Summer Internship Report

on

Voice Activity Detection using Machine Learning Technique

Manya Dave

IIT Ropar

Supervised by

Prof. Arun Kumar

Centre for Applied Research in Electronics (CARE)

Indian Institute of Technology, Delhi

May-July 2018

Acknowledgment

I would like to express my sincere gratitude to my guide Prof. Arun Kumar for his invaluable guidance, constructive criticism, and constant encouragement.

I would like to thank Shimoli Shinde Ma'am and Goverdhan Sir for their co-operation and guidance, and for helping me throughout, from the learning stage to the implementation stage.

I would also like to acknowledge Mr. Purshottam who has always been very helpful for the smooth conduct of the work.

Table of Contents:

List of Figures:

List of Abbreviations:

VAD: Voice Activity Detection

SPP: Speech Presence Probability

ASR: Automatic Speech Recognition

SNR: Speech to Noise Ratio

DFT: Discrete Fourier Transform

SVM: Support Vector Machine

GMM: Gaussian Mixture Model

DNN: Deep Neural Network

LSTM: Long short-term memory

CNN: Convolutional Neural Network

NAT: Noise-aware training

RNN: Recurrent Neural Network

BLSTM: Bidirectional long short-term memory

ANN: Artificial Neural Network

MFCC: Mel frequency cepstral coefficient

HPS: Harmonic Product Spectrum

LP: Linear Prediction

CPP: Cepstral Peak Prominence

SRH: Sum of Residual Harmonics

MS: Modulation Spectrum

GPU: Graphics Processing Unit

AUC: Area under the curve

ROC: Receiving Operating Characteristics

EER: Equal error rate

DBN: Deep Belief Network

BPTT: Backpropagation through time

1) Introduction

Voice activity detection (VAD) (or speech activity detection, or speech detection) refers to a class of methods which detect whether a sound signal contains human speech or not.
A closely related and partly overlapping task is speech presence probability estimation. Instead of a present/not-present decision, SPP gives a probability level that the signal contains speech. A VAD can be derived from SPP by setting a threshold probability above which the signal is considered to contain speech. [1]

VAD a fundamental task which finds a wide range of applications in voice technology: speech coding, automatic speech recognition (ASR), audio surveillance and monitoring, speech enhancement, or speaker and language identification. In the workflow of these applications, VAD is generally involved as the very first block. The identification of non-speech regions is useful in reducing the amount of processing required. Consequently, the main characteristics expected from a VAD algorithm are generally a high efficiency and robustness to noise, as well as a low computational latency.

2) Literature Survey

2.1 General Approaches for Voice Activity Detection:

There are three main approaches for the implementation of a VAD algorithm

1. Thresholding rule-based approach
The thresholding decision rules are the most commonly used techniques due to their simplicity and low complexity. These techniques often require the features extracted to be sufficiently discriminating and noise robust. However, when SNR drops, the feature space is no longer linearly separable, causing the performance to degrade drastically. More sophisticated non-linear approaches are therefore required. They are helpful for rather clean speeches.

2. Statistical model-based approach
In the statistical modeling approaches, speech or noise has been modeled statistically in different domains like in time domain and DFT domain. Sometimes speech models are estimated from a large speech set developed off-line, and noise models are estimated during an initial silence period, or sometimes initial speech statistics has been assumed to be a Laplacian, Gaussian or Gamma distribution. Then, a likelihood ratio test has been applied to make decisions.

3. Machine learning based approach

Methods such as Support Vector Machine, Neural Networks, and GMM have been used for VAD. These have higher complexity but a better performance is reported. Machine learning based VADs have gained importance due to the following reasons. First, the machine-learning-based VADs can be integrated into the speech recognition systems naturally. Second, they can fuse the advantages of multiple features much better than traditional VADs.

As concluded in [2] the thresholding based VADs have lower computation time but higher false alarm rate as compared to machine learning based VADs. Among the studied VAD algorithms in, SVM based VAD has the highest percentage of speech detection. But the difference in the percentage of speech detection of machine learning based VAD and percentage of speech detection of SVM based VAD is not significant. The machine learning based algorithm has lower false alarm percentage and less computational complexity than SVM based VAD. Thus, machine learning proves to be the best approach for Voice Activity Detection. In [2] Drugman VAD approach has been used.

VAD Algorithm	ITU-T Recording		MEMS Microphone Recording	
	P_d	P_f	P_d	P_f
CVAD	81.73	32.91	88.62	47.78
G.729B	86.43	31.80	91.22	57.18
Sohn	82.92	25.70	77.26	21.13
Davis	94.6	46.21	97.65	68.12
ANN classifier with Drugman feature set	**86.32**	**2.94**	**84.42**	**2.83**
SVM classifier using MFCC features	79.65	3.32	67.98	2.25
SVM classifier using Drugman feature set	86.60	3.70	81.31	3.23

Fig. 2.1 Comparison of performance of VAD algorithms

2.2 Machine Learning based methods:

In [3] three different deep learning approaches for Voice Activity Detection, i.e. VAD models based on Deep Neural Networks (DNN), Long-short term Memory (LSTM) and Convolutional Neural Networks (CNN) are thoroughly compared at both frame and segment level under various noisy conditions. It has been shown that LSTM is more robust than CNN and DNN under multiple circumstances. Although all deep learning approaches performed well under noise-matched conditions, very large performance degradations were observed in conditions with unseen noise or very low SNR for all approaches. To improve the robustness of deep learning based VAD models a new noise-aware training (NAT) approach is also proposed. By incorporating NAT, significant performance gains can be obtained in these conditions.

Table 4. *Performance of VAD after post-processing.*

Metric	Model	A+B+C+D	SEN+2ND	LOWSNR
EER	DNN	3.52	11.19	29.31
	LSTM	**3.15**	**9.24**	**23.64**
	CNN	5.05	9.76	28.36
$\mathcal{J}_{VAD}$	DNN	**84.66**	48.68	20.80
	LSTM	83.80	**65.39**	**41.47**
	CNN	82.28	55.23	14.43

Fig. 2.2 Comparison of performances of machine learning based algorithms

2.3 Recurrent Neural Networks:

In [4] different types of RNN models and optimization methods are investigated. Three kinds of RNNs are compared: a basic RNN, long short-term memory (LSTM) network with peepholes, and a coordinated-gate LSTM (CG-LSTM) network. Among all the tested SAD methods the CG-LSTM model gives the best results, but it has been noticed that the difference between the performances of LSTM and CG-LSTM is not appreciable, so LSTM based algorithm will be preferred. Similarly, in [7] Bidirectional LSTMs and Augmented BLSTMs (BLSTM+) were compared with Multi-layer perceptrons. Though BLTM+ showed the best results, there was no significant difference between BLSTM and BLSTM+.

