RESEARCH AND DEVELOPMENT PROJECT
EE691
FINAL REPORT

LANGUAGE IDENTIFICATION USING ACOUSTIC FEATURES

KALPESH PATIL
130040019
DEPARTMENT OF ELECTRICAL ENGINEERING
IIT BOMBAY
EMAIL : kalpeshpatil@iitb.ac.in

GUIDE: PROF. PREETI RAO
DEPARTMENT OF ELECTRICAL ENGINEERING,
IIT BOMBAY
EMAIL : prao@ee.iitb.ac.in

Contents

1 Abstract

Identification of spoken language is an important area of research which lies at the intersection of Speech Processing and Machine Learning. There is enough scope for development in the context of recognition of Indian languages. The following report will summarize some of the major state of the art techniques in this domain along with few experimental findings on the CallFriend dataset [1].

2 Introduction

We will first briefly describe the problem and give an overview of the various steps involved in it. Later we will explain various traditional as well as more recent techniques used for language identification and experimental results obtained using them. Some of the hybrid techniques will also be explained in the end.

There are various models of machine learning which can carry out task of language classification effectively. On a broader level the task can be divided into subtasks mentioned below. Most of the methods in the literature will implement various techniques for one or multiple subtasks.

- **Preprocessing**
  This majorly involves converting raw audio files into the segments of audio which can be used by Feature Extraction module. Channel seperation (for multichannel input), silence removal (Voice Activity Detection), noise removal etc. are some of the common techniques used here.

- **Feature Extraction**
  Unlike images speech data can't be directly fed to the classifier. It needs to be passed through a feature extractor which will convert raw utterances or transcripts (if available) into features which can be interpreted by classifier. This involves following types of features:

  - *Acoustic Features*
    These are purely based on raw utterances of sound. The features like MFCC, PLP, SDC are used. There has been some work on comparing these features for the specific task of Language Identification, but MFCCs are used widely.
  - *Phonotactic*
    These features try to discriminate languages based phones. Hence either a phone level transcription of speech is required or phone recognizer of the the language under consideration is required. Since these are difficult to build, it puts some practical restrictions on the their usage. This involves features like N-grams, C-V etc.

– *Prosodic Features*
These features try to quantify prosodic information in the speech signal using various methods. F_0 contours, stress, intonation are some of the examples of prosodic features.

We will focus on methods which involve acoustic features only, because they need the least amount of data and their individual performance is better than others. Note that performance can be enhanced by using combination of multiple types of feature. [2]

- **Feature Accumulation**
This is the most important part of a Language Identification system. All the approaches mainly differ in this part. The basic idea is to combine acoustic features across frames into a single representation of the entire utterance. Some of the approaches here will be described more in detail later.

- **Decision Layer**
Decision layer is simply a classifier which will predict output based on the accumulated features. Neural Networks, SVM etc. have been used in the literature extensively. When feature accumulator is absent and prediction is made for each frame and later the decisions are clubbed using majority voting, highest likelihood etc in the decision layer.

3 Dataset Description

We are working with two datasets, CallFriend dataset for Hindi and Tamil languages and Mandi dataset for classification of dialects in Marathi language.

- **CallFriend Dataset**
The datasets consist of roughly half an hour long conversation on telephone calls between two parties. There are 40 such conversations for Hindi language and 35 for Tamil language. Although actual length of speech after silence removal may vary.

- **Mandi Dataset**
This dataset contains speech samples of 6 sec for each speakers belonging to 34 districts in Maharashtra. Each speaker has 10 utterances while each district has roughly 15 such speakers. The task is to find insights about the dialects entirely based on acoustic features from this dataset.

Since Mandi dataset is highly noisy and distinction of dialect based on geographical regions isn't clearly available, some of the preliminary experiments didn't perform well

Language	Train	Test
Hindi	3388	813
Tamil	2356	767
Total	5744	1580

Table 1: Number of segments of duration 10 sec generated from conversations in CallFriend dataset

on Mandi dataset. Hence the numbers reported later in the report are for CallFriend dataset. We carried out some of the experiment on Mandi dataset as well, which are mentioned separately in the end.

