one sense makes modeling problem easier. Phonemes are the perpetually distinct units of sound that distinguish words. They are very approximate. It is not clear how fundamental this is. Therefore, this phoneme representation adds more complexity to this traditional pipeline. Because, acoustic model does not associate with words, instead it associates with another kind of transcription into phonemes. Therefor pipeline needs kind of dictionary or a lexicon to understand how it convert phonemes into words. This pipeline is highly tweakable[5], but also hard to get work well. Each part of it has its own challenges like choosing representation.

3. Deep Learning Approach

The first place that deep learning made an impact on speech recognition is to take one of the core machine learning component of the system and replace it with a deep learning algorithm. One of the fundamental problems in speech recognition is to build a neural network that map audio signal to a transcription that have a variable length. Unlike an image, which has 2D grid of pixels, audio is just 1D signal. One second of audio clip it can broke down in to small parts and each element would be a floating-point number.[6]

A. *Preprocessing*

There are two ways to start the preprocessing.

1) *Minimally pre-processing (by simple spectogram)*
2) *Training model raw audio wave*

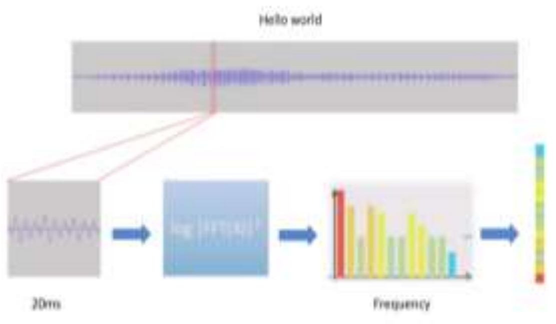


Figure 2: Small window of audio in terns on frequency

Minimally processing is some vanilla preprocessing like convert to a simple spectrogram. Mel-Frequency cepstral coefficients is a popular kind of features, and engineer complex representations. Spectrogram is sort of like a frequency domain representation. Figure 2 represent a small window in term of frequencies. [7]

The small window is about 20ms long. It can see that when it gets down to this scale, that audio waves are made up of combinations of different frequencies of sine waves. Then it has to compute Fourier transformation of the sine values and then take the log of the power of each frequency. Result of this is, that for every frequency of sin waves, what is the amount of magnitude represent by that sine wave that makes up the original signal. Therefor transformation need to apply to all the windows of frames in audio signal. This will result a creation of spectrogram.

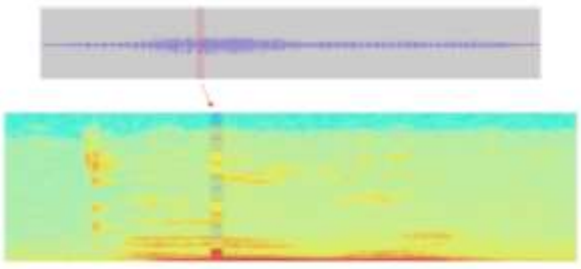


Figure 3: Spectrogram for all the window of frames

4. CTC Based End-To-End Speech Recognition

Recent years with the availability of high computing power and more data, the speech transcription has been developed and equipped with many powerful tools and libraries.[8] CTC is one of the successful approach because it explicitly uses the history of the target character without condition.

CTC is a probabilistic mode p(Y|X) where X is sequence of frames of data. Y is output tokens of data of equal length of L. X is the spectrogram.[9]

$$X = x_1, x_2, x_{31} \dots . x_T \quad (3)$$
$$Y = y_1, y_2, y_{13} \dots . y_L \quad (4)$$
$$T \geq L$$

The spectrogram frequencies feed into recurrent neural network as the input. The neural network is a bidirectional RNN. There, for any time step is depend on entire data, so it can compute fairly complex function over entire data. Output is bank of Softmax neurons.[10] They represent the distribution over different symbols (Softmax vocabulary), that could be hear in the audio. For example, for the first frame of audio, it will be processed by RNN and for output, it has Softmax neurons that correspond to each of the neuron.