	WER	Corr	Subs	Del	Ins
Baseline	57.2	49.2	34.7	16.1	6.4
CrossCorr	56.1	48.8	34.4	16.8	5.0
LTSV	55.6	48.9	33.5	17.6	4.5
MLP	55.4	49.0	34.2	16.8	4.4
BLSTM	54.8	49.1	34.0	16.9	3.9
BLSTM+	**54.6**	**49.0**	**34.1**	**16.9**	**3.6**

Table 2: *Detailed results of the best settings for each algorithm on the Vietnamese evaluation set.*

Fig. 2.3 Comparison of BLSTMs and other Machine Learning techniques

TABLE III
WERs ON THE OpenKWS Evaluation Data for Four Languages After Optimizing Each SAD Model Using the L_{WER} Loss Function

Metric	WER				
Cost function	FER (L_{FER}; $\alpha = 0.5$)	L_{WER}			
Corpus	Vietnamese		Pashto	Turkish	Tagalog
CrossCorr	56.7 (58.8)	56.1 (58.8)	63.9 (64.9)	60.6 (62.9)	57.1 (60.0)
LTSV	57.1 (58.5)	55.6 (58.5)	64.0 (64.5)	60.6 (61.2)	56.7 (56.9)
MLP	56.6 (61.5)	55.4 (61.1)	63.5 (64.2)	60.3 (62.4)	56.2 (59.4)
Basic RNN	56.5 (56.8)	55.2 (56.5)	63.5 (63.8)	60.2 (60.7)	56.3 (57.7)
LSTM	56.5 (56.4)	54.8 (56.0)	63.2 (63.7)	60.2 (60.5)	56.2 (57.5)
CG-LSTM	56.4 (56.2)	54.6 (55.7)	63.2 (63.6)	60.0 (60.5)	56.0 (57.4)

Contrastive results based on the FER and the L_{FER} loss function are given for Vietnamese. Numbers in parentheses are without any QPSO optimization.

Fig. 2.4 Comparison of RNN based methods and other machine learning algorithms

2.4 The selected approach:

Thus, Bidirectional LSTMs were selected as the machine learning model for the problem. The parameters like learning rate, number of nodes, activation function, etc. were selected from [8] which also suggested using Keras toolkit. Thus, Keras toolkit, which works in Python over Tensorflow was selected. [9]

Feature extraction was done using the Drugman Toolkit [10], which was available in MATLAB.

2.5 Previous Work

In work done on “Design & Development of Spatial Environment Sensing Algorithm Using Digital Microphones” by Avani Kulkarni, CARE IIT Delhi, thresholding based, statistics based and machine learning based models were compared for Voice Activity Detection. WSJ Training database and ITU-T Testing database were used. Percentage of speech detected and false alarm rates, Area under ROC curve (AUC) and Equal error rate (EER) were used as the metrics.

The comparison in figure 2.1 showed the machine learning based approach to be the best suitable one, but its computational complexity is not very good. Deep Belief Networks was chosen as the model for Machine learning based algorithm. Features were extracted from the Drugman toolkit. The three proposed feature sets – MFCC, Sadjadi, and New, were compared for different SNR levels, for the DBN model.

SNR	Model trained using MFCC features		Model trained using Sadjadi feature set		Model trained using New feature set	
	P_d	P_f	P_d	P_f	P_d	P_f
0 dB	62.25	**4.09**	62.14	**1.19**	62.31	**1.64**
5 dB	79.8	**3.07**	79.7	**1.79**	79.22	**1.95**
10 dB	87.61	2.19	87.11	3.75	87.08	3.10
15 dB	89.65	1.12	89.53	5.78	89.76	5.08
20 dB	90.53	0.75	90.68	6.15	90.84	4.79

Table 3.4: Comparison of performance using different features. Bold values indicate the false alarm rate obtained using different feature sets at low SNR.

Fig. 2.5 Comparison of performance using different features at different SNRs

Database and metrics in the current work were kept the same as that in the previous work to compare their results. The selected model, BLSTM was expected to give better results than DBN according to the papers followed.

3) Proposed VAD Technique

3.1 Database

3.1.1 WSJ Training Database

1500 clean speech sentences at 8 kHz sampling frequency from 300 speakers are taken from WSJ0 and WSJ1 databases. Speech utterances in WSJ database are in NIST sphere format. MATLAB code in Voicebox toolbox is used to convert NIST sphere format to .wav format. Suppose that the length of the clean speech utterance is 4 seconds, then 4 seconds noise before and 4 seconds noise after the speech is also added to ensure that the training database is roughly balanced between speech and non-speech frames. Clean speech utterances are labeled in speech and non-speech frames of 10ms by applying an energy-based threshold.

3.1.2 ITU-T Test Database

16 clean speech sentences from 5 male and 5 female speakers in 3 languages were downloaded from the ITU-T website. The three languages are American English, British English, and Hindi language. Seven environmental noises (beach, mall, office, house, stadium, street and nature) and white noise are added at SNR -10dB to 25dB in the steps of 5dB. The clean speech sentences are hand labeled in 10ms speech and non-speech frames. The sampling frequency is 8 KHz.

3.2 Features

Drugman feature set [5] has been used for the problem. Drugman features include 13 filter-based MFCC features, 4 voicing features, called Sadjadi Features: Harmonicity, Clarity, Harmonic Product Spectrum (HPS) and Linear Prediction (LP) error; and 3 New source-based Features: Cepstral Peak Prominence (CPP), Sum of Residual Harmonics (SRH) and normalized SRH (SRH*). Before being fed to the Neural Network, the feature vector at time goes through two processing steps. First, the feature trajectories are smoothed using a median filter with a width of 11 frames (5 on each side). Working with a frameshift of 10 ms, this roughly corresponds to the phone scale. This operation allows removing possible spurious values. Secondly, contextual information is added by

including the first and second derivatives, computed using the following finite difference equation: $x'_t = \frac{1}{N}\sum_{i=1}^{N} x_t - x_{t-i}$
This makes a total of 60 features.
To keep working at the phone level the number of contextual frames is set to 10. When in the test, the Neural Network outputs the posterior of speech activity. As the last post-process, the posterior trajectories are smoothed out by a median filter whose width is again set to 11 frames to remove possible erroneous isolated decisions.

