B.E. PROJECT ON

Spoken Language Identification using Neural Network

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS OF AWARD OF
B.E. (COMPUTER ENGINEERING)
DEGREE OF UNIVERSITY OF DELHI

SUBMITTED BY:

Aditya Jain 210/CO/13
Anmol Pandey 233/CO/13
Anmol Varshney 234/CO/13

GUIDED BY:

Dr. Shampa Chakraverty

COMPUTER ENGINEERING (COE)

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI

2016-17

नेताजी सुभाष प्रौद्योगिकी संस्थान
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi-110 045
Telephone : 2509 9036-42, 2509 9050 Fax : 2509 9022 Website : http://www.nsit.ac.in

CERTIFICATE

The project titled "**Spoken Language Identification using Neural Networks**" by **Aditya Jain (210/CO/13), Anmol Pandey (233/CO/13) and Anmol Varshney (234/CO/13)** is a record of bona fide work carried out by us, in the Division of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi, under the supervision and guidance of **Dr. Shampa Chakraverty** in partial fulfilment of requirement for the award of the degree of Bachelor of Engineering in Computer Engineering, University of Delhi in the academic year 2016 - 2017.

Dr. Shampa Chakraverty

Prof. and Head of Department
Computer Engineering Department
Netaji Subhas Institute of Technology
New Delhi

Date: _____

Candidates' Declaration

This is to certify that the work which is being hereby presented by us in this project titled "Spoken Language Identification using Neural Networks" in partial fulfilment of the award of the Bachelor of Engineering submitted at the Department of Computer Engineering, Netaji Subhas Institute of Technology Delhi, is a genuine account of our work carried out during the period from January 2017 to May 2017 under the guidance of Dr. Shampa Chakraverty, Head of Department(COE), Netaji Subhas Institute of Technology, Delhi. The matter embodied in the project report to the best of our knowledge has not been submitted for the award of any other degree elsewhere.

Aditya Jain Anmol Pandey Anmol Varshney

Date: _____

This is to certify that the above declaration by the students is true to the best of my knowledge.

Dr. Shampa Chakraverty

Date: _____

Acknowledgement

Any Project indisputably plays one of the most important roles in an engineering student's life to make him a successful engineer. It provides the students with an opportunity to gain valuable experience on the practical application of their technical knowledge and also brings out and hones their technical creativity. Thus the need for one is indispensable.

We would like to express our deep gratitude towards our mentor **Dr. Shampa Chakraverty,** Head of Department, Computer Engineering Department, Netaji Subhas Institute of Technology, New Delhi under whose supervision we completed our work. Her invaluable suggestions, enlightening comments and constructive criticism always kept our spirits up during our work. She was always there to help whenever we faced any problems.

Our experience in working together has been wonderful. We hope that the knowledge, practical and theoretical, that we have gained through this term B.E. Project will help us in our future endeavors in the field.

We are grateful to all our friends for providing critical feedback and support whenever required.

We regret any inadvertent omissions.

Aditya Jain (210/CO/13)

Anmol Pandey (233/CO/13)

Anmol Varshney (234/CO/13)

Abstract

In this study we attempt to identify the spoken language of audio samples of speech using Deep Neural Networks (DNNs). Language Processing has been a subject of wide interest and a lot of work has been done towards that end. The first step in a language processing pipeline is always language identification (LID). Language identification facilitates transferring the control to an appropriate next stage of the processing system. In this dissertation, we adapt DNNs to the problem of language identification using the mel-frequency cepstral coefficients of audio signals as primary features. The audio signals are divided into several frames and the short term acoustic features extracted from each of these frames. Mean and Variance of these short-term acoustic features across several frames are calculated to capture the fluctuations of speech in order obtain a reasonable accuracy. These average features are then stacked together taking measures from both before and after the frame under consideration forming context windows to further capture the time dependent behavior of the signal. These windows result in a superior performance than many of the existing systems. These features are feeded into a Neural Network which is used to identify the top two prospective candidates for the given speech signal. Binary classification is then applied among these candidates to determine the final output. At both of these stages feature selection is applied to include only the most promising features and reduce the dimensionality of the inputs. Feature Selection for the binary classification stage results in a matrix, which contains the selected features specifically aimed to differentiate between any two specified languages. Our findings suggest that DNNs can be used for LID tasks with reasonable accuracy. Although our model is tested using only 3 languages it can easily be extended to any number of languages.

Keywords: Language Identification, Deep Neural Networks, mel-frequency coefficients, Context Window, Feature Selection

List of Contents

List of Figures

Chapter 1: Introduction

The problem of automatic language identification (LID) can be defined as the process of automatically identifying the language of a given spoken utterance. The first step in a language processing pipeline is always LID. LID is used either as a standalone task or as a preprocessing step, capturing the first seconds of the recording and processing it in order to transfer the control to the appropriate next stage; e.g. speech recognition systems, multilingual translation systems or call-centers[1] (e.g., emergency calls) routing, where the response time of a native operator might be critical. LID is also a topic of great importance in areas of intelligence and security, where the language identities of recorded messages and archived materials need to be established before any information can be extracted.

Our project aims at spoken language identification in speech audio samples using Deep Learning Models (DNNs). Speech audio recognition features well-defined, clear voices and shows very little background noise. This is in contrast to song audio which combines background music, instruments and the singer's voice into a complex mix. Even though several high level approaches based on phonotactic and prosody are used as meaningful complementary sources of information, nowadays, many state-of-the-art LID systems still include or rely on acoustic modelling. While previous works on neural networks applied to LID report results using shallow architectures or convolutional neural networks, in this study, we propose the use of DNNs as a new method to perform LID at the acoustic level. Recent breakthroughs in signal processing and acoustic modelling have shown the promise of deep learning models.

Motivated by those results and also by the discriminative nature of DNNs, we adapt DNNs to work at the acoustic frame level to perform LID. Particularly, in this work, we extract the acoustic features from an audio signal, mfcc and their first and second order differentials to be exact which has been divided into frames and compute their mean and variance across several frames in order to account for the fluctuation in the speech signal. This results in stacking of average features across several consecutive frames known as context windows which also result in much superior performance. The mel-frequency

cepstrum is a representation of an audio signal on the mel scale, a nonlinear mapping of frequencies that down-samples higher frequencies to imitate the human ear's ability to process sound. Delta features and the Delta Delta Features are the first and second time derivatives of the cepstral coefficients, capturing the change of the cepstral features over time. Feature Selection using Chi Square Statistics is also applied to obtain the best possible results as well as reduce dimensionality of the input. We compare the obtained results with a baseline Support Vector Machine (SVM) based system trained from exactly the same acoustic features.

The remaining thesis is organized as follows. In chapter 2, previous work done and the current state of the art in this area are detailed, along with the limitations and the roadblocks encountered in this field. Chapter 3 provides and overview of the technologies used and the theoretical concepts behind using the frameworks or tools. Chapter 4 discusses about our contribution to this field of research. A detailed experimental framework and the course of action is elaborated. Chapter 5 provides an experimental evaluation of our proposed approach, and the findings of the experiment are reported and analyzed. Chapter 6 summarizes the thesis with conclusions and avenues of future work.

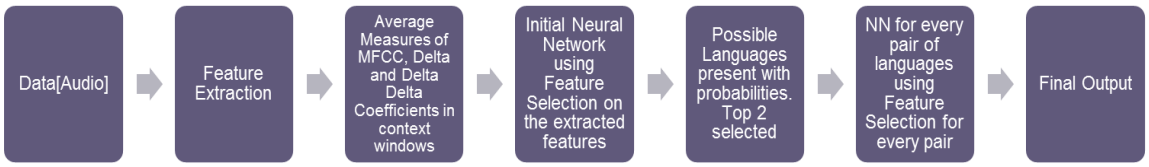

Fig **1** Workflow of the project

Chapter 2: Literature Survey

2.1 Phonotactic Language Identification Systems

Lincoln Laboratory has investigated the development of a system that can automatically identify the language of a speech utterance. To perform the task of automatic language identification, they have experimented with four approaches:[2]

1. Gaussian mixture model classification
2. Single-language phone recognition followed by language modelling (PRLM).
3. Parallel PRLM, which uses multiple single-language phone recognizers, each trained in a different language.
4. Language-dependent parallel phone recognition.

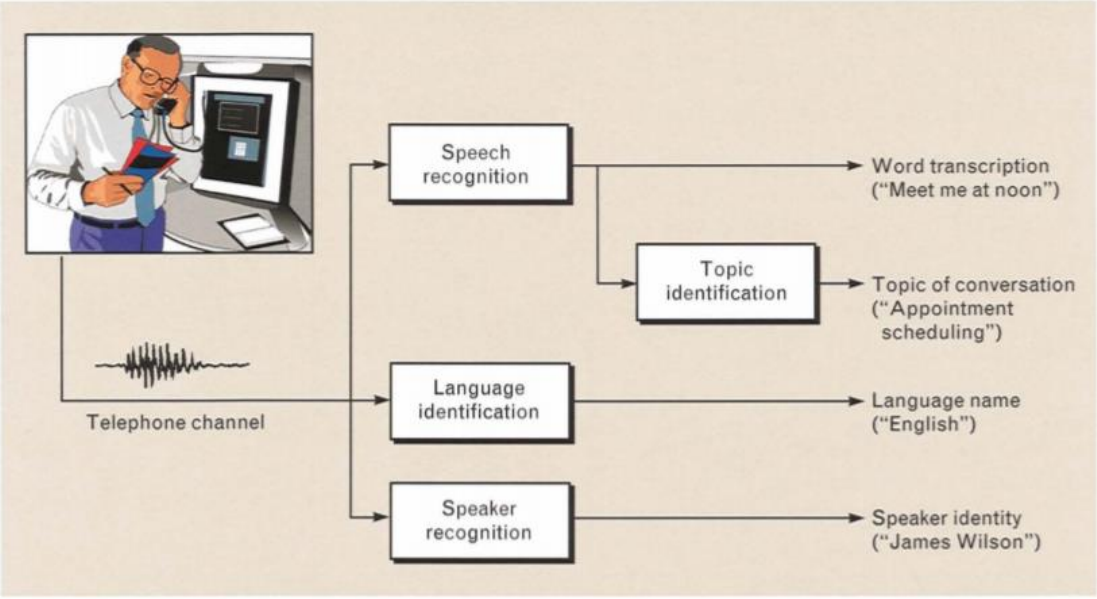

Fig **2** Phonotactic Language Identification Systems

2.1.1 GMM Based

A GMM language-ID system[3-5] served as the simplest algorithm for this study. As shown below, GMM language ID is motivated by the observation that different languages have different sounds and sound frequencies. Under the GMM assumption, each feature vector V t at frame time t is assumed to be drawn randomly according to a probability density that is a weighted sum of unimodal multivariate Gaussian densities. Following is a brief introduction to GMM.