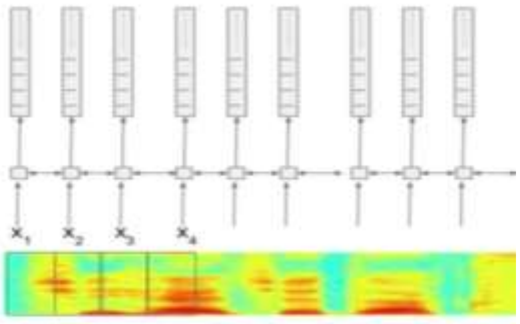


Figure 4: Spectrogram feeding sequence of frames of data

In addition, the set of Sofmax neurons represent the probability of containing each symbol in the spectrogram frame and their output is summing to 1.

For an English word transcription, the Softmax vocabulary would be {a, b, c, d, ...z,} and extra blank token. Black is a special character for identification of intervals and rest times of the speech. Softmax at step t, gives a score s(k,t) =logPr(k,t|X) at the category k in the output at time t.

Therefore, the target is to find a transcript through this individual Softmaxes over the time. It is essential that each symbol must go through a blank symbol. You can transfer either from same character to itself or from that symbol to a blank. Therefor it can end up different sequences, representing the same symbol.

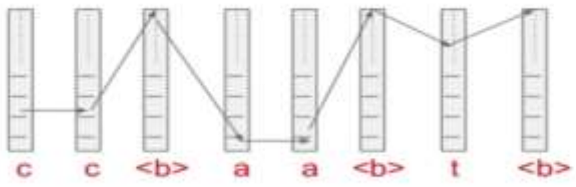


Figure 5: One possible sequence of transcription for word cat

In figure 5, it represents one possible sequence that can be obtain for the transcription for word cat.

Following are some possible sequences as well.

cc <b>aa<b>t<b>
cc<b><b>a<b>t<b>t<b>
ccccc<b>aa<b>tttt<b>

Output neurons define distribution over whole character sequences c, assuming independence:

$$P(c|x) \equiv \prod_{i=1}^{N} P(c_i|X) \quad (5)$$

This process makes a transcription of same length to the audio. Therefore, the second step is to define a mapping that crunches down this sequence into actual transcription. Given a specific sequence c, squeeze out the duplicates plus blanks that yield to transcription.

$$\beta(C) \rightarrow y$$
$$\mathrm{Y} = \beta(C) = \beta(\mathrm{cc\_aa\_\_t\_}) = \mathrm{cat} \quad (6)$$

The probability of transcription is the sum of the probabilities of all the possible character sequences that is correspond to the audio.

$$P(Y|X) = \sum_{c:\beta(c)=y} P(c|x) \quad (7)$$

Because of the dynamic programing, it is possible to compute both the log probability p(Y|X) and its gradient exactly. This gradient can be propagated to neural network by backpropagation and update network parameter θ to maximize likelihood of correct label y*.[11]

$$\theta^* = argmax \sum_i logP(y^{*(i)} | x^{(i)}) \quad (8)$$

Today the above process is at the level of technology and implemented that efficiently calculate CTC functions cost function that can calculate the likelihood and also can give back the gradient.

Some of the common issue of CTC based speech recognition are "Tchikovky" problem, relationships in between words, and identifying proper nouns. "Tchikovky" problem is there are certain word which you cannot spell unless you have seen the exact spelling for that problem (knife, Tsunami), since P(Y|X) molded directly from audio. Therefor with the CTC implantation it, fuse a language model by taking small step back from end to end transcription system.

5. Speech Recognition Model

The following system has been implemented and currently at the training and testing stage which is a sub part of the developing system that generates content captioning (subtitle generation) for a video/audio input, by using deep learning techniques.

The key was to give more data and computing power. These types of neural networks are used in services like Siri, Echo and Google Now. The following speech recognition deep neural network is constructed with the Google machine learning framework Tensor flow that learns to recognize spoken numbers. A labeled data set that people saying numbers are used to train the network and currently testing it from validation data in order to improve the accuracy.

A. *Fetch Data*

The model is trained with set of audio files. Each wave file is a recording of different spoken digit. These files contain different people (male/female) speaking numbers from 0 to 9 and they have variable lengths. Each file is labeled with a written digit. A function is written such that it returns list of labeled speech files as a batch. So, this batch is a set of audio files and their corresponding label of written digit. This batch is use for training the neural network from supervised learning technique.