[9] shows the use of Modulation Spectrum (MS) features along with MFCC features, which can be calculated by taking the Fourier transform of an MFCC feature. This MS feature, when used along with MFCC features, reduced False Alarm Ratio and False Rejection Ratio. So, I had decided to use this MS feature along with the 20 other features.

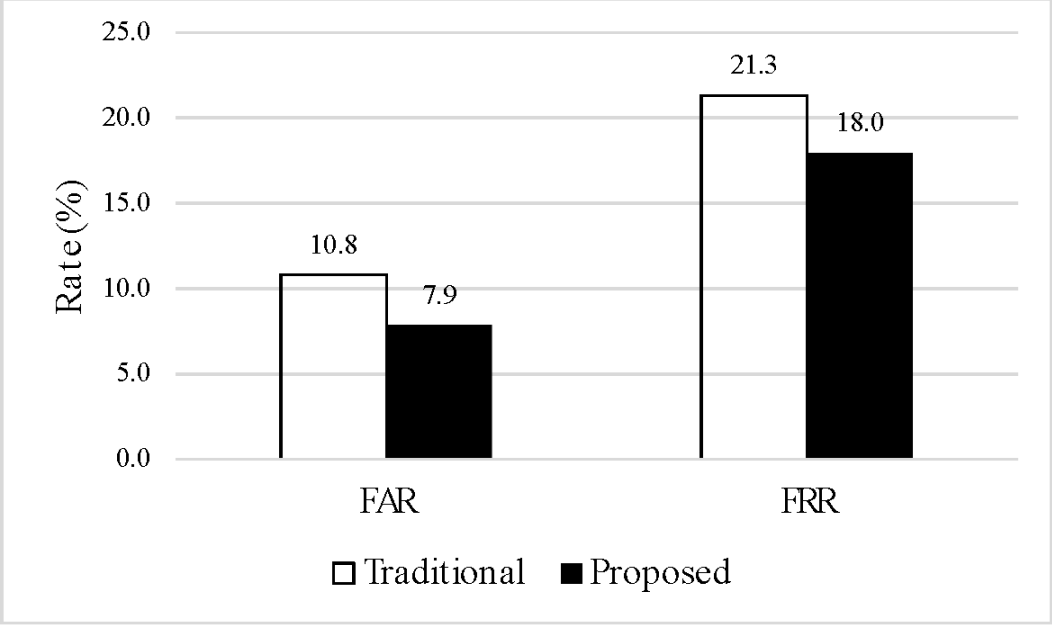


Fig. 3.1 FAR and FRR with and without using MS feature

But as there were already 60 features for the input to the network, the 3 MS features (MS feature, its first and second derivatives) do not seem to give a significant difference in the results. There were a few other problems while training the network, which will be discussed in section 3.

According to the mechanism of voice production, speech is considered as the result of a glottal flow (also called source or excitation signal) filtered by the vocal tract cavities. A

VAD exploits information from both these two complementary components of speech. With the combined effect of the 4 main factors [5]: the joint use of filter and source-related information, the design of robust source features, an efficient strategy of information fusion and multi-condition training, an optimum VAD technique can be formed.

3.2.1 Filter-based features: MFCC

According to the source-filter model of speech, the spectral envelope, defined as a smooth function passing through the prominent peaks of the spectrum, is the transfer function of the filter. In this work, the Mel Frequency Cepstral Coefficients (MFCCs) are considered which are robust high-resolved representations of the filter resonances.

Mel-frequency cepstral coefficients (MFCCs) [13] are coefficients that collectively make up an MFC. Cepstral is defined as the log power spectrum of the log power spectrum. They are derived from a type of cepstral representation of the audio clip. The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound.

MFCCs are derived as follows:

1. Take the Fourier transform of a windowed excerpt of a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using overlapping triangular windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

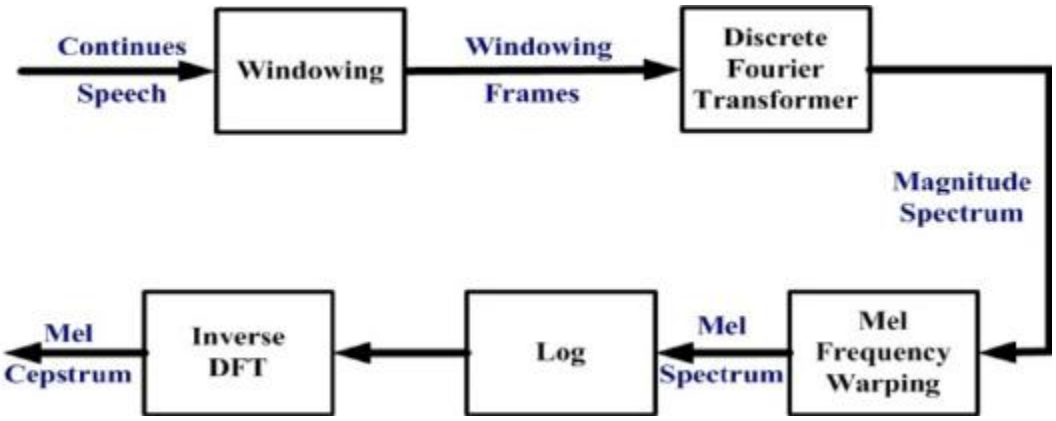


Fig. 3.2 Method to extract Mel frequency cepstral coefficients from the speech sample

A vector of 13 coefficients is used for the feature type. In addition, the dynamic features; their delta and delta-delta coefficients were used.

3.2.2 Source related features: Sadjadi and New feature sets

In this work, we aim at using robust source-related features which are compatible with the noisy environments targeted by our VAD. Two feature sets were made [5] named Sadjadi feature set and New feature set.

Sadjadi Feature set:

- Harmonicity: Harmonicity is defined as the relative height of the maximum autocorrelation peak in the voice pitch range. The autocorrelation of a periodic signal is also periodic is with the same period, and its maximum takes on values close to the autocorrelation at zero lag. Accordingly, for voiced segments which have a periodic structure, the harmonicity shows sharp peaks.

- Clarity: Clarity is defined as the relative depth of the minimum average magnitude difference function valley in the human speech range. Clarity exhibits large values for voiced and speech-like segments while maintaining a minimum for background sounds.

- Prediction Gain: Prediction gain is defined as the ratio of signal energy to the linear prediction (LP) residual energy. The signal energy is obtained from the

autocorrelation at zero lag.