- GMMs are used to represent frame-based speech features
- Used for estimating acoustic likelihoods
- GMMs are a weighted sum of multivariate Gaussians

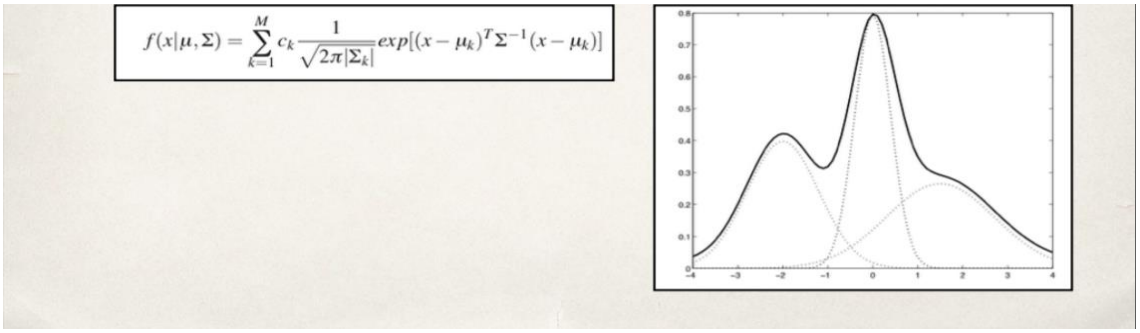

Fig **3** GMM Model

The basic aim of using Gaussian Mixture Model is to minimize the log likelihood function

Consider log likelihood

$$\ln p(X \mid \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln p(x_n) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right\}$$

ML does not work here as there is no closed form solution. Parameters can be calculated using Expectation Maximization(EM) technique.

Steps involved are:

1. From Bayes rule

$$\gamma_k(x) = \mathrm{p}(k|x) = \frac{\mathrm{p}(k)p(x|k)}{\mathrm{p}(x)} = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

$$where, \qquad \pi_k = \frac{N_k}{N}$$

2. **E step**. Evaluate the responsibilities using the current parameter values[7-8]

$$\gamma_j(x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

3. **M step**. Re-estimate the parameters using current responsibilities

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \qquad \Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)}$$

$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n)$$

4. Evaluate log likelihood

$$\ln p(X \mid \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\}$$

   If there is no convergence, return to step 2.

2.1.2 Phone Recognition Followed by Language Modelling (PRLM)[12]

The second language-ID approach they tested comprises a single-language phone recognizer followed by language modelling with an n-gram analyzer, as shown in Fig **{number}.** In the PRLM system, training messages in each language l are tokenized by a single-language phone recognizer, the resulting symbol sequence associated with each of the training messages is analyzed, and an n-gram probability-distribution language model is estimated for each language l. During recognition, a test message is tokenized and the likelihood that its symbol sequence was produced in each of the languages is calculated.

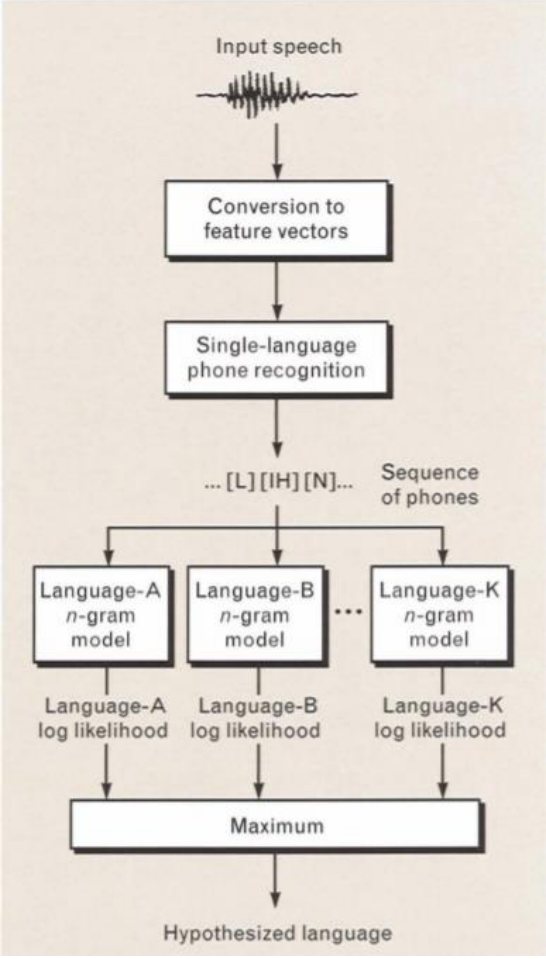

Fig **4** Phone Recognition and Language Modelling (PRLM)

2.1.3 Parallel PRLM

Although PRLM is an effective means of identifying the language of speech messages, we know that the sounds in the languages to be identified do not always occur in the one language that is used to train the front-end phone recognizer. Thus we look for a way to incorporate phones from more than one language into a PRLM-like system. Alternatively, the approach is simply to run multiple PRLM systems in parallel with the single-language front-end recognizers each trained in a different language. This approach requires that labelled training speech be available in more than one language, although the training speech does not need to be available for all, or even any, of the languages to be recognized. **Figure {number}** shows an example of such a parallel PRLM system.

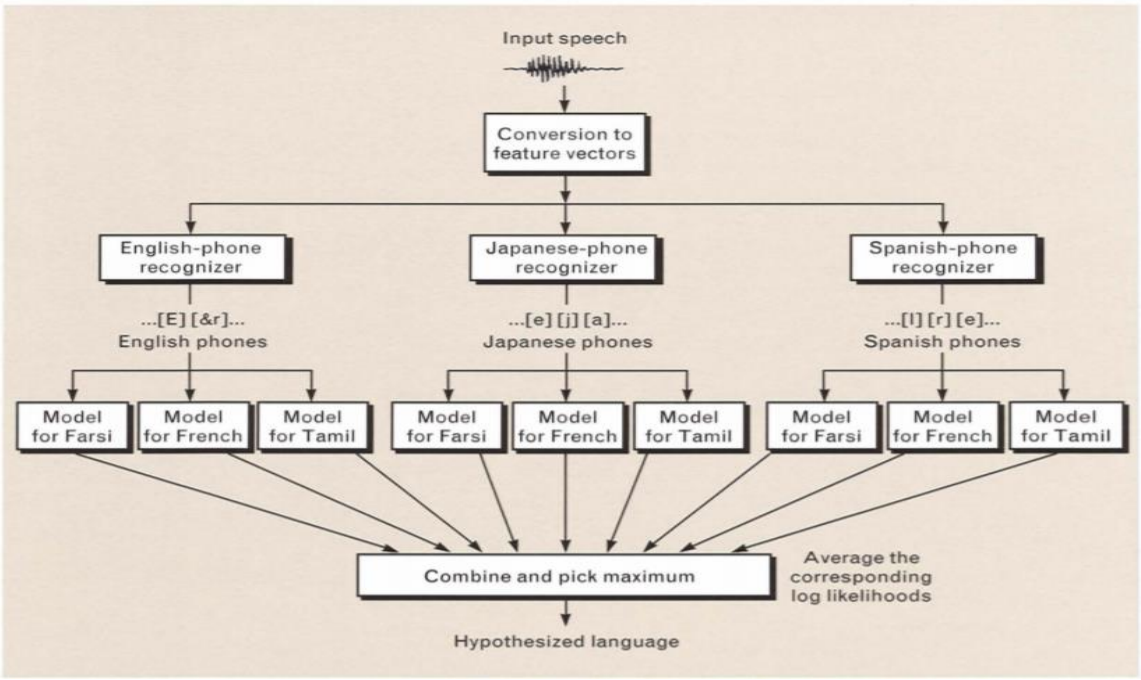

Figure **5** Parallel PRLM

2.1.4 Parallel Phone Recognition

The PRLM and parallel PRLM systems perform phonetic tokenization followed by phonotactic analysis. Though this approach is reasonable when labelled training speech is not available in each language to be identified, the availability of such labelled training speech broadens the scope of possible language-ID strategies; for example, it becomes easy to train and use integrated acoustic phonotactic models. If we allow the phone recognizer to use the language-specific phonotactic constraints during the Viterbi-decoding process rather than applying those constraints after phone recognition is complete (as is done in PRLM and parallel PRLM), the most likely phone sequence identified during recognition will be optimal with respect to some combination of both the acoustics and phonotactics. The joint acoustic-phonotactic likelihood of that phone sequence would seem to be well suited for language ID.

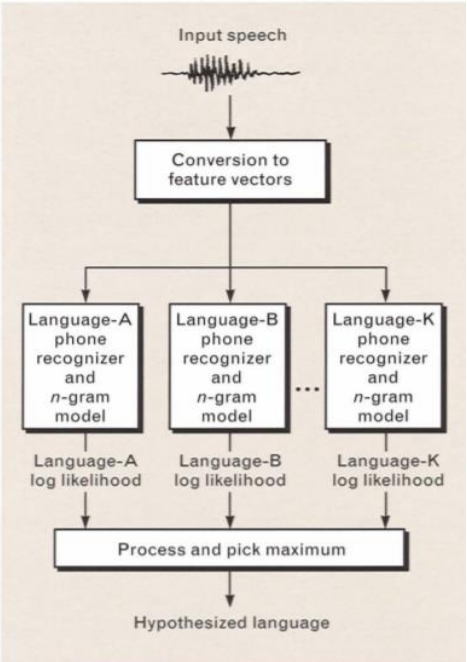

Fig **6** Parallel Phone Recognition

2.2 Language Identification based on GMM and Ceptral Features

The approaches discussed in the previous section were phonotactic in nature. These approaches require tokenization of speech in separate phones which is a computationally complex task and difficult to implement.

The ceptral features on the other hand are easy to compute and give a good representation of physical shape of vocal chords of the speaker. The variation of these coefficients can be used as a good measure for detecting the language.