Data Preprocessing

Below are some of the steps needed to create appropriate data for the task.

- *Channel Separation*
  Each conversation has speech inputs coming from two channels. The channels were separated and each channel was considered as independent speech stream of that language.

- *Silence Removal*
  Simple energy based thresholding technique was used for silence removal. -20dB threshold was used to characterize silence and silence regions longer than 0.5 sec were truncated to 0.5 sec. This process roughly reduced length of each sample from 30 minutes to 10 minutes.

- *Train-test splitting*
  Roughly 80% of the conversation files streams were used for training while 20% for testing. All speech streams were divided into chunks of 10 sec each. This 10 sec segment will be unit of evaluation now onwards. Note that we are designing 'Speaker Independent' Language Recognition system i.e. there are no common speakers in train and test data. Therefore the system is expected to capture language specific insights rather than speaker specific. Table 1 describes statistics of these chunks.

Cluster size	200	300	400	500
Accuracy (%)	75.57	76.77	77.08	77.91

Table 2: Performance of VQ based approach with different number of clusters

4 Feature Extraction

We used *MFCC* features as suggested in the literature. First raw speech is passed through a pre-emphasis filter. Sliding window of length 25ms with shift of 10ms was used to generate frames. Filterbank of 26 filters placed between 200Hz and 4000Hz was used to extract coefficients. Then we choose top 13 coefficients as features and append their delta and delta-delta coefficients to the original feature vector. *CMVN* normalization was carried out on each feature to remove biases due to local environment conditions. Thus we get 39 dimensional feature vector of for each frame in the utterance.

5 Vector Quantization approach

We tried to implement method based on Vector Quantization suggested in [3]. This method involves generation of a codebook of certain size from training data for each language. The codebook is generated by clustering feature vectors of all frames of all training speech samples for a given language. *MiniBatchKmeans* algorithm [4] was used due to its computational effectiveness against normal K-Means algorithm for large number of data points. Once the codebook of centroids is generated, we compute euclidean distance of each frame of a test utterance from the centroids. The frame is labeled by the language corresponding to the centroid which minimizes this distance. Finally output is labeled by taking majority voting of frames. Table 2 shows results of this technique by varying size of the codebook. As we had expected theoretically, we do obtain improvement in performance as number of clusters increases. As long as we have sufficient data for finding optimal centroids for given number of clusters, we would find improvement on increasing number of clusters. But this assumption might fail if we are operating in the regime of 'overfitting' i.e. large number of cluster centroids to be optimize using small number of parameters.

6 GPPS based approach

Classical approaches based on *GMM-UBM* setting have been extensively studied in literature. Gaussian Posterior Probability Supervector *(GPPS)* [5] has been shown to work

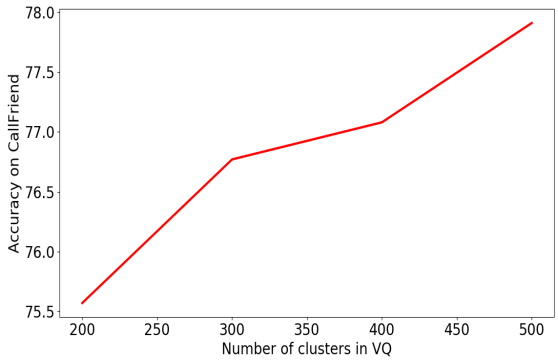

Figure 1: Performance of VQ based approach with different number of clusters

for accent recognition task. We tried to implement it for Language Recognition task. The approach can be summarized as follows:

6.1 GMM-UBM training

From the MFCC feature vectors a Universal Background Model *(UBM)* is trained. It is assumed to be mixture of Gaussian distributions *(GMM)*. Number of distributions used for training an UBM is an important parameter here. We have studied results by varying number of distribution components of GMM. Using the standard procedure (EM algorithm) we fit a GMM model on training data and estimate weights, means and covariances of mixture components. Since we have limited amount of data, large number of parameters can't be estimated. Hence we assume diagonal covariance matrices for mixture components. Note that UBM can also be trained from a large dataset separately (prior) and modified slightly according to our training dataset using MAP. This is known as MAP adaptation.