All these utterances are examples of red speech. Audio that are clear with no noise. Choosing a data set is highly dependable with the application that is going to be develop. Reason that red speech is popular is because of its availability. Moreover, even it does not perfectly match to the application, it is free and getting a lot is helping the accuracy. But, it will lose factors such as inflection/conversational tone, Lombard effect and speaker variety. However, it still can be helpful.

B. *Feature Extraction*

Next step is feature extraction process from the audio files. The extracted features and number of features to be extracted depend on application that is going to develop. For subtitle generation, feature extraction, system uses Mel-frequency Cepstrum Coefficients (MFCC)[12], the power spectrum that implements Fourier transformation analysis since input data are red speech data (clear with no noise).

MFCC features are set of specific features defined for set of applications constrains. MFCC values are not very robust in the presence of additive noises it is common to normalize their values in speech recognition system to reduce the influence of noise. Many audio analysis library packages include MFCC[13] feature extraction and the for the system LibROSA[1] python package for audio music analysis have been used.

C. *Neural Network*

Network consist of 3 layers input layer, hidden Long Short Term Memory (LSTM) recurrent neural network layer and output Softmax layer.

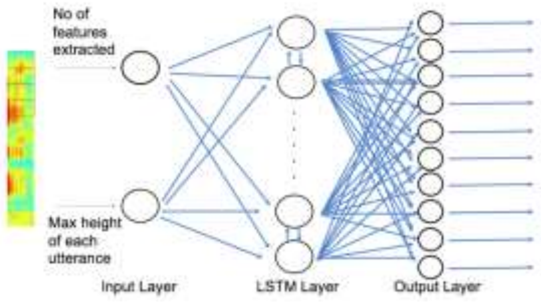


Figure 6: Neural network model diagram

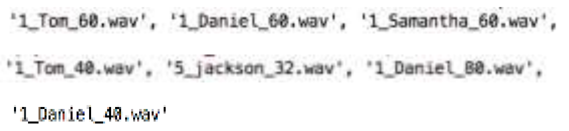
Initial input layer takes output of MFCC features as the input. Here, the input parameters are number of features extracted and the maximum height of each utterance. This input layer initializes by calling tflearn's input data function. This initial input layer will be the gateway to that data is fed into the network and parameter it will help to define shape of input data (input tensor)[14]. Tensor is multi-dimensional array of data.

Next layer is the hidden layer which is Long Short Term Memory Recurrent[15] neural cells layer. A recurrent net was used since its processing sequences. In a recurrent net, the output data's content is influenced not only by the input just put in, but also by the entire history of inputs thorough recurring loop. LSTMs can remember everything it's been fed.[16] Too few neurons will lead to bad predictions and too many will over fit to training data. Currently 128 LSTM neurons have been used for the model.

Final layer is fully connected Softmax layer. There are 10 classes since model is predicting numbers from 0 to 9. Each Softmax cell provide a probability of occurrence of that label, for the given input. Output layer is a regression, which will output single predicted number for the utterance. Adam optimizer have been used to minimize categorical entropy loss function.

D. Training the Model

Model has been trained on Tensor flow platform. Number of epochs and neurons for LSTM layer have been modified randomly at training epochs

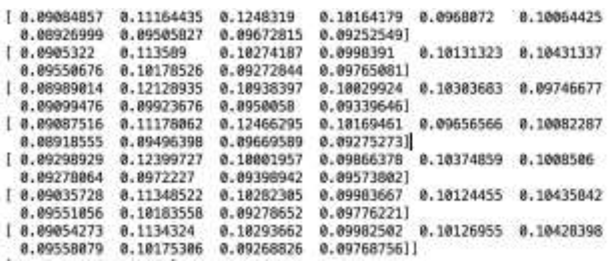


```
'1_Tom_60.wav', '1_Daniel_60.wav', '1_Samantha_60.wav',

'1_Tom_40.wav', '5_jackson_32.wav', '1_Daniel_80.wav',

'1_Daniel_40.wav'
```

Figure 7: Sample of testing data

Figure 7 shows sample of set of training data. The label of the file. Label gives the 3 information, number spoken in the file, person who spoked the number and the maximum length of the utterance. At the stage of fetching data, these labels and audio are separated and features are extracted from the audio, and text digit of the spoken number is extracted from the label. The above sample shows the testing data which people are speaking 1,1,1,1,5,1,1 and 1. This batch the audio wav files and the corresponding labels used to train the neural network.