2.2.1 Kshirod Sarmah and Utpal Bhattacharjee[13]

Title - GMM based Language Identification using MFCC and SDC Features

In this paper, a baseline system for the LID system in multilingual environments has been developed using GMM as a classifier and MFCC combined with Shifted-Delta-Cepstral (SDC) as front end processing feature vectors. In this works, we used the Arunachali Language Speech Database (ALS-DB), a multilingual and multichannel speech corpus which was recently collected from the four local languages namely Adi, Apatani, Galo and Nyishi in Arunachal Pradesh including Hindi and English as secondary languages.The Gaussian mixture model with 1024 Gaussian components has been used for constructing language models. The individual language models were trained using the algorithm Expectation Maximization (EM) of 10 iterative steps. Training for the language model with equal number of male (50) and female (50) speaker's data with the same language. For the any one language suppose Adi language, the language model was created from 100 speaker's utterance of Adi language. Similar approach is also applied for other five languages models Apatani, Galo, Nishi, Hindi and English.

2.2.2 Pedro A. Torres-Carrasquillo[14]

Title - Language Identification using Gaussian Mixture Model Tokenization

Phone tokenization followed by n-gram language modeling has consistently provided good results for the task of language identification. In this paper, this technique is generalized by using Gaussian mixture models as the basis for tokenizing. Performance results are presented for a system employing a GMM tokenizer in conjunction with multiple language processing and score combination techniques. On the 1996 CallFriend LID evaluation set, a 12-way closed set error rate of 17% was obtained.

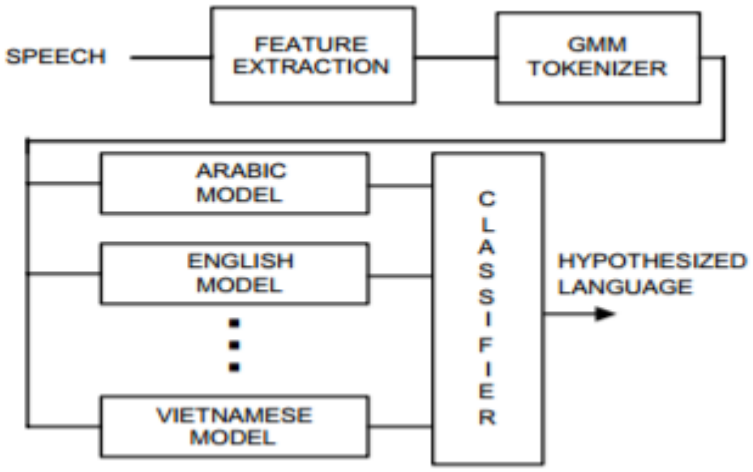

Fig **7** Gaussian Mixture Model Tokenization

2.3 Language Identification based on DNN and Ceptral Features

2.3.1 Ignacio Lopez-Moreno[11]

Title: Automatic Language Identification Using Deep Neural Networks

This work studies the use of deep neural networks (DNNs) to address automatic language identification (LID). Motivated by their recent success in acoustic modelling, we adapt DNNs to the problem of identifying the language of a given spoken utterance from short-term acoustic features. The proposed approach is compared to state-of-the-art i-vector based acoustic systems on two different datasets: Google 5M LID corpus and NIST LRE 2009. Results show how LID can largely benefit from using DNNs, especially when a large amount of training data is available. We found relative improvements up to 70%, in Cavg, over the baseline system.

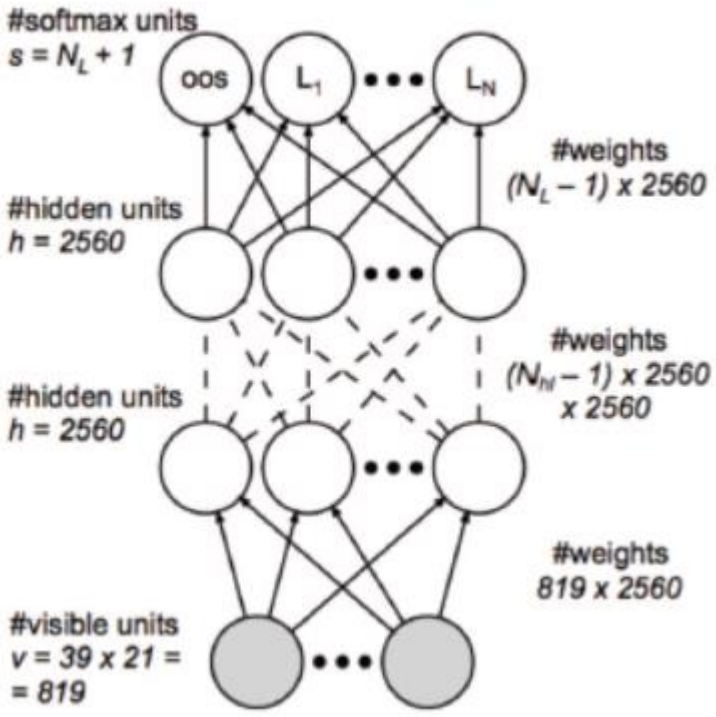

Fig **8** DNN Network Topology

2.3.2 Julien De Mori[15]

Title: Spoken Language Classification

This paper presented various approaches to language detection and they were compared. These methods were Support Vector Machine(SVM) based, Gaussian Mixture Model(GMM) based and Neural network.

However, we were keen in its neural network implementation as it was quite simple and different from other implementations. Instead of computing features for each 25ms frame and then making predictions on the frame level, they modelled each clip as a multivariate Gaussian distribution by computing the mean and variance of each feature across each clip, changing the dimension of the feature vector to 78. This has the effect of averaging out any noise in the signal, as well as reducing the number of training examples, considerably decreasing computation time. The minimum ERR rates for the features MFCC and SDC individually are 19.70% and 11.83% respectively.

2.3.3 Najim Dehak[3]

Title: Language Recognition via I-vectors and Dimensionality Reduction

In this paper, a new language identification system is presented based on the total variability approach previously developed in the field of speaker identification. Various techniques are employed to extract the most salient features in the lower dimensional i-vector space and the system developed results in excellent performance on the 2009 LRE evaluation set without the need for any post-processing or backend techniques. Additional performance gains are observed when the system is combined with other acoustic systems.

The total variability space or i-vector approach concept was first introduced in the context of speaker verification. The basic idea of the total variability space consists of adapting the Universal Background Model (UBM) (which is trained on all the available language data for this paper) to a set of given speech frames based on the eigenvoice adaptation technique in order to estimate the utterance dependent GMM. The eigenvoice adaptation technique operates on the assumption that all the pertinent variability is captured by a low rank

rectangular matrix T named the Total variability matrix. The GMM supervector (vector created by stacking all mean vectors from the GMM) for a given utterance can be modeled as

$$M = m + T\omega + \in$$

where m is the Universal Background Model supervector, the i-vector w is a random vector having a normal distribution N (0, I), and the residual noise term $\sim$ N (0, Σ) models the variability not captured by the matrix T. In our new modeling, we apply an SVM directly to the low dimensional i-vector (which is the coordinate of the speech segment in the total variability space) instead of applying the SVM in the GMM supervector space as done in. The process of training the total variability matrix T is a little bit different compared to learning the eigenvoice adaptation matrix. In eigenvoice training for speaker recognition, all the recordings of a given speaker are considered to belong to the same person; in the case of the total variability matrix however, we pretend that every utterance from a given speaker is produced by different speakers. If we follow the same total variability matrix training process for language identification, we assume that every utterance for a given language class is considered a different class.

It further uses dimensionality reduction by using a popular technique of Linear Discriminant Analysis and Neighbouring Component Analysis. The maximum error obtained was 18.3% on 3 second uttarances.

Chapter 3: Materials and Methods

Machine Learning

Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. Machine Learning is a scientific discipline that addresses the following question: 'How can we program systems to automatically learn and to improve with experience?' Learning in this context is not learning by heart but recognizing complex patterns and make intelligent decisions based on data. The difficulty lies in the fact that the set of all possible decisions given all possible inputs is too complex to describe. To tackle this problem, the field of Machine Learning develops algorithms that discover knowledge from specific data and experience, based on sound statistical and computational principles.

The field of Machine Learning integrates many distinct approaches such as probability theory, logic, combinatorial optimization, search, statistics, reinforcement learning and control theory. The developed methods are at the basis of many applications, ranging from vision to language processing, forecasting, pattern recognition, games, data mining, expert systems and robotics.

A learner can take advantage of examples (data) to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new case. We have followed the same principle, where we have designed algorithms to generalize from the given examples and then produce a useful output. Since the amount of data available for the different languages was limited, we used KFold Cross

Validation which makes efficient use of data available and avoids overfitting, which occurs when a model begins to "memorize" training data rather than "learning" to generalize from trend. Machine learning algorithms are described as either 'supervised' or 'unsupervised'.

Supervised Learning

Supervised learning[17] is the machine learning task of inferring a function from supervised (labeled) training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. It is called a classifier (if the output is discrete) or a regression function (if the output is continuous). Supervised learning is when the data you feed your algorithm is "tagged" to help your logic make decisions.

Our classifying approach is a supervised learning method that uses labelled dataset of audio speech samples, using the features extracted from the frames as the basis of the classifier. Using some of the audio speech samples as the training dataset, classifier is applied on testing dataset to predict the spoken language. The testing method used is KFold Cross Validation which makes efficient use of the available data and avoids overfitting.

pyAudioAnalysis[21]

pyAudioAnalysis is an open Python library that provides a wide range of audio-related functionalities focusing on feature extraction, classification, segmentation and visualization issues. The purpose of the pyAudioAnalysis library is to provide a wide range of audio analysis functionalities through an easy-to-use and comprehensive programming design.

pyAudioAnalysis implements the following functionalities:

- Feature extraction: several audio features both from the time and frequency domain are implemented in the library.
- Classification: supervised knowledge (i.e. annotated recordings) is used to train classifiers. A cross-validation procedure is also implemented in order to estimate the optimal classifier parameter. The output of this functionality is a classifier model which can be stored in a file.
- Regression: models that map audio features to real-valued variables can also be trained in a supervised context. Again, cross validation is implemented to estimate the best parameters of the regression models.
- Segmentation: the following supervised or unsupervised segmentation tasks are implemented in the library: fix-sized segmentation and classification, silence removal, speaker diarization and audio thumbnailing.
- Visualization: given a collection of audio recordings pyAudioAnalysis can be used to extract visualizations of content relationships between these recordings.

One of the best things of using pyAudioAnalysis was the ease of its use. It was very easy to extract the time and frequency domain features from an audio signal. However, since the time domain and the frequency domain features apart from the mfcc features did not provide any improvement in the results, the library was used only to extract the mfcc features. Inspite of providing very accurate classifiers it did not contain any DNN Classifiers, which was the reason it wasn't used for classifying spoken audio samples. A drawback in this library is the large processing time it takes to extract all the features from an audio. It extracts all the features regardless of how many maybe required. It is advised

to tweak the code for the library to extract only the required features. It was used both in the test and training phases to extract features from the audio speech samples.