- Harmonic Product Spectrum (HPS): HPS is a frequency domain feature in which 2048-point DFT of Hamming windowed frames is taken after zero padding. It is based on the principle that for voiced speech, compressing the frequency scale by integer factors would cause harmonics of the fundamental frequency to coincide at the fundamental frequency.

New Feature Set:

- Cepstral Peak Prominence (CPP): CPP is a measure of the amplitude of the cepstral peak corresponding to the fundamental period, normalized for the overall signal. A cepstral peak is first located between the minimum and maximum expected fundamental period. To normalize overall amplitude a linear regression line is calculated relating frequency to cepstral magnitude. The CPP measure is the difference in amplitude (in dB) between the cepstral peak and the corresponding value on the regression line that is directly below the peak.

- Sum of Residual Harmonics (SRH and SRH*): An auto-regressive modeling of the spectral envelope is first estimated from the speech signal s(t) and the residual signal e(t) is obtained by inverse filtering. The order of AR model is 12. This whitening process has the advantage of removing the main contributions of both the noise and the vocal tract resonances. The amplitude spectrum E(f) is computed. From this spectrum, and for each frequency in the range [F0min, F0max], the Summation of Residual Harmonics (SRH) is computed as:

$$SRH(f) = E(f) + \sum_{k=2}^{N}[E(k.f) - E(\left(k - \frac{1}{2}\right).f]$$

The number of harmonics considered is 5. The SRH feature value is taken as the maximum of SRH(f) vector for the given frame. Two SRH values have been calculated, one for the original spectrum (SRH) and other for the normalized spectrum (SRH*).

3.3 Model and Network

Two strategies to merge the information of the features were presented in Drugman's paper: feature fusion and decision fusion (also called early and late fusion). In the feature fusion case, synchronous feature vectors are simply concatenated, and a single

ANN is trained. In the decision fusion case, one specific ANN is trained for each feature set. Each ANN outputs a trajectory of posteriors, and the trajectories from the various ANNs are further merged to derive one final posterior value. In [5], the arithmetic and the geometrical mean have been tried. The differences in performance were however negligible, and thus the geometrical mean has been used throughout the work.

3.3.1 Long Short-term memory

Unlike other machine learning algorithms like deep neural networks (DNN) or convolutional neural networks (CNN), Recurrent Neural Networks (RNN) have network loops in them which allow the information to persist [11], i.e. to predict the current output, RNNs make use of the current input as well as the previous output. This helps in predicting the outputs which have a kind of pattern in them.

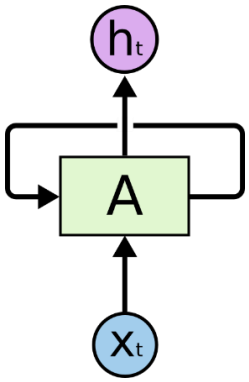


Fig. 3.3 A Recurrent Neural Network cell

Long short-term memory [12] is a particular kind of RNN in which the current output depends not only on the current input and the previous output but also on the outputs of the past. LSTM-RNNs are capable of learning long-term dependencies, and so they work tremendously well on a large variety of problems. A common LSTM unit is composed of *a memory cell, an input gate, an output gate* and *a forget gate*. The cell remembers values over arbitrary time intervals, and the three *gates* regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data since there can be lags of unknown duration between important events in a time series. LSTMs

were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

here the initial values are $c_o = 0$ and $h_o = 0$ and the operator o denotes the element-wise product. The subscript t indexes the time step.

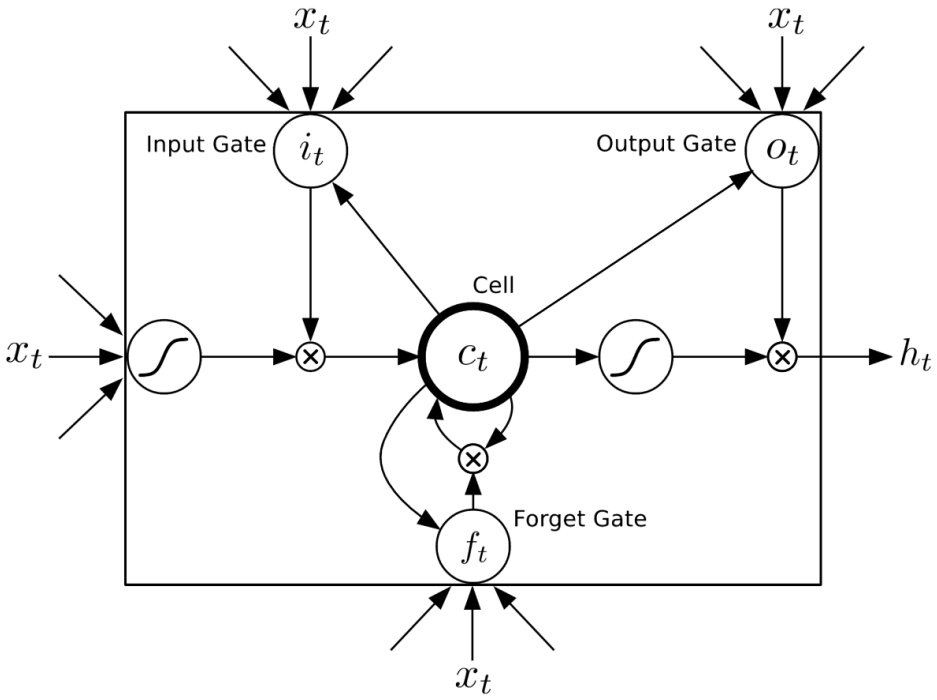


Fig. 3.4 Long short-term memory cell

Variables:

- $x_t \in R^d$: input vector to the LSTM unit
- $f_t \in R^h$: forget gate's activation vector
- $i_t \in R^h$: input gate's activation vector
- $o_t \in R^h$: output gate's activation vector
- $h_t \in R^h$: output vector of the LSTM unit
- $c_t \in R^h$: cell state vector
- $W \in R^{hXd}$, $U \in R^{hXh,}$ and $b \in R^h$: weight matrices and bias vector parameters which need to be learned during training

where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Activation Functions:

- $б_g$: sigmoid function
- $б_c$: hyperbolic tangent function
- $б_h$: hyperbolic tangent function