At the output, each Softmax neuron represent a digit from 0 to 9. Finally, Softmax



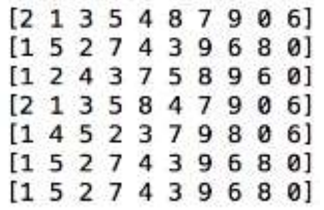
```
[ 0.09084857  0.11164435  0.1248319   0.10164179  0.0968072   0.10064425
  0.08926999  0.09505827  0.09672815  0.09252549]
[ 0.0905322   0.113509    0.10274187  0.0998391   0.10131323  0.10431337
  0.09550676  0.10178526  0.09272844  0.09765081]
[ 0.08989014  0.12128935  0.10938397  0.10029924  0.10303683  0.09746677
  0.09099476  0.09923676  0.0950058   0.09339646]
[ 0.09087516  0.11178062  0.12466295  0.10169461  0.09656566  0.10082287
  0.08918555  0.09496398  0.09669589  0.09275273]
[ 0.09298929  0.12399727  0.10001957  0.09866378  0.10374859  0.1008506
  0.09278064  0.0972227   0.09398942  0.09573802]
[ 0.09035728  0.11348522  0.10282305  0.09983667  0.10124455  0.10435042
  0.09551056  0.10183558  0.09278652  0.09776221]
[ 0.09054273  0.1134324   0.10293662  0.09982502  0.10126955  0.10428398
  0.09558079  0.10175306  0.09268826  0.09768756]]
```

Figure 8: probabilities produced by the Softmax layer for each wav file in the sample

function of the output neurons (Softmax neurons) will produce a probability for the given input for being that digit.

Figure 8 shows probabilities produced by the neural network, for the sample data in figure 7. One array contains probability of that spoked number being 0 to 9. So together, there are 7 arrays for each 7 audio files. Output of the network is an array of array of probabilities for multiple inputs.

Next probabilities are sorted descending from highest accuracy to lowest accuracy and the corresponding digit for that probability has been returned. Therefor the final output is an array for each audio input such that it contains predicted digit with highest confidentially to lowest confidentially.



```
[2 1 3 5 4 8 7 9 0 6]
[1 5 2 7 4 3 9 6 8 0]
[1 2 4 3 7 5 8 9 6 0]
[2 1 3 5 8 4 7 9 0 6]
[1 4 5 2 3 7 9 8 0 6]
[1 5 2 7 4 3 9 6 8 0]
[1 5 2 7 4 3 9 6 8 0]
```

Figure 9: predicted digits from its highest confidentiality to lowest

In above example, out of 7 inputs, the model has predicted 3 correctly. This result may differ for same inputs in another instance.

E. Testing and Evaluation

The neural network model, currently trained with different training iterations, by changing number of neurons in hidden layer and also changing the drop out value. Model was trained over 2400 audio data with both male and female voices on 2.7 GHz intel processor.

Currently it shows 70.19% accuracy on the Tensor flow platform. This is a simple prototype model that can be configurable according to language and also the type of application. Currently, this neural network model has integrated to offline subtitle generation application as the speech recognition engine.

Application takes video/audio file of mp4 or mp3 format. It recognizes digits from 0 to 9 spoken in the content and transcript it to text. Then application will create a subtitle file (.srt format) for generated text. Subtitle file is timely synced along with the audio so it can play the video file with the subtitle file.

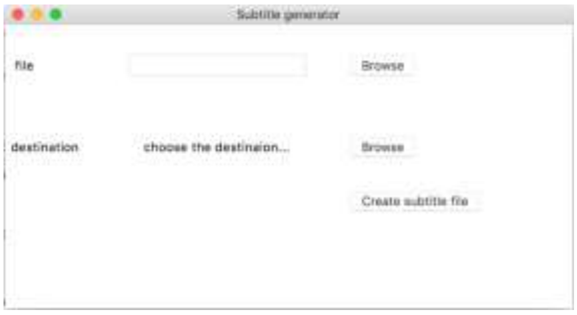


Figure 10: GUI interface of the subtitle generating system

This prototype neural network is reusable for any language with any amount of words with the availability of data and the computing power. Integrating neural network model will decrease the space and runtime complexity of the system than traditional ASR pipeline since there is no any word dictionary or lexicon to map phonemes and hence no mapping. However, since there is a lack of data to train some words like proper nouns, it is good to have some language model process too.