LibROSA[22]

LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It is built to ease the transition of music information retrieval (MIR) researchers into Python (and modern software development practices), and also to make core MIR techniques readily available to the broader community of scientists and Python programmers. It has a relatively flat package layout, and following scipy rely upon numpy data types and functions, rather than abstract class hierarchies. Librosa's functions expose all relevant parameters to the caller. While this provides a great deal of flexibility and a consistent interface to process audio files by defining a set of general conventions and standardized default parameter values shared across many functions. LibROSA supports the following basic features among others:

- Compute mel spectrogram, MFCC, delta features, chroma features
- Locate beat events
- Compute beat-synchronous features
- Display features
- Save beat tracker output

We used LibROSA to compute the delta and delta delta coefficients from the obtained mfcc coefficients. The best thing about using librosa was the ease of its use. Both delta and delta delta functions could be calculated in a straightforward manner. Althouh librosa does provide functionality to extract mfcc coefficients from an audio signal, it lacks the versatility that pyAudioAnalysis provides and hence wasn't used for feature extraction.

Chi Square Feature Selection

Feature selection is the machine learning task of selecting subset of features which are more relevant for use in model construction. It is applied because:

- Data contains many features that are redundant or irrelevant.
- Less resource (computational time and memory) is required.
- Shorter training as well as prediction time.
- Avoids curse of dimensionality.
- Prevents overfitting of data and generalizes the model.

There are many ways to implement selection of features. One way is by scoring the features in their usefulness. When we have set of categorical data, we can apply chi-squared testing (χ^2). It is a measure of goodness of fit and this test allows us to compare a collection of categorical data with some theoretical expected distribution. Chi square test can be thought of as a test of independence and tests if null hypothesis is true or not with null hypothesis being two categorical variables are independent. It is evaluated using the formula:

$$\chi^2 = \sum_{i=1}^{n} \frac{(\mathrm{O}_i - \mathrm{E}_i)^2}{\mathrm{E}_i}$$

where

χ^2 = chi – squared stat of null hypothesis

O_i = observed value of feature for class i

E_i = expected value of feature for class i

n = number of classes

The chi-squared value can be directly mapped to its corresponding p-value by subtracting cumulative distribution function from 1. Low p-value indicates greater statistical significance. A p-value of 0.05 is often used as a cut off between significant and non-significant results.

Deep Learning

Deep learning (also known as **deep structured learning**, **hierarchical learning** or **deep machine learning**) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations. Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. It is one of the fastest growing fields of Machine Learning - it's true bleeding edge.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures which remove redundancy in representation. Deep learning is nothing but a neural network with a lot of hidden layers of nonlinear processing and supervised or unsupervised learning of feature representations in each layer, such that the layers form a hierarchy from low level to high level features. Deep learning is called 'deep' learning, as its algorithms transform their inputs through more layers than shallow learning algorithms. At each layer, the signal is transformed by a processing unit, like an artificial neuron, whose parameters are learned through training.

We have used DNNs because they have the ability to learn complicated feature representations and classifiers jointly. They learn much better models of data that lie on or near a non-linear manifold and their performance does not saturate with increase in training data.

Theano[18]

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- Tight integration with NumPy.
- Transparent use of a GPU
- Efficient symbolic differentiation
- Speed and stability optimizations dynamic C code generation
- Extensive unit-testing and self-verification

Theano defines a language to represent mathematical expressions and manipulate them, a compiler to create functions that can compute values for these expressions, and a library which will execute these functions when evaluated on numeric values.

The Keras software used in our project builds on the strengths of Theano, by providing a higher level user interface. Keras makes it easier to express the architecture of deep learning models, and training algorithms, as mathematical expressions to be evaluated by Theano.

Keras[19]

Keras is an open source neural network library written in Python. It is capable of running on top of Deeplearning4j, Tensorflow or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible. *Being able to go from idea to result with the least possible delay is key to doing good research.* It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

Features of Keras:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

We used Keras to design and implement the DNNs at both the initial stage as well as the binary classification stage. The main advantage of using Keras was how easy it was to ignore the difficult mathematical details of the underlying neural network and focus only on optimizing the performance of the net. Keras provided a smooth interface to change every parameter of the neural network including the number of layers, type of connections, activation functions, error function, number of neurons in each layer, weight initialization, dropout, regularization and performance metrics among others. It even has an option of loading and saving a model which enables fast learning. Keras is strongly advised for anyone who wishes to analyse the performance of DNN on a problem in a short amount of time.

scikit-learn[20]

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance.

scikit-learn provides the following functionalities:

- Classification: Identifying to which category an object belongs to.
  Algorithms: SVM, nearest neighbors, random forest

- Regression: Predicting a continuous-valued attribute associated with an object.
  Algorithms: SVR, ridge regression, Lasso
- Clustering: Automatic grouping of similar objects into sets.
  Algorithms: k-Means, spectral clustering, mean-shift
- Dimensionality reduction: Reducing the number of random variables to consider.
  Algorithms: PCA, feature selection, non-negative matrix factorization
- Model selection: Comparing, validating and choosing parameters and models.
  Modules: grid search, cross validation, metrics
- Preprocessing: Feature extraction and normalization.
  Modules: preprocessing, feature extraction.

We used scikit-learn to build the SGD Classifier which was used as a baseline for comparison with our proposed Model. The library was used to perform KFold Cross Validation which makes efficient use of the limited data available and avoids overfitting. It was also used to preprocess the inputs before feeding it to the neural network to convert the output labels into one hot vectors and to shuffle the input before training.

pyAudio (Real time analysis)

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS. PyAudio is inspired by:

- pyPortAudio/fastaudio: Python bindings for PortAudio **v18** API.
- tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.

It can be used in two modes:

- Blocking mode audio I/O
- Callback mode audio I/O

We have used it for doing real time analysis of audio. For this we need callback mode I/O. In callback mode python program's main thread listens to audio being input from specified source and stores the audio in its buffer. When buffer is filled to a specified capacity, PyAudio will call a specified callback function with the audio data in it's buffer. This process creates a new thread on which we input the data to our prediction model and plot the result resulting in real time processing of data.

PyQt[23]

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British Firm Riverbank Computing. PyQt is available in two editions: PyQt4 which will build against Qt 4.x and 5.x and PyQt5 which will only build against 5.x. Both editions can be built for Python 2 and 3. PyQt supports Microsoft Windows as well as various flavours of Unix, including Linux and macOS.

PyQt implements around 440 classes and over 6,000 functions and methods including:

- Substantial set of GUI widgets
- Classes for accessing SQL databases (ODBC, MySQL, PostgreSQL, Oracle, SQLite)
- QScintilla, Scintilla-based rich text editor widget
- Data aware widgets that are automatically populated from a database
- XML parser
- SVG support
- Classes for embedding ActiveX controls on Windows
- To automatically generate these bindings, Phil Thompson developed the tool SIP, which is also used in other projects.

The main advantage of using PyQt is the strong object oriented behavior which makes using different modules in the program very easy. We used PyQt to develop the GUI which

provides a visual tool for our model's working. Object oriented behavior led to easy integration of a graph window with the main window containing the option for selecting a file. PyQt is a cross platform GUI/XML/SQL C++ framework which makes it very efficient as well as makes it possible to run on any platform available.

NumPy[24]

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy address the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring (re)writing some code, mostly inner loops using NumPy.

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- Powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

The main attraction of using NumPy is the fast and efficient processing of the NumPy arrays as compared to native Python lists. Performance in terms of processing time would have been much worse if NumPy wasn't used. NumPy was used extensively in the project ranging from the extraction of features to the plotting of features against a suitable measure for visual representation. Features were extracted as NumPy arrays. They were processed

as NumPy arrays using fast mathematical functions provided by NumPy. NumPy array functions were used to compute mean and variance of the features extracted. Both the baseline and proposed models accepted NumPy arrays as their input. And finally the results were obtained using NumPy arrays and the matplotlib graphs used NumPy array values as the values for plotting.

Matplotlib[25]

matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. Itis a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

The main advantage of matplotlib is the ease of usability and vast functions available for plotting. We used matplotlib for the visual representation of real time spoken language identification.

It displays the predicted language at each second interval using the model built and running the prediction in a multiple threads. The predicted language was shown using a graph. Matplotlib provided easy methods to plot the graph that varies with time.

Chapter 4: Experimental Framework

The research carried out is a four phase project. The first phase involves extraction of the features from the speech audio samples. This also involves making context windows out of the frames of the audio speech sample and calculating the mean and variance of several consecutive context frames to further capture the fluctuations in the signal. The features are also normalized to ensure every sample contributes equally. The second phase involves feature selection among the extracted features for collecting the best contributing features and inputting these features to the Initial Neural network. The Initial Neural network is then trained with the selected features. The third phase involves applying feature selection among all possible pairs of languages under consideration so as to extract the best contributing features for every pair. These features are then inputted to the Binary Neural Network and the Binary Neural Network is trained. The fourth and the final phase involves testing the samples against the Hybrid Neural Network. The third and the fourth phase are intertwined in the sense that the training and testing data are split according to K Fold Cross Validation which makes efficient use of data. The samples are divided for training and testing purposes. Features are extracted from the test samples. This also involves making context windows out of the frames and calculating the mean and variance of several consecutive context frames. The features are then normalized and the same features during the training/second phase are selected and then inputted to the Initial Neural Network. This network generates the top two best candidates for the given sample. These candidates are provided to the appropriate Binary Neural Network which selects the same features during the second training/third phase. These features are focused on classification between the two given languages. These features are then inputted to the Binary Neural Network which provides the final output.

4.1 Feature Extraction

Dataset Description

The data set used for the purpose of this project was extracted from https://www.audio-lingua.eu/[26]. A number of different sources were tried but the most promising results were obtained by using the data from the above mentioned website. The site contains large number of 16kHz frequency, mono audio speech samples for both male and female voices in a variety of languages. The clips do not have a fixed length and vary in their duration greatly. We created a script which automatically opened up the browser and loaded a specified page. It then copied the link to download each .mp3 provided on the site till a certain specified number. These links were collected into a file and were used to download the files into appropriate folders. These .mp3 files were converted into .wav format which was suitable for feature extraction. We scraped the recordings of 3 languages: Chinese, French and German. A total of 200 samples were downloaded for each language to be used for both training and testing purposes.