6.2 GPPS extraction

GPPS vector for an utterance can be computed as follows.

$$Pr(o_t|\lambda) = \sum_{j=1}^{J} w_j Pr(o_t|\mu_j, \Sigma_j)$$
$$\lambda = \{w_j, \mu_j, \Sigma_j\}, j = 1, 2..J$$

Here o_t is acoustic feature vector at time t (39 dimensional MFCC in our case). w_j, μ_j, Σ_j are estimated parameters of j^{th} mixture component. $Pr(o_t|\mu_j, \Sigma_j)$ is the probability

value according to Gaussian distribution. After computing these values for each frame, we proceed to compute GPPS vector (feature accumulation step).

$$\kappa_j = \frac{1}{T}\sum_{t=1}^{T}\frac{w_j Pr(o_t|\mu_j,\Sigma_j)}{\sum_{j=1}^{J} w_j Pr(o_t|\mu_j,\Sigma_j)}$$

$$\kappa = [\kappa_1, \kappa_2, ...\kappa_J]$$

Here T is total number of frames in an utterance and κ is GPPS vector of that utterance. Thus we extract J dimensional GPPS vector for train and test data. This feature vector is fed to top layer classifier.

6.3 Classifier

Our aim is to train a classifier which takes J dimensional GPPS vector as input and predicts the language. We can use any off-the-shelf classifier for this purpose. *SVM* and *Neural Network* are the most famous. We tried with different architectures of Neural Network for the classification task. Following architecture seems to produce good results.

> *InputLayer(J), Dense(100,relu), Dropout(0.5), Dense(10,relu), Dropout(0.5), Dense(2)*

We have reported the results for varying J (number of mixture components) in Table 3 with the same architecture. As we had expected theoretically, we do obtain improvement in performance as number of GMM component increases.

GMM components	64	128	256	512
Accuracy (%)	83.35	84.81	88.10	90.06

Table 3: Performance of GPPS-NN approach with different number of mixture components

We also analyzed performance of SVM classifier for this particular Hindi vs Tamil task. Table 4 describes the performance of SVM classifier and optimal parameters found for the classification task. We used 'Radial Basis Function (RBF)' kernel for SVM. Optimal parameters were found by varying 'C' from 10^{-3} to 10^{5} in logspace and 'γ' also in the same range in logspace. It is interesting to note that as number of components increases, optimal value of C decreases while optimal value of γ increases. We also find increase in accuracy similar to the NN case as number of components increases.

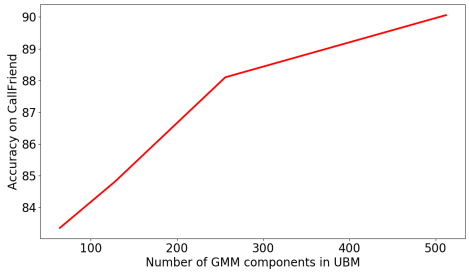

Figure 2: GPPS-NN

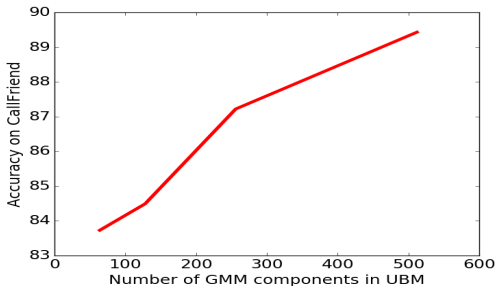

Figure 3: GPPS-SVM

Figure 4: Performance of GPPS approach with different number of mixture components

GMM components	64	128	256	512
Accuracy (%)	83.73	84.49	87.22	89.43
Optimal C	100000	10000	1000	100
Optimal γ	0.001	0.1	1	10

Table 4: Performance and optimal parameters of GPPS-SVM (rbf kernel) approach with different number of mixture components

7 DNN based approaches

Deep Neural Networks have been extensively used in literature for classification tasks with highly non-linear class boundaries. We studied mainly two approaches in this section; with and without context.