6. Conclusion

With the proliferation of Speech-to-text technology, it is clear that it has become an essential technology in almost every electronic device. Content captioning is an application area which we can use ASR efficiently but it still needs human influence to do a better captioning. Therefore, it is essential to have a good speech recognition system for these type of applications

Traditional Automatic Speech Recognition pipeline comprises of different modules Feature representation, Acoustic model, and Language model. Phonemes are the intermediate in between extracted features and acoustic model because way something is spelled is not always correspond to its sound. Therefor a dictionary or a lexicon is need to map words and phonemes. This phonetics to word mapping make the traditional ASR pipeline more complex.

Deep learning techniques can be used to replace those modules in traditional ASR pipeline. Conventionalist Temporal Classification (CTC) is such an implementation and it provide a solution for the problem of mapping audio signal to a transcription that have a variable length without having any intermediate of phonetics.

Today CTC based implantations are at a level of technology and lot of libraries are available for such implementations. Speech recognition neural networks can be implement to generate content captioning (subtitle generation) and these models can develop by having more data and computing power and can be adopt into many words in many languages.

A simple neural network model has been developed for content captioning application. This neural network model has used to replace the Acoustic model in the traditional ASR pipeline. Currently, this model can recognize English spoken digits from 0 to 9 and transcript it into text. Since there is no any phonetic interaction, this model makes less runtime and space complexity. In addition, the model is reusable for any language with the availability of the data and computing power.

This speech recognition neural network has been integrated to the subtitle generation application. It is an offline application that takes video or audio file as the input and provide .srt formatted subtitle file with timely synced digits. Since the reusability of the neural network model, the application can develop for many language transcriptions or different language transcriptions.

References

[1] B. McFee *et al.*, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[2] M. A. Anusuya and S. K. Katti, "Speech recognition by machine, a review," *ArXiv Prepr. ArXiv10012267*, 2010.

[3] L. Deng and X. Li, "Machine Learning Paradigms for Speech Recognition: An Overview," *IEEE Trans. Audio Speech Lang. Process.*, vol. 21, no. 5, pp. 1060–1089, May 2013.

[4] A. Mathur, T. Saxena, and R. Krishnamurthi, "Generating Subtitles Automatically Using Audio Extraction and Speech Recognition," 2015, pp. 621–626.

[5] S. K. Gaikwad, B. W. Gawali, and P. Yannawar, "A Review on Speech Recognition Technique," *Int. J. Comput. Appl.*, vol. 10, no. 3, pp. 16–24, Nov. 2010.

[6] D. Gerhard, University of Regina, and Department of Computer Science, *Audio signal classification: history and current techniques*. Regina: Dept. of Computer Science, University of Regina, 2003.

[7] N. Q. Duong and H.-T. Duong, "A Review of Audio Features and Statistical Models Exploited for Voice Pattern Design," *ArXiv Prepr. ArXiv150206811*, 2015.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[9] K. S and C. E, “A Review on Automatic Speech Recognition Architecture and Approaches,” *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 9, no. 4, pp. 393–404, Apr. 2016.

[10] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, “Multi-category classification by soft-max combination of binary classifiers,” *Mult. Classif. Syst.*, vol. 2709, pp. 125–134, 2003.

[11] M. K. Brown, M. A. McGee, L. R. Rabiner, and J. G. Wilpon, “Training set design for connected speech recognition,” *IEEE Trans. Signal Process.*, vol. 39, no. 6, pp. 1268–1281, 1991.

[12] K. R. Ghule and R. R. Deshmukh, “Feature Extraction Techniques for Speech Recognition: A Review,” *Int. J. Sci. Eng. Res.*, vol. 6, no. 5, pp. 2229–5518, 2015.

[13] J. Martinez, H. Perez, E. Escamilla, and M. M. Suzuki, “Speaker recognition using Mel frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques,” in *Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on*, 2012, pp. 248–251.

[14] M. Abadi *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *ArXiv Prepr. ArXiv160304467*, 2016.

[15] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[16] H. Palangi *et al.*, “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval,” *IEEEACM Trans. Audio Speech Lang. Process. TASLP*, vol. 24, no. 4, pp. 694–707, 2016.