Extraction of Features

We experimented with using a number of features for classification but the most promising results were obtained by using the mfcc features along with delta mfcc and delta delta mfcc features. This resulted in 39 features for each frame. The mel-frequency cepstrum is a representation of an audio signal on the mel scale, a nonlinear mapping of frequencies that down-samples higher frequencies to imitate the human ear's ability to process sound. In our implementations, we used the first 13 cepstral coefficients as our primary features, as is common in similar applications. Delta features and the Delta Delta Features are the first and second time derivatives of the cepstral coefficients, capturing the change of the cepstral features over time, which are useful in classifying language, since pace is an important factor in language recognition by humans. We calculated these features as the central finite difference approximation of these derivatives. The audio was divided into frames and the

features were extracted using 400 frames in a single window with 100 frames overlap between the windows.

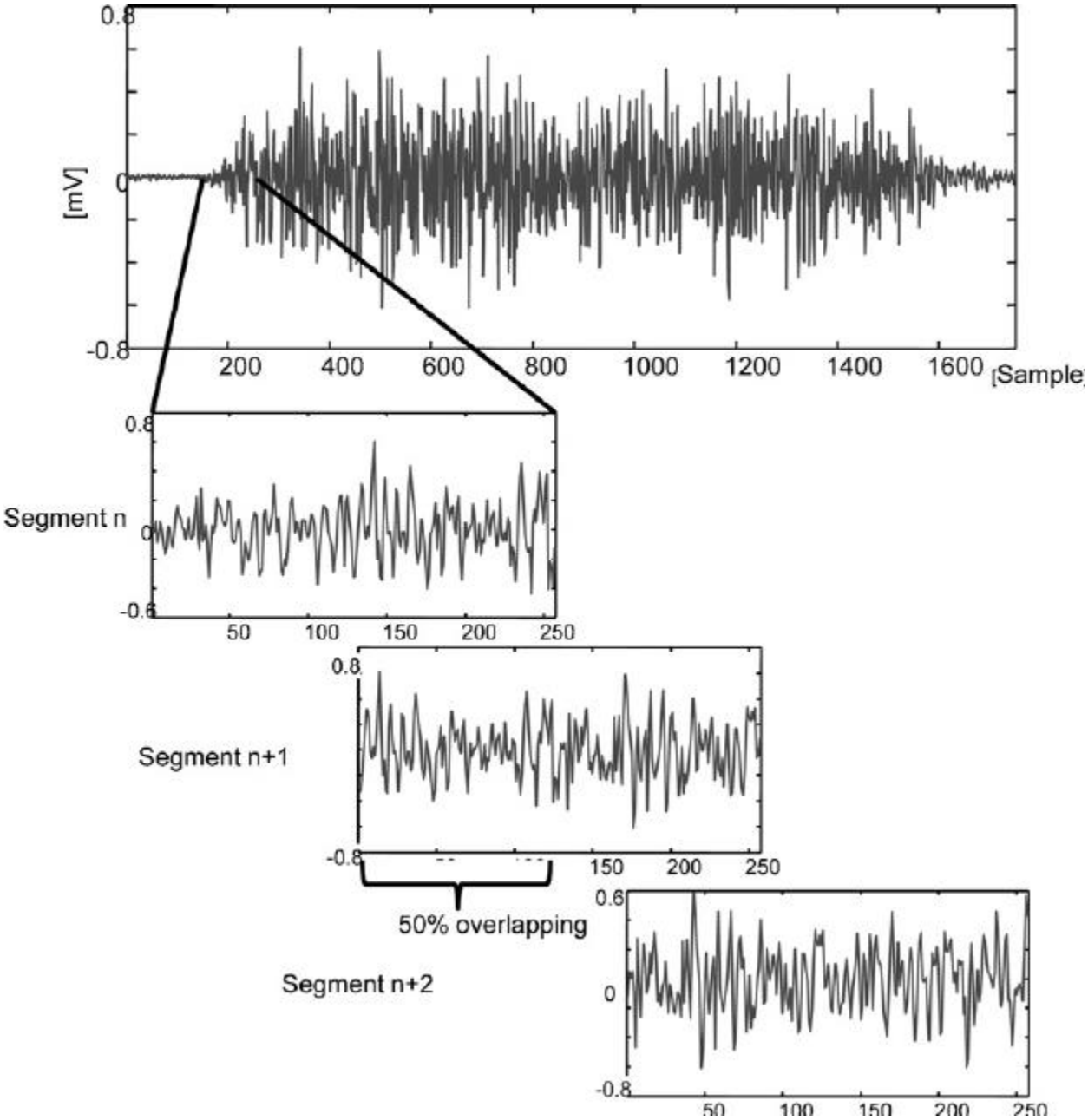

Fig **9** Sliding Window

The 13 mfcc features were extracted using pyAudioAnalysis. We tweaked the code for the library to extract only the required 13 mfcc features rather than extracting all the frequency and time domain features to reduce the processing time. Delta and Delta Delta features

were calculated using LibROSA which used 3 frames to calculate the estimation of Delta and Delta Delta. We used a context window of size 5 to capture the fluctuations in the speech sample. These were made by stacking the 5 frames after the current frame to make a single feature vector. Average Windows were made after this by specifying the average frames per sample and breaking the audio into equal windows of this size. Mean and Variance for all the features in each feature vector was calculated by taking all the average windows into consideration. This resulted in a feature vector of length 78. This feature vector was then normalized using standard normalization using mu and sigma calculated on the entire test data.

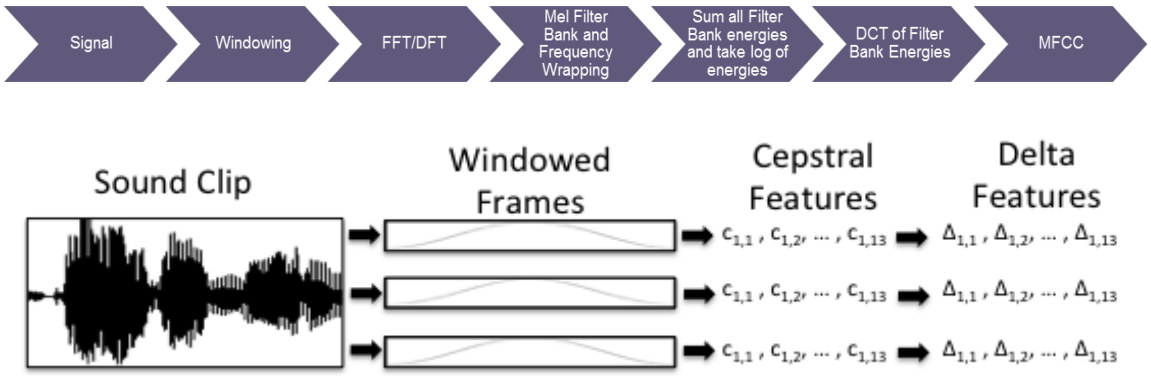

Figure **10** Feature Extraction Process

Preprocessing of Data

Since the recordings were of variables size we transformed the data by splitting each file such that we obtain a large number of equal duration files for consistency during the training as well the testing phase and also equalize the contribution from each recording. We obtained 1341 samples which were used for both training and testing. Since learning was supervised the labels for each language were to be specified. This was done by organizing the data into directories and while splitting each file the label was specified according to the directory.

4.2 Training the Initial Neural Network

Feature Selection was applied to the extracted features to extract a total of 180 features from each context frame. The features were ranked on the basis of chi squared statistics and the features with the k highest scores were selected, k being the number of features to be selected. The selected features are provided to the Baseline SGD Classifier as well as the Initial Neural Network. The SGD Classifier was implemented using the scikit-learn library and required only calling the fit method to train the model. Since the training was required to be in batches, warm start parameter which remembers the model trained previously was set to True. The Initial Neural network was implemented using Keras which provide a higher level of abstraction on top of the theano backend. The underlying neural network consisted of 3 fully connected layers with the first layer called the input layer containing neurons equal to the selected number of features i.e 180. The second layer called hidden layer contained 12 neurons which were determined by extensive tuning of the model. The third layer called the output layer contained neurons equal to the number of languages to be identified which are 3 in this project. The associated activation function for each of the neurons in the input and hidden layer is relu which stands for rectifier liner unit. The activation function used for the output layer is softmax which produces probability density as its output. The weights were initialized using a uniform initialization scheme and adadelta optimizer was used to control the learning rate. This optimizer makes the learning independent of the initial learning rate. Activity regularization L1 and L2 was added at each layer to avoid over fitting. The model aims to minimize the error function which the and is the actual process through which the model learns. Categorical crossentropy was used which is ideal for multiclass classification. The metric that we sought to maximize was accuracy which we found through extensive research was the only ideal metric for Speech related applications. All other parameters were left at default specified by the keras library. The model was trained for 30 epochs using a batch size of 30 samples. The data provided to the Neural Network was labeled data aimed at supervised learning. However, to adhere to the architecture of the Neural Network. The single output label was transformed into a 1 hot vector with the 1 corresponding to the actual spoken language. The training and testing was done using K Fold Cross Validation with K = 10.

Therefore, the training data consisted of 9 folds of the total data available. The data samples included in training were further shuffled using sklearn library before fitting the Initial Neural Network to the data. The model changes the weights and bias associated with each neuron in order to minimize the error function. The model generates the best two candidates for further classification by producing probabilities for each language. The two languages with the highest probabilities are specified as the candidates.

4.3 Training the Binary Neural Network

This phase involves feature selection for all possible pairs of languages (excluding the same language pair). This involves applying chi squared statistics on all pairs and selecting k best scoring features. The number of selected features for each pair were found out through tuning of the models and was finally settled at 380. The Binary Neural network was implemented using Keras which provide a higher level of abstraction on top of the theano backend. The underlying neural network consisted of 3 fully connected layers with the first layer called the input layer containing neurons equal to the selected number of features i.e 380. The second layer called hidden layer contained 22 neurons which were determined by extensive tuning of the model. The third layer called the output layer contained neurons equal to the number of languages to be identified which are 2 for the Binary case. The associated activation function for each of the neurons in the input and hidden layer is relu which stands for rectifier liner unit. The activation function used for the output layer is softmax which produces probability density as its output. The weights were initialized using a uniform initialization scheme and adadelta optimizer was used to control the learning rate. Activity regularization L1 and L2 was added at each layer to avoid over fitting. The model aims to minimize the error function which the and is the actual process through which the model learns. Binary crossentropy was used which is ideal for binary classification. The metric that we sought to maximize was accuracy which we found through extensive research was the only ideal metric for Speech related applications. All other parameters were left at default specified by the keras library. The model was trained for 30 epochs using a batch size of 30 samples. The data provided to the Neural Network

was labeled data aimed at supervised learning. However, to adhere to the architecture of the Neural Network. The single output label was transformed into a 1 hot vector with the 1 corresponding to the actual spoken language. The labels specified here were consistent with the Initial Neural network. The data samples included in training were further shuffled using sklearn library before fitting the Binary Neural Network to the data. The model changes the weights and bias associated with each neuron in order to minimize the error function. The model generates the final predicted language according to the trained model.