7.1 Context-free DNN

All the frames across all utterances in training data are gathered together and tagged with respective language id. This forms training set for the neural network. The network takes input as a 39 dimensional MFCC vector and output as its language id. We train this network on available training data with usual *cross entropy loss*. The architecture of the network is described below,

> *InputLayer*(39), *Dense(100,relu)*, *Dropout(0.5)*, *Dense(10,relu)*, *Dropout(0.5)*, *Dense(2)*

Once this network is trained, we use it to predict language id of all frames in the test utterance. The test utterance is assigned the language which has been predicted for maximum number of frames *(majority rule)*. Apart from majority rule we have also explored *maximum likelihood*, where we take sum of log probabilities of each language

across frames. Sum of log probabilities corresponds to product of probabilities which is similar to maximum likelihood. Let $Pr(y_n = i|x_n)$ be the probability that n^{th} frame belongs to language i given input x_n. Therefore

$$\hat{L} = \underset{i}{argmax} \prod_{n=1}^{N} Pr(y_n = i|x_n)$$
$$\therefore \hat{L} = \underset{i}{argmax} \sum_{n=1}^{N} log(Pr(y_n = i|x_n))$$

$\hat{L}$ is the language predicted for the given test utterance. One might argue that $\{y_n\}$ are not independent, hence probability can't be written as product of individual terms.

It is observed that in general this total likelihood approach performs better than majority rule. We tried various architectures for the network and realize that network architecture affects performance of such system. The maximum accuracy obtained using this approach was 65.21 which is quite less compared to other approaches. The major drawback of this approach is that it lacks in accounting for temporal patterns across frames. It treats each frame as independent and neglects all the temporal connections. This problem is addressed in the next subsection.

7.2 Context DNN

This is similar to previous approach in terms of method of classification, just there is change in input to the classifier. The approach is described in [6]. Here we append few frames before and after to the original frame and give this as an input. Let N_c be number of context frames. Therefore input will be $39(2N_c + 1)$ dimensional. In [6] authors have used $N_c = 10$. Note that as N_c changes network architecture changes too. Hence Due to computational inefficiency we would demonstrate results obtained with $N_c = 5$. Also note that we didn't use out-of-set languages as described in the original paper. The final decision making component was same as described in previous section (majority rule or maximum total likelihood). The exact architecture used is described below,

> *InputLayer(*429*), Dense(1000,relu), Dropout(0.5), Dense(200,relu), Dropout(0.5), Dense(50,relu), Dropout(0.5),Dense(2)*

As expected context DNN performed significantly better than context-free DNN. Here also we observe that maximum total likelihood performs better than majority voting rule. The maximum accuracy obtained using Context DNN is 80.12. Note that Context DNN and context free DNN cannot be compared to each other directly because network architecture used for both of them was significantly different due to change in sizes of input. We have tried our best to find the optimal architectures for both the approaches. Anyways the results are summarised in Table 5.

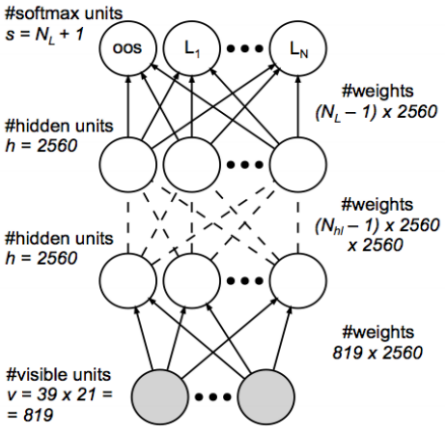

Figure 5: Context DNN network topology. Image credits: [6]

	Majority vote	Maximum total likelihood
Context-free DNN	64.34	65.21
Context DNN	78.57	80.12

Table 5: Performance of DNN based approaches

8 Bottleneck Features

8.1 What are bottleneck feature?

Traditional features like MFCC or PLP are kind of hand-crafted features. They are extracted from the speech utterances using a fixed method without giving much thought to the end goal. It has been a key problem to find an effective representation of the speech utterances which contain enough discriminative information about languages. Language information is assumed to be weak and present latently in speech utterances, therefore traditional features may not bring out the latent information. Hence features specific to language recognition task are needed to be extracted. Bottleneck features are quite famous in ASR (automatic speech recognition) systems especially in Tandem based ASR systems.