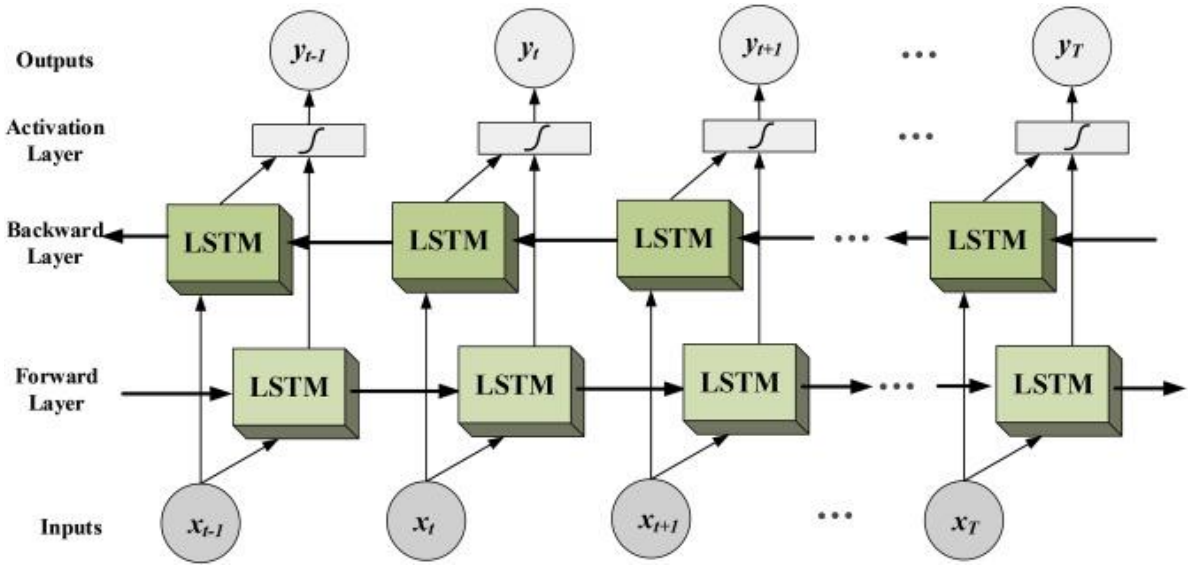


Fig. 3.5 Bidirectional LSTMs and Backpropagation through time

3.3.2 Proposed Network

The training was done by using Keras toolkit [7]. The input layer had 60 input nodes associated with the dimension of the input feature vector, for feature fusion method, and 39, 12 and 9 nodes respectively for MFCC, Sadjadi and New feature sets under the decision fusion methods. The hidden layer of the network was 256 LSTM layer with hyperbolic tangent activation, densely-connected layer with softmax activation function and the dropout value of which was 0.25%. The backpropagation through time (BPTT) algorithm used 10^{-3} learning rate. This was in accordance with the model used in [9]. When tested on 16 GB laptop, this network was showing quite poor results, so it was changed.

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, input_shape=(None, data_dim)))
model.add(Bidirectional(LSTM(256, activation='tanh', dropout=0.25, return_sequences=True)))
model.add(TimeDistributed(Dense(1, activation='softmax')))

RMSprop = optimizers.RMSprop(lr=0.003)
model.compile(loss='binary_crossentropy', optimizer=RMSprop, metrics=['accuracy'])
```

Fig. 3.6 Network parameters suggested in [9]

But this model was not giving satisfactory results. After a few modifications, the following model was made, which gave the best possible results for the dataset containing 20 out of 1500 samples in the 16 GB laptop.

```
model = Sequential()
model.add(LSTM(20, input_shape=(None, data_dim), return_sequences=True))
model.add(Bidirectional(LSTM(20, return_sequences=True)))
model.add(TimeDistributed(Dense(1, activation='sigmoid')))

RMSprop = optimizers.RMSprop(lr=0.003)
model.compile(loss='binary_crossentropy', optimizer=RMSprop, metrics=['accuracy'])
```

Fig. 3.7 Modified network parameter for 16GB machine

Metrics used for this work were ROC curve, the area under the ROC curve (AUC), Equal error rate (EER). A confusion matrix was built which would give the number of true positives, true negatives, false positives and false negatives frames as outputted by the network out of the given number of frames in the testing data. Accuracy and loss given by the model were calculated while training and testing to monitor the performance of

the model. These metrics were selected to compare the performance of the given model with the ones presented in the papers which had been referred to for the work. AUC and EER were used in [2], and the model in this work was supposed to give better results from that in the previous work.

```
# Building Network
print("Building network...")
# expected input data shape: (batch_size, timesteps, data_dim)
model = Sequential()

model.add(Dense(64, input_shape=(None, data_dim)))
model.add((CuDNNLSTM(128, return_sequences=True)))
model.add(Dropout(0.25))
model.add((Dense(1, activation='linear')))

print("Learning rate = ", lr)
model.compile(loss='binary_crossentropy', optimizer=Adam(lr), metrics=['accuracy'])
model.summary()
```

Fig. 3.8 Final LSTM network for 256GB machine

3.4 Experiment

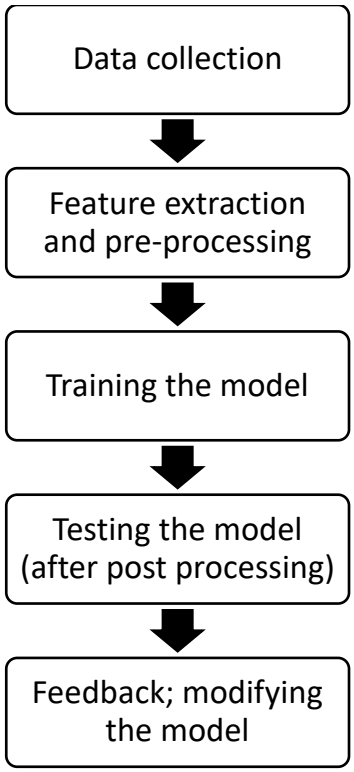


Fig. 3.9 Process of Voice Activity Detection structure

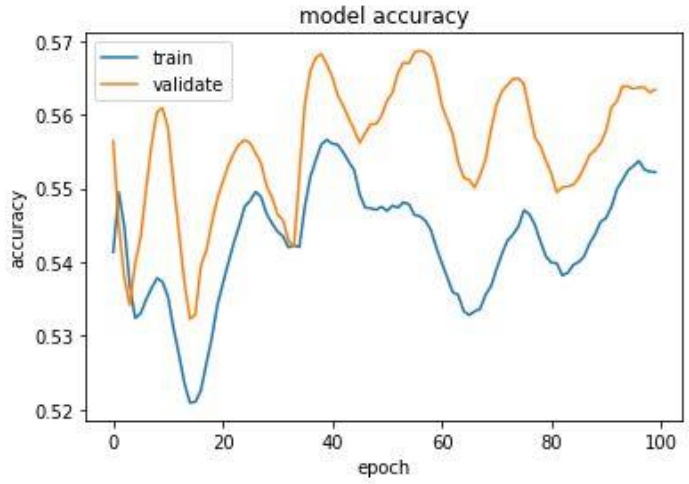
A validation set was made up of 10 audio samples out of the 500 samples which were being used for training. Training was done on those 500 samples, having 6 different types of noises. Even with a hundred epochs, that took almost an hour for a single LSTM layer while more than 2 hours for a network with 2 LSTM layers, a stable training, and validation accuracy percentage was not being achieved even though the loss was decreasing exponentially. Different learning rates, activation functions, batch sizes and no. of layers were tried, but not more than 60% training and validation accuracy were achieved. Training was done over 100 epochs. The trained model was saved as a .h5 file, which could be used for testing after that. In the given result, 16 audio samples from ITU-T testing database was used. The noisy samples were at an SNR level of 20dB, having Mall noise. Training accuracy and loss were measured and plotted per epoch and metrics used were AUC and EER for testing data. An array of predicted labels for testing samples was formed. A Confusion matrix was also made to find the number of true positivies, false positives, true negatives and false negatives out of the lables predicted by the network v/s true labels.

3.5 Result

Though more than 80% accuracy was expected from the network, due to various problems to be discussed in the next section, the final accuracy of the network was significantly less than that.