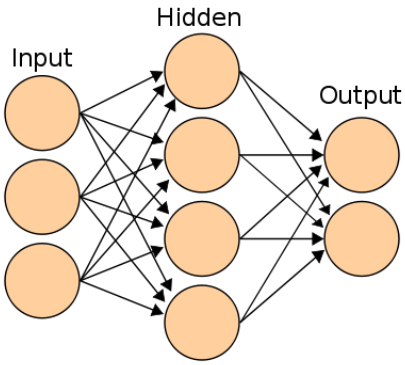

Fig **11** Representative Neural Network Architecture

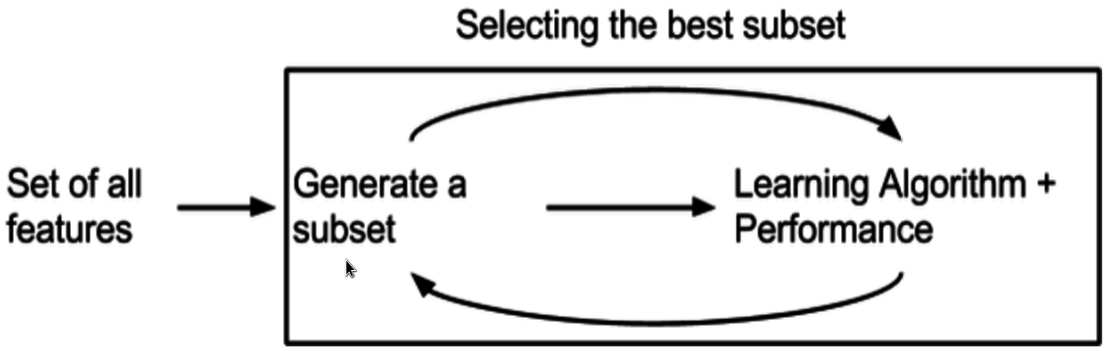

Fig **12** Feature Selection Process

4.4 Testing the Hybrid Neural Network

Testing of entire Language Detection System

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on "new" data. This is the basic idea for a whole class of model evaluation methods called *cross validation*.

The evaluation metrics were Accuracy, Recall and Precision. For a multiclass case it is calculated in the way depicted in the following figure.

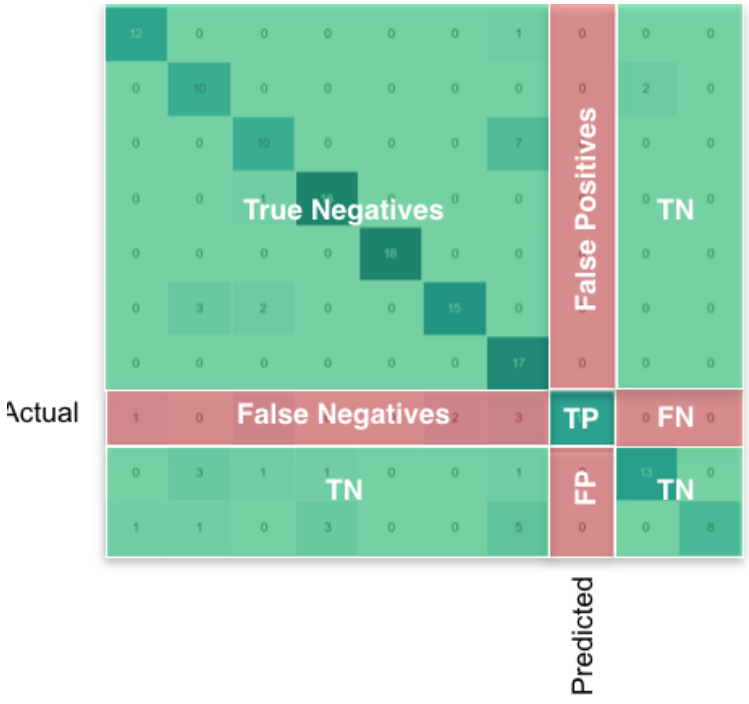

Figure **13** Multiclass Metric Calculation

Formulas for precision, recall and accuracy is given below

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Obtaining Test Data

Various audio samples were downloaded from Audio Lingua of the languages (Chinese, French, German). They were assumed to be clean. The audio was available in mp3 format. For each language approximately 2.5 hrs of clean speech was obtained. The speech samples spread over 170 different speakers including both male and female speakers. Each of the audio speech spoken by a single speaker was then broken in 5 sec speech samples. So, a one-minute speech was equivalent to 12 samples.

They were broken in 5 sec consecutive speech samples because we average out each speech sample into a single data point so as to reduce the effect of noisy samples. If an entire audio (2-3 min avg. audio length) is transformed to a single data point, we will lose a significant amount of training data. It was kept in mind that a speech from a single speaker do not become part of both training and test data.

Evaluation of test samples

Each speech sample (5 sec sample) was tested against the system. The system returned the estimated category of language to which it belongs along with the probability of its belongingness to that category.

Using this information, we constructed a confusion matrix of languages. Each row represents the language to which the sample actually belongs and column represents the result as given by the language identification system.

Using this confusion matrix, the true positives, true negatives, false positives and false negatives were calculated and henceforth accuracy, recall and precision was found.

Language	ch	fr	ge
ch	100	20	30
fr	10	80	60
ge	40	12	98

Figure **14** Sample confusion matrix for a total 150 samples

We used two types of validation testing.

1) **Holdout Method**

The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model. The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute. However, its evaluation can have a high variance. The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.

In this type of validation, we set out 60% of the entire data for training and rest 30% for holdout test. The 30% test samples were independent of training samples, i.e. even the speakers were distinct in both the sample sets.

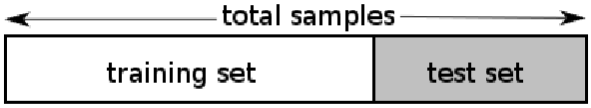

Figure **15** Division between Train and Test set

2) K Fold Validation

K-fold cross validation is one way to improve over the holdout method. The data set is divided into *k* subsets, and the holdout method is repeated *k* times. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set *k-1* times. The variance of the resulting estimate is reduced as *k* is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch *k* times, which means it takes *k* times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set *k* different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

While doing K-Fold Validation we used k=10, thus the entire model had to be trained 10 times. The variance and average accuracy is specified in the results section.

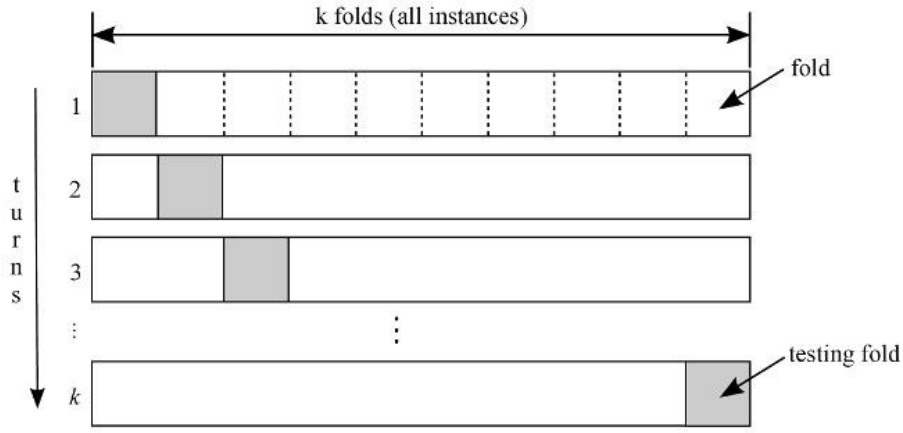

Figure **16** K Fold Cross Validation

Chapter 5: Results

Studying the variation of MFCC feature across languages in the set

To establish the strong grounds for using MFCC features as a discriminator we studied the variation of the features plotted as histograms for all the languages. The values of mean and standard variation were plotted as no of frames in a histogram. The following figures clearly show that MFCC features can serve as a good discriminating feature for languages