Deep bottleneck features are obtained from the bottleneck layer of a network trained in a generative paradigm. There are various techniques such as *Restricted Boltzman Machines (RBM), Autoencoders, Stacked Autoencoders* which can be used here. We have used autoencoders for the extraction of bottleneck features. The basic idea behind autoencoders is to reconstruct given input at the output layer by passing it through the

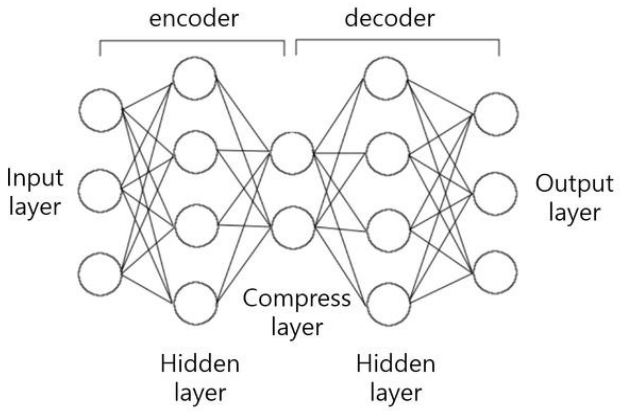

Figure 6: Autoencoder concept diagram

network. Usually dimensions of the bottleneck layers are quite less compared to that of input-output layers and network architecture is symmetric across the bottleneck layer. Fig. 6 illustrates the concept of autoencoders. Part of the network from input to the bottleneck layer is called *encoder*, while from bottleneck to the output layer is called *decoder*. The training is done to minimize the reconstruction error with some optional regularization. Once network is trained, we cut out the decoder part. All the inputs are passed through the encoder and respective bottleneck features are extracted from the bottleneck layer.

8.2 BNF and fine tuning

We create input similar to the DNN section ($39(2N_c+1)$ dimensional) by appending MFCC features of N_c context frames. The output is also same. Then we train autoencoder by minimizing *mean squared loss* as described in previous section. After training the network, decoder part is cut out and a top level softmax classifier layer is added in front of bottleneck feature layers. Then we *'fine-tune'* the network by using small learning rate (smaller than the one used in autoencoder training). These kind of methods are also known as *'pretraining'* in Deep Learning domain. The philosophy behind pretraining is that rather than initializing a classifier network with random weights, we initialize it with the weights learned by autoencoder. Autoencoder pretraining puts the network in a position in the weights-space from where it is easy to learn optimal weights for classification. The learning rate is reduced to ensure that weights remain within the vicinity of the value to which they were initialized. The results obtained using this method are summarized below in Table 6. Note that architecture used here for autoencoder is similar to the one used in DNN based approach.

*InputLayer(*429*), Dense(1000,relu), Dropout(0.5), Dense(200,relu), Dropout(0.5),*

	Majority vote	Maximum total likelihood
Context DNN	80.27	82.46

Table 6: Performance of pretrained DNN

*Dense(50,relu)(also the bottleneck layer), Dense(200,relu), Dropout(0.5), Dense(1000,relu),OutputLayer(*429*)*

8.3 BNF and GPPS

So far in all the DNN based approaches, the unit of training was a single frame or multiple context frames. In GPPS approach mentioned earlier in section 6 unit of training was the GPPS vector which was obtained from combining posteriors of all frames together. Here also we repeat the same thing. But instead of using MFCC features, we used bottleneck features obtained form training the autoencoder.

Similar to the previous section an autoencoder is trained using context MFCC context frames. The extracted bottleneck frames are put into the GPPS pipeline described earlier in 6 instead of MFCC. Entire GPPS procedure is repeated (GMM-UBM training, GPPS extraction, final classifier). This approach has given the best results so far. With the autoencoder architecture given above and 512 Gaussian distributions, the accuracy was found to be **92.39%**. The major reason behind the success of this approach can be explained by bringing out best of both the worlds, classifiers and features. We have observed that GPPS has produced the best results as a classifier compared to all other classifiers discussed so far. We have also observed that use of bottleneck features gives performance improvement over simple MFCC features. Thus combining both of them produced better results compared to individual improvements.