```
Final Score: [Loss, Accuracy] = [1.1660325527191162, 0.359965980052948]

Predicted labels:  [ 0.18265866  0.18742904  0.24566944 ... -0.02772831  0.49277914
  0.54550844]
```



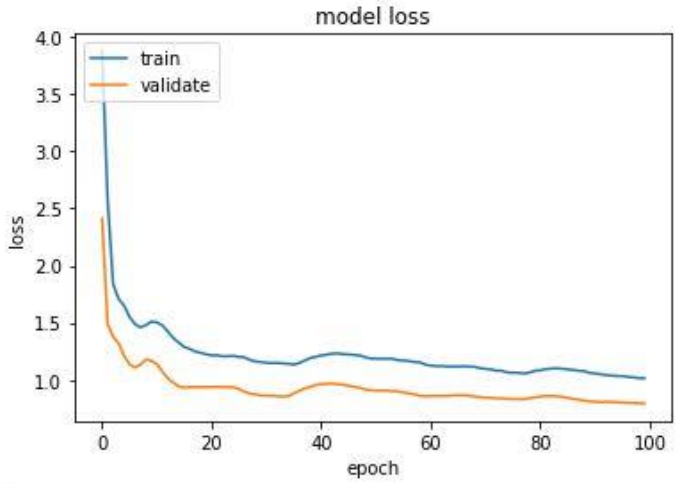
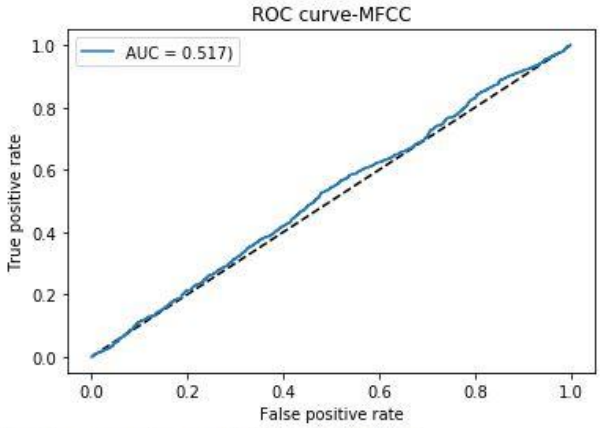


Fig. 4.1 Final results using only MFCC features

```
Confusion Matrix :
 [[2556 1290]
 [7361 4075]]
True Negatives: 2556, False Positives: 1290, False Negatives: 7361, True Positives: 4075
```

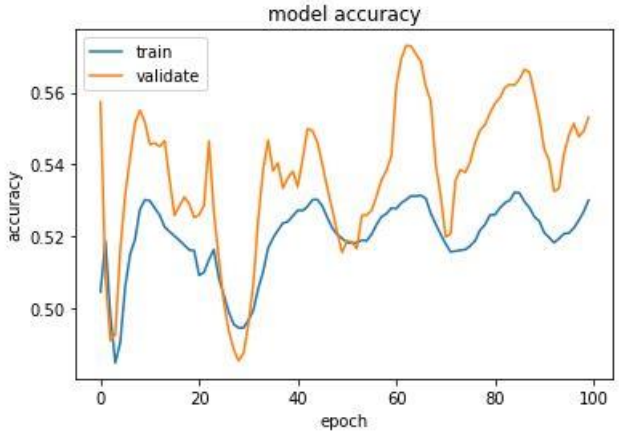



```
Equal error rate =  0.4774084878794193
```

Fig. 4.2 ROC curve using only MFCC features

```
Final Score: [Loss, Accuracy] = [1.2803922891616821, 0.3855516314506531]

Predicted labels:  [0.08431575 0.13458085 0.21816535 ... 0.47849855 0.11189629 0.6149116 ]
```



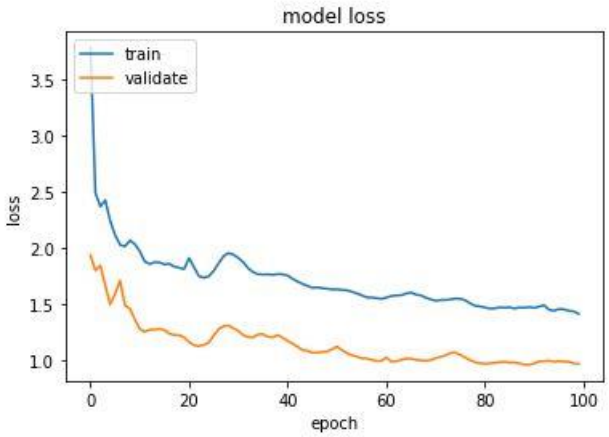
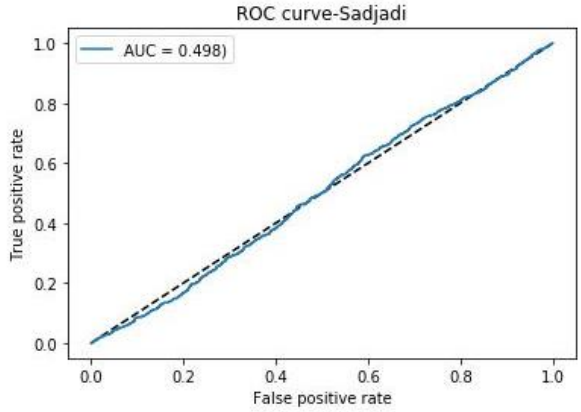


Fig. 4.3 Final results using only Sadjadi features