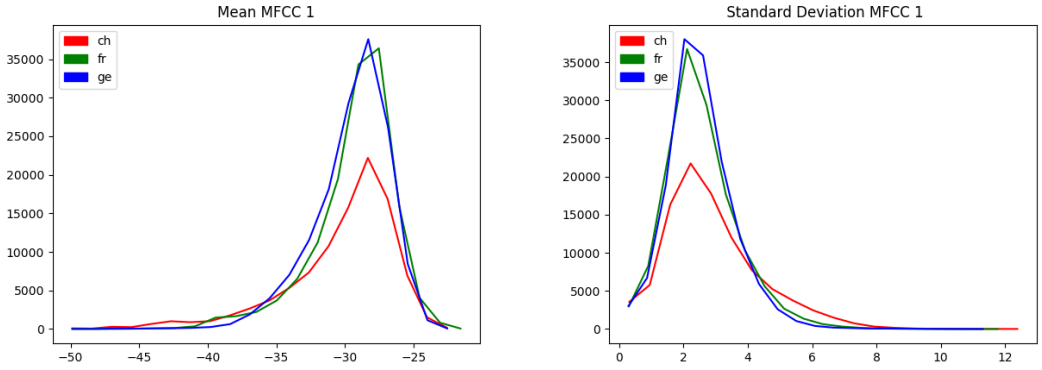

Fig **17** Mean and Standard Deviation of MFCC 1 across dataset

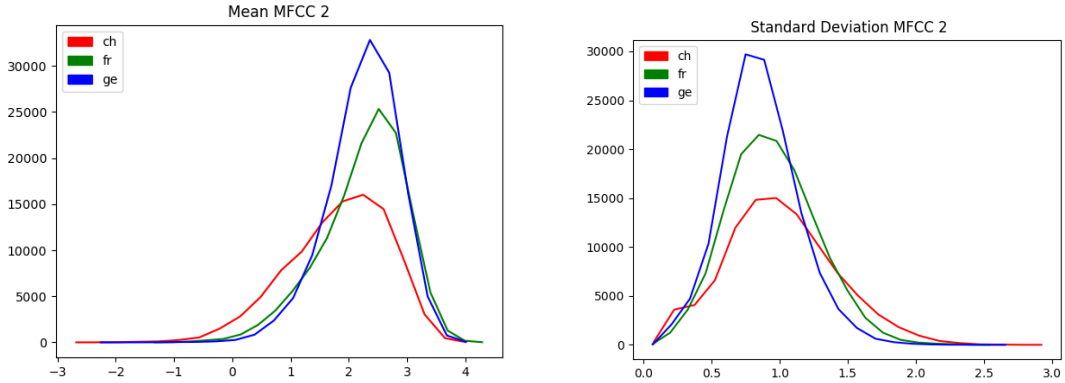

Fig **18** Mean and Standard Deviation of MFCC 2 across dataset

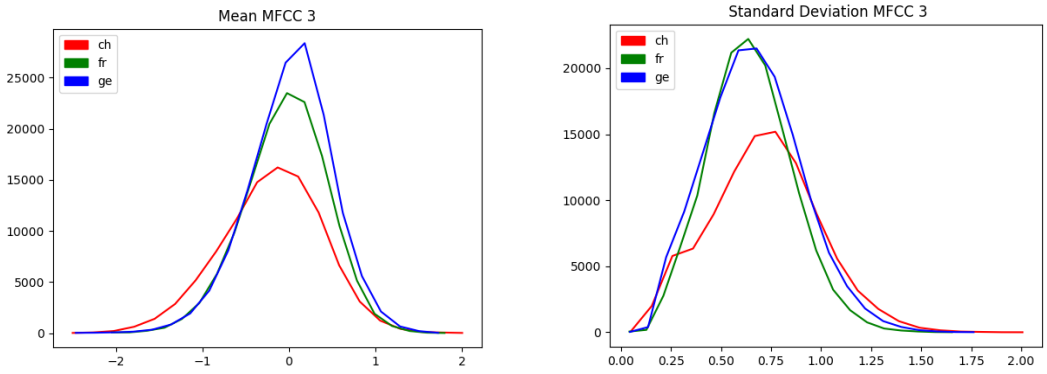

Fig **19** Mean and Standard Deviation of MFCC 3 across dataset

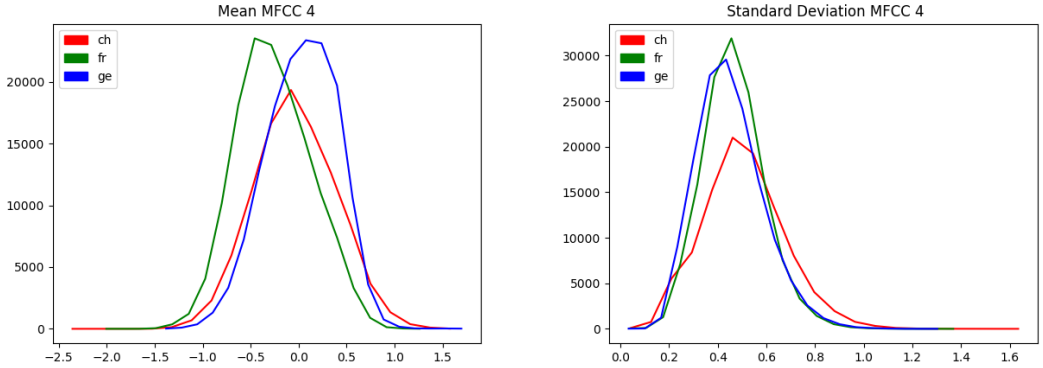

Fig **20** Mean and Standard Deviation of MFCC 4 across dataset

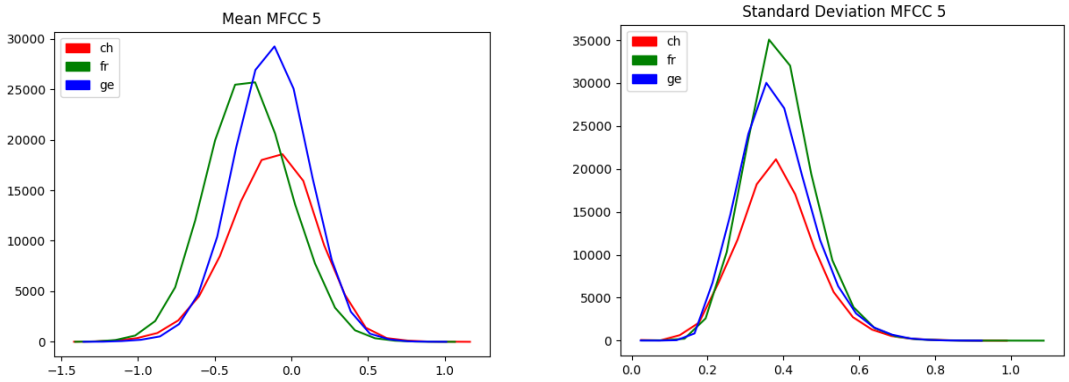

Fig **21** Mean and Standard Deviation of MFCC 5 across dataset

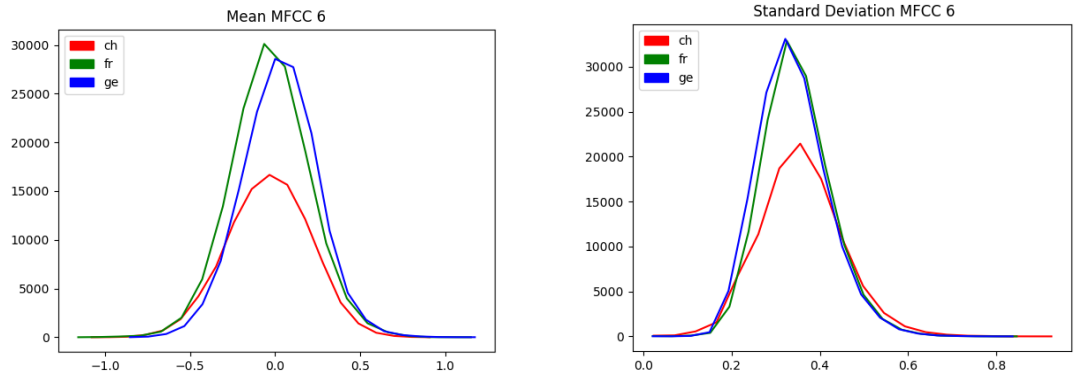

Fig **22** Mean and Standard Deviation of MFCC 6 across dataset

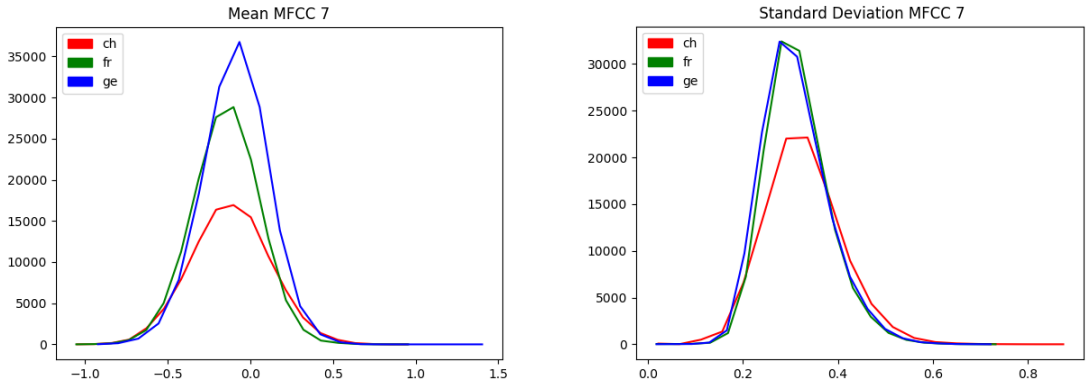

Fig **23** Mean and Standard Deviation of MFCC 7 across dataset

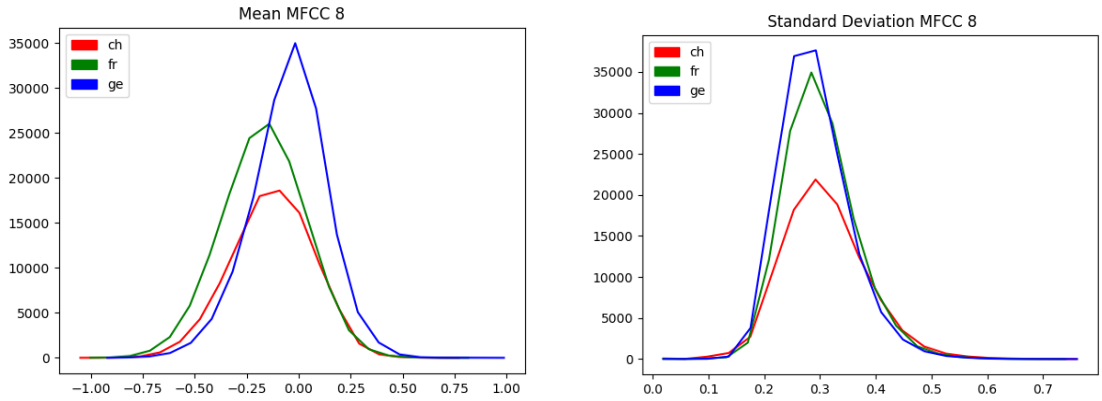

Fig **24** Mean and Standard Deviation of MFCC 8 across dataset

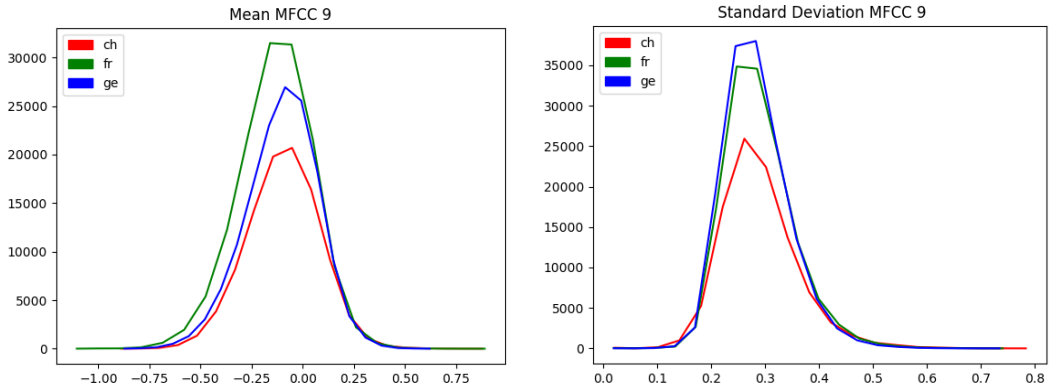

Fig **25** Mean and Standard Deviation of MFCC 9 across dataset

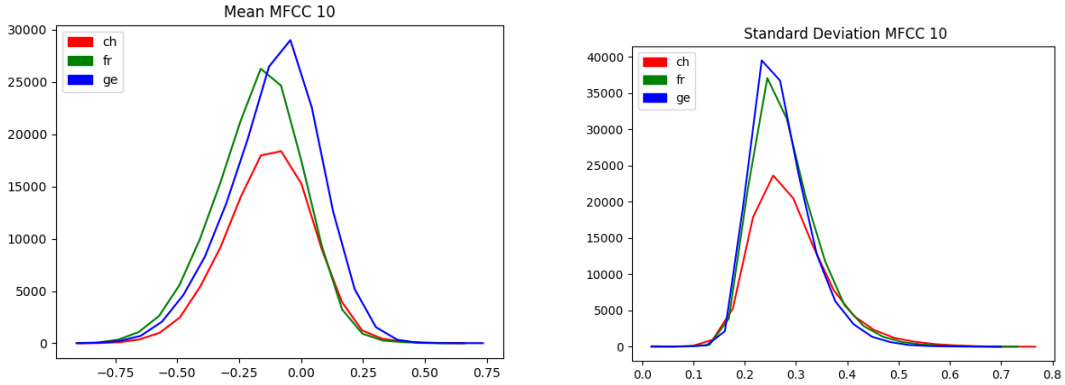

Fig **26** Mean and Standard Deviation of MFCC 10 across dataset

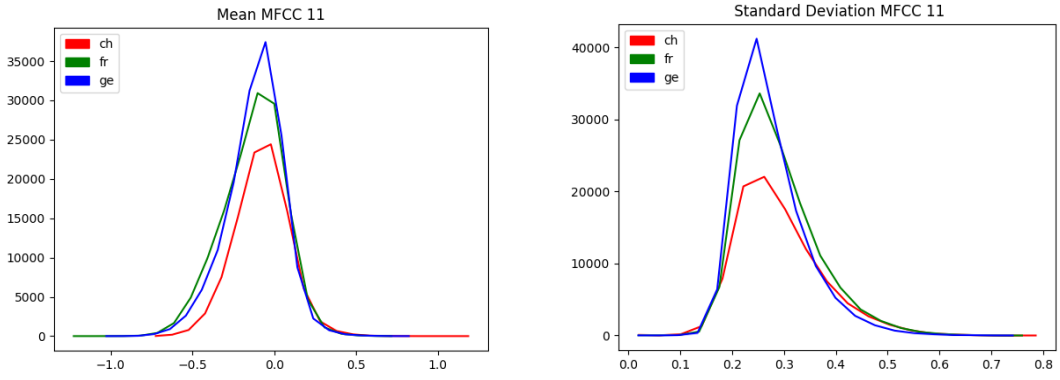

Fig **27** Mean and Standard Deviation of MFCC 11 across dataset

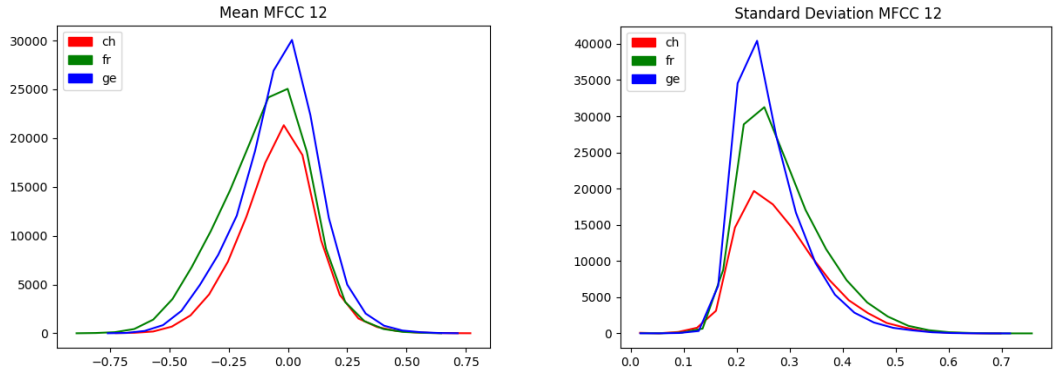

Fig **28** Mean and Standard Deviation of MFCC 12 across dataset

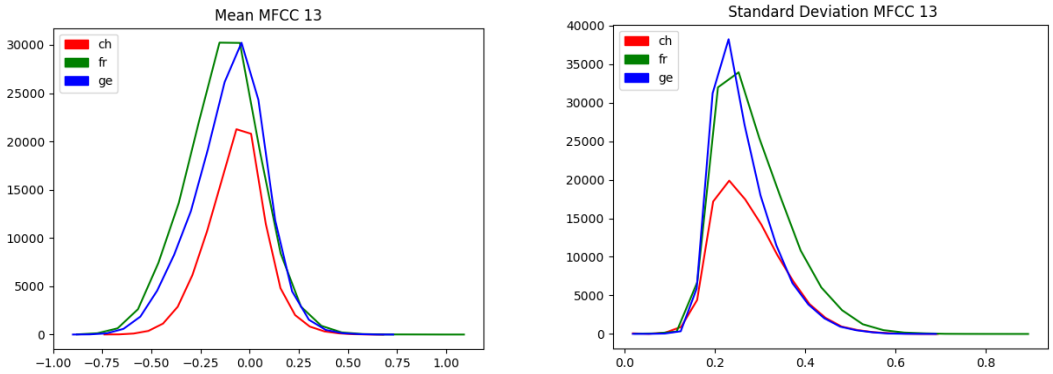

Fig **29** Mean and Standard Deviation of MFCC 13 across dataset

Calculating overall accuracy of the system using Holdout Validation Technique

	Chinese	French	German
Chinese	160	23	53
French	36	155	45
German	25	41	170

Accuracy: 79.20%

Fig **30** Overall Confusion Matrix

	Chinese	French
Chinese	222	14
French	11	225

Accuracy: 94.70%

Fig **31** Chinese vs French

	French	German
French	225	11
German	27	209

Accuracy: 91.95%

Fig **32** French vs German

	Chinese	German
Chinese	192	44
German	27	209

Accuracy: 84.96%

Fig **33** Chinese vs German

Studying the effect of context windows on accuracy