9 Other Approaches

Apart from the approaches mentioned above, there are few more approaches which were studied during the course of the project. Some of the notable ones are described below. These approaches aren't fully explored from the implementation point of view.

9.1 RNN-LSTM based approaches

There has been several attempts to tackle problem of Language Identification using approaches based on *sequence classifier* deep neural networks such as LSTM or RNN [7]. The basic idea is to feed a sequence of MFCC features to the LSTM or RNN network and train it to predict their output labels. RNNs are the networks unrolling in time. RNN

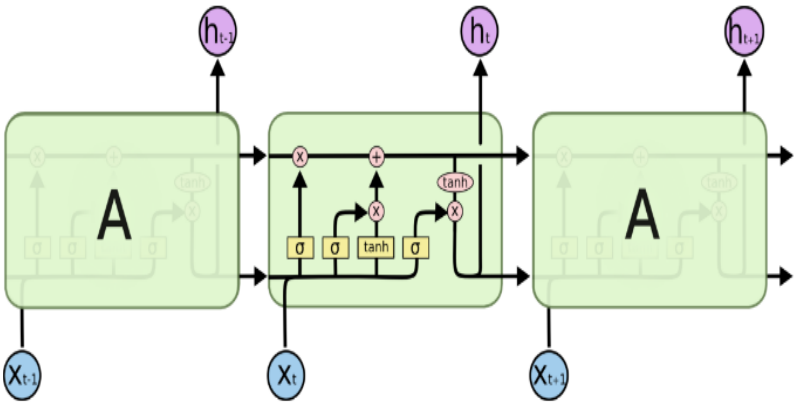

Figure 7: LSTM concept diagram. Image credits: [8]

predicts current state of the network from current input (MFCC feature frame in our case) and previous state in time. They suffer from problem of vanishing gradients due to long dependencies in time. Hence an advanced sequence classifier network like LSTM is used. It consists of a *memory cells* which kind of acts like a conveyor belt running in time carrying the important information. The *input gates* and *forget gates* are used to add or remove information from the cell state. In our case the cell state would contain information relevant for the language identification. Multiple layers of such LSTM networks can be built to generate a deeper architecture. Please refer [8] for intuitive understanding of LSTM. We have analyzed results of LSTM based approach on our dataset. Even after trying various architectures for LSTM, we observe that performance is not up to the mark (**accuracy 70%**). The primary reasons could be lack of enough training data or choice of an inappropriate network architecture. We wish to analyze this issue in future.

9.2 *i-vector*

i-vector [9] based methods are also quite famous in the literature. The basic idea behind using i-vector is to reduce dimensionality of a long supervector of an utterance by using matrix factorization techniques.

$$M = m + Tw$$

Here M is GMM supervector created by stacking up all the GMM mean vectors, m is UBM supervector, T is called *Total Variability matrix*. *i-vector* training involves training of T matrix to obtain a low dimensional representation in the form of w. These vectors can further be used with any vanilla classifier (SVM, NN) etc.

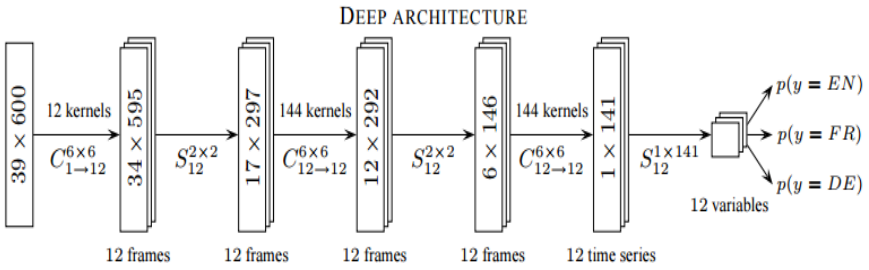

Figure 8: CNN concept diagram. Image credits: [10]

9.3 CNN

CNN are quite popular for classification of images. The major benefit of CNN is *local connectivity* and *weight sharing* which allows us to look upon the neural network as set of filters sliding over input. CNNs for the task of language identification are studied in [10]. The input is 600 frames of 39 dimensional MFCC features (which sort of acts like a 600×39 dimensional image). The architecture used in the paper can be seen in 8. Rest of the things are similar to the normal CNN used for images. One might argue that 2D-CNN isn't a good choice for speech data, because there isn't much local connectivity along MFCC feature dimension of the input. There is indeed local connection across time dimension and hence 1D-CNN could be a better choice. Therefore we decided to implement 1D-CNN. The obtained accuracy is **78.32%** which is also quite less compared to GPPS and VQ. But still it is a good point to start thinking along the lines of 1D-CNN for language classification.