```
AUC score:  0.49806791764500236

Confusion Matrix :
 [[2203 1643]
 [6628 4808]]
True Negatives: 2203, False Positives: 1643, False Negatives: 6628, True Positives: 4808
```

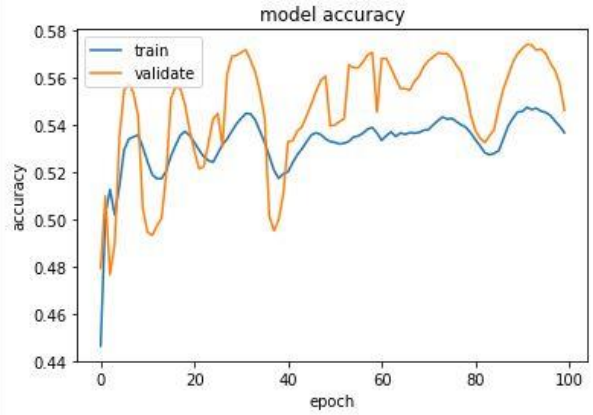



```
ipdb> Equal error rate =  0.49965022735187414
```

Fig. 4.4 ROC curve using only Sadjadi features

```
Final Score: [Loss, Accuracy] = [0.803106427192688, 0.3984426259994507]

Predicted labels:  [0.42279795 0.48991507 0.49254474 ... 0.6318588  0.5734001  0.8058895 ]
```



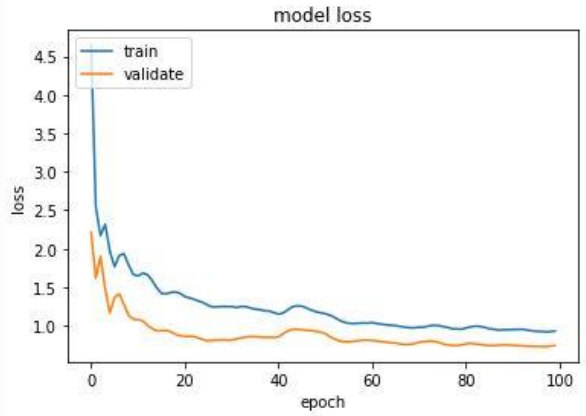
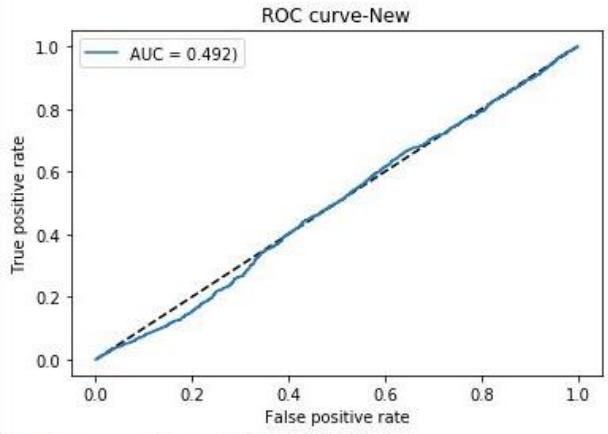


Fig. 4.5 Final results using only New features

```
Confusion Matrix :
 [[1330 2516]
 [3747 7689]]
True Negatives: 1330, False Positives: 2516, False Negatives: 3747, True Positives: 7689
```

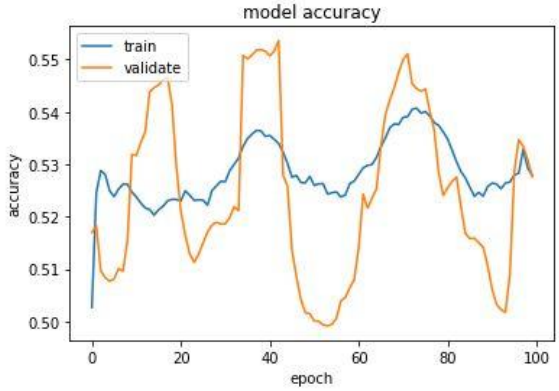



```
Equal error rate =  0.499869127077622
```

Fig. 4.6 ROC curve using only New features

```
Final Score: [Loss, Accuracy] = [1.4388329982757568, 0.41984033584594727]

Predicted labels:  [0.377697   0.33043727 0.30337673 ... 0.00747354 0.12320007 0.37601247]
```



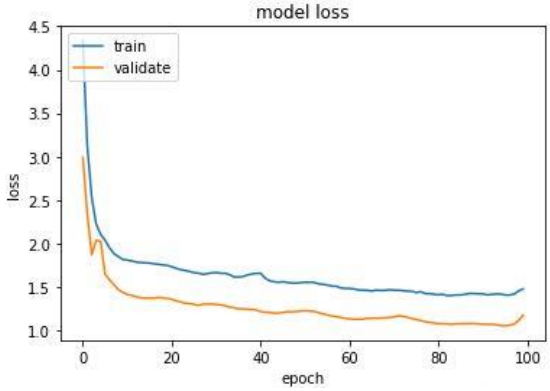
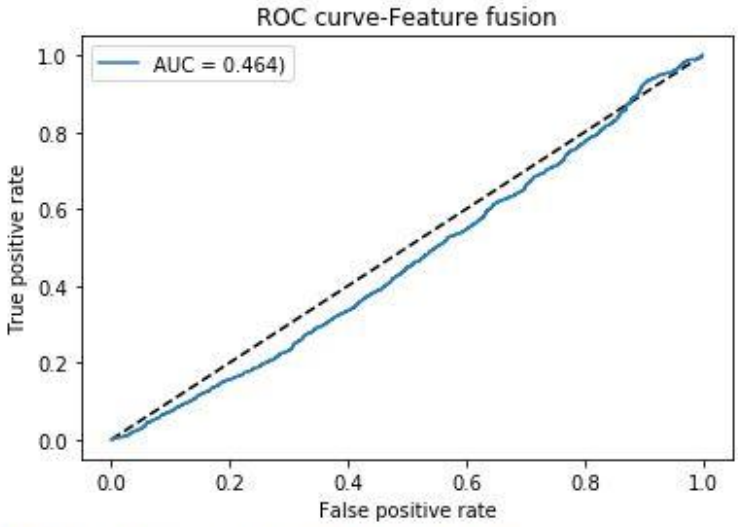


Fig. 4.7 Final results using Feature fusion