10 Preliminary Experiment on Mandi Dataset

As described earlier, we had done some preliminary experiment on the Mandi dataset. We implemented GPPS method with 512 clusters on a subset of Mandi dataset. We formulated a binary classification problem of accent recognition of Marathi language by collecting speech data of *coastal districts (Konkan region)* and *eastern districts (Vidarbha region)* of Maharashtra. Here also we repeat similar 'speaker independent' train test splitting of utterances and then divide them into smaller chunks of duration 6 sec each. The details of number of chunks in train and test split are mentioned in **??**. On passing this data through the GPPS method with 512 clusters, we obtain accuracy of **66.57%**, which is quite less. The primary reason could be that the data was collected by asking people to read out certain sentences (different for each speaker), which might have suppressed their natural accent. Also the data contain large amount of background noise, which might have affected performance significantly.

Accent	Train	Test
Coastal	567	170
Eastern	568	180
Total	1135	350

Table 7: Number of samples generated from Mandi dataset

11 Conclusion and Future Work

In this project we studied the language identification problem from various angles and decided to focus on using acoustic features. We studied and implemented various techniques ranging from traditional ones like Vector Quantization, GMM-UBM based techniques to the relatively newer ones based on deep learning and bottleneck features and different variants of this techniques. We briefly touched upon techniques like 1D-CNN and LSTM which require more attention in future. We also studied few of the hybrid approaches like BNF+GPPS which involved combination of both kinds and found out that it performs better than others. We understand that most of the deep learning based approaches lack clear insights into the model and work like blackbox, hence it becomes difficult to tune them; but at the same time they are effective in terms of representation of data (BNF against MFCC).

There are few more experimental techniques which can be tried on CallFriend dataset. Having tested various techniques on the standard dataset like CallFriend, we would like to extend them to the Mandi dataset. There are techniques like *Transfer Learning*, which can be used to exploit a well trained models on large datasets (like CallFriend) to be able to use on smaller dataset like (Mandi) by fine-tuning only the last few layers. We wish to study them in future.

12 Acknowledgments

I would like to thank **Prof. Preeti Rao** for providing me this opportunity and express my gratitude to her for the constant support.
I would also like to thank **Digital Audio Processing lab**, Department of Electrical Engineering, IIT Bombay for providing GPU support to conduct experiments.

References

[1] "Callfriend dataset," *https://catalog.ldc.upenn.edu/byproject*.

[2] V. R. Reddy, S. Maity, and K. S. Rao, "Identification of indian languages using multi-level spectral and prosodic features," *International Journal of Speech Technology*, vol. 16, no. 4, pp. 489–511, 2013.

[3] J. Balleda, H. A. Murthy, and T. Nagarajan, "Language identification from short segments of speech.," in *INTERSPEECH*, pp. 1033–1036, 2000.

[4] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, ACM, 2010.

[5] M. H. Bahari, R. Saeidi, D. Van Leeuwen, *et al.*, "Accent recognition using i-vector, gaussian mean supervector and gaussian posterior probability supervector for spontaneous telephone speech," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 7344–7348, IEEE, 2013.

[6] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 5337–5341, IEEE, 2014.

[7] R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Language identification in short utterances using long short-term memory (lstm) recurrent neural networks," *PloS one*, vol. 11, no. 1, p. e0146917, 2016.

[8] "Understanding lstm networks," *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*.

[9] N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction.," in *Interspeech*, pp. 857–860, 2011.

[10] G. Montavon, "Deep learning for spoken language identification," in *NIPS Workshop on deep learning for speech recognition and related applications*, pp. 1–4, 2009.