```
Confusion Matrix :
 [[1558 2288]
 [5250 6186]]
True Negatives: 1558, False Positives: 2288, False Negatives: 5250, True Positives: 6186
```




```
Equal error rate =  0.5263828238719068
```

Fig. 4.8 ROC curve using Feature fusion

Metrics	MFCC Features	Sadjadi Features	New Features	Feature Fusion
Final Loss	1.660	1.2804	0.8031	1.4388
Final Accuracy	0.3599	0.3855	0.3984	0.4198
AUC score	0.517	0.498	0.492	0.464
Equal Error Rate	0.4774	0.4996	0.4999	0.5264

Fig. 4.9 Table of results of the proposed method

The results are poor as compared to what was expected and even what was done previously on VAD. The reasons for these results have been discussed in the next topic.

4) Conclusion

4.1 Problems Faced

Initially, the model was tested for 20 (out of 1500) audio samples, to check the codes (on a 16GB machine). The accuracy was around 60% when 20 samples were used for training. It seemed that using the full dataset would improve the performance of the network. But when the 256 GB machine (GPU) was used, the BLSTM model was not able to be trained. Even for 20 samples, the machine was using more computational time than the 16GB machine. The reason behind this problem was found out to be that normal LSTMs or BLSTMs are not supported by the GPU and another network, called CuLSTM (compatible with CuDa toolkit – used for training machine learning algorithms on GPU) had to be used. CuLSTMs are not bidirectional, so a bidirectional LSTM model could not be made and thus I was unable to get the expected result. This network, when trained on 1500 samples, crashed. To make use of the GPU on the 256GB machine the Tensorflow version 1.8 compatible with Windows 7 was installed. Correspondingly, an older version of CuDa software (a computing platform needed to develop GPU-accelerated applications, by Nvidia), i.e. CuDa 8.0 had to be installed. When the model in fig. 3.5 was run using 1500 audio samples on this machine on Spyder (Anaconda), it showed a “Resource Exhaustion” error. More than 500 samples were not getting trained by the network.

Various sources on the internet suggested some solutions to this problem like reducing the batch size, but they did not work very well. I tried to use data generator functions for the training of the dataset instead of the inbuilt function in Keras, model.fit() as suggested in some reports for large datasets. But this function took even more time for training than model.fit() function. Also, similar “Resource exhaustion” error showed up when I tried to make the model deeper or increase the number of nodes in the layers. Thus, using more than 128 nodes in 2 CuLSTM nodes were not possible. Some online sources suggested that the problem was due to older versions of Tensorflow and CuDa for the GPU, but newer versions could not be installed as they were not compatible with Windows 7. Moreover, due to the huge time is taken by the code for training, and the corresponding time constraint, a good number of combinations of network parameters like learning rate, activation function, optimizer, dropout, number of nodes, batch size, etc. could not be tried.

4.2 Future Work

1. Training must be done on the full dataset, i.e., 1500 audio samples.
2. The model selected was based on Bidirectional LSTMs, but they could not be used due to GPU issues. It is expected that the use of BLSTMs will give improved results.
3. There was a restriction on the number of layers and the number of nodes in each layer, which should be resolved. More number of LSTM layers have shown to give improved results in many papers.
4. Different combinations of network parameters must be tried on the full dataset with BLSTMs to get optimum accuracy.
5. The current work was done on 100 epochs, but a larger number of epochs can be used for the training to get saturated, but the number should not be so large as to create the problem of overfitting.
6. The above-stated modifications must be done with Tensorflow 1.8 or more with Cuda 9.0 or Cuda 9.2 so that better training can be done on the GPU.
7. Once appreciable accuracy is obtained, further minute improvements can be made by incorporating methods like Noise-aware training (NAT), and models like CG-LSTMs, BLSTM+ (as discussed in the prior sections). Modulation spectrum feature (MS) can also be used.

4.3 Conclusion

Several different approached for Voice Activity Detection were studied and compared, and a technique for the problem was devised. The combination of Drugman feature set with Long Short-term Memory model was a novel approach; there was no work available to compare the results. The proposed approach for Voice Activity Detection, Bidirectional LSTM is a state-of-art technique for VAD as suggested by the papers followed and it was expected to give better results than the approach of Deep Belief Networks using the same database and same set of features used in the previous work. The Drugman feature set has been used with one-layer deep CuLSTM in the work finally. Due to several problems faced during the implementation of the idea, satisfactory results could not be obtained. These problems can be overcome using the ideas presented in the future work (section 4.2).

References

1) https://mycourses.aalto.fi/pluginfile.php/146209/mod_resource/content/1/slides_07_vad.pdf
2) Avani Kulkarni, "Design & Development of Spatial Environment Sensing Algorithm Using Digital Microphones," M.Tech. Project Report, CARE IIT Delhi, 2017
3) Sibo Tong, Hao Gu, and Kai Yu, "A Comparative Study of Robustness of Deep Learning Approaches For VAD," IEEE International Conference on Acoustics, Speech and Signal Processing, 2016
4) Gregory Gelly and Jean-Luc Gauvain , “Optimization of RNN-Based Speech Activity Detection,” IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, MARCH 2018
5) Gregory Gelly and Jean-Luc Gauvain, “MinimumWord Error Training of RNN-based Voice Activity Detection,” Interspeech, ISCA, 2015
6) Phuttapong Sertsi, Surasak Boonkla, Vataya Chunwijitra, Nattapong Kurpukdee, and Chai Wutiwiwatchai, "Robust Voice Activity Detection Based on LSTM Recurrent Neural Networks and Modulation Spectrum," APSIPA Annual Summit and Conference, 2017
7) T. Drugman, Y. Stylianou, Y. Kida, M. Akamine, “Voice Activity Detection: Merging Source and Filter-based Information,” IEEE Signal Processing Letters, 2015
8) Drugman Feature Extraction Toolkit: tcts.fpms.ac.be/~drugman/files/VAD.zip
9) Kera Toolkit: keras.io
10) https://github.com/jefflai108/LSTM
11) http://colah.github.io/posts/2015-08-Understanding-LSTMs/
12) https://en.wikipedia.org/wiki/Long_short-term_memory
13) https://en.wikipedia.org/wiki/Mel-frequency_cepstrum