Using pure and static MFCC can be used as a measure for physical features of the vocal chords. However, these physical features are not enough to distinguish languages from one another. The variation of MFCC features over time, on the other hand, can be used for differentiating languages. Hence we use context windows to capture those temporal variations and here we study its effect on overall accuracy of the system.

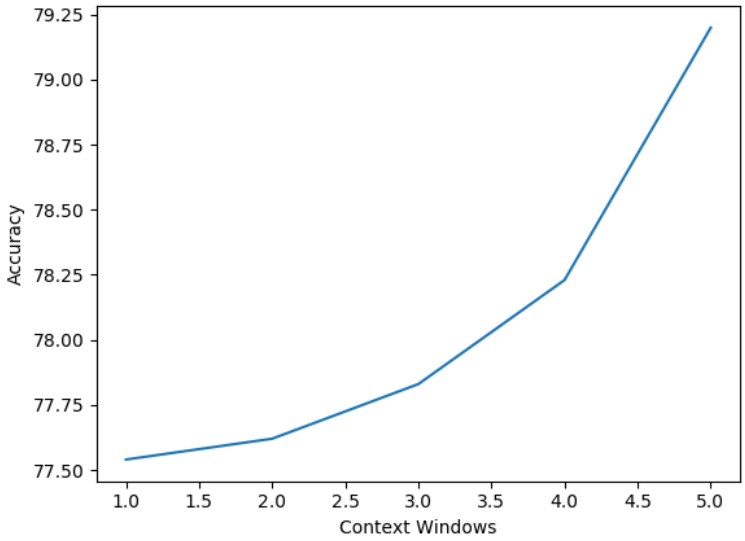

Fig **34** Effect of context window on overall accuracy of the system.
The increase in context window size increases
the overall accuracy of the system.

Studying the effect of number of features on accuracy

The features input to any Machine Learning model play a vital role in how the model performs. However, some of the features either have a little impact on the decision making or no impact whatsoever. These features do not impact the performance of the model but only increase the processing time and the dimensionality of the input. Therefore, we applied feature selection to focus only on the important features and reduce processing time. As seen from the graph below, accuracy decreases after increasing the number of features to a certain maximum.

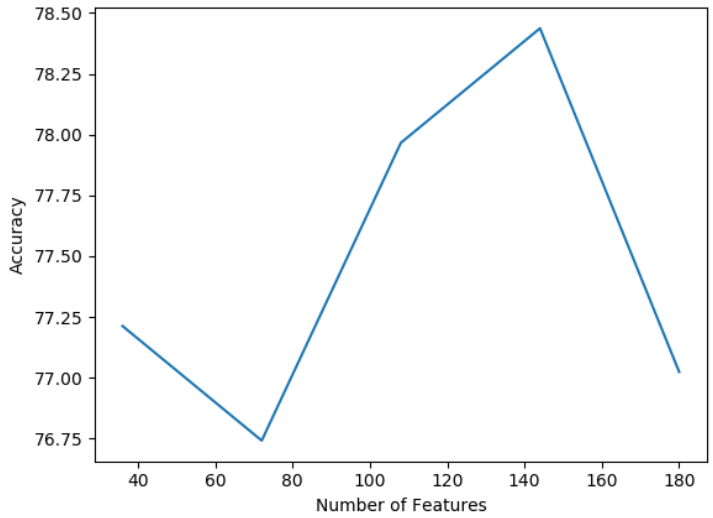

Fig **35** Effect of number of features on overall accuracy of the system. Increase in number of features after a certain limit decrease the accuracy

Calculating the overall accuracy using K-Fold Validation Technique

The entire dataset was divided into K (where K=10) randomly chosen equally partitioned blocks. Out of which one set was chosen as test set whereas other K-1 (where K-1=9) sets were chosen as training set. This process was repeated K times hence every data point was part of training sample at least and at most once in the whole validation process. The following plot represents the accuracy against the trial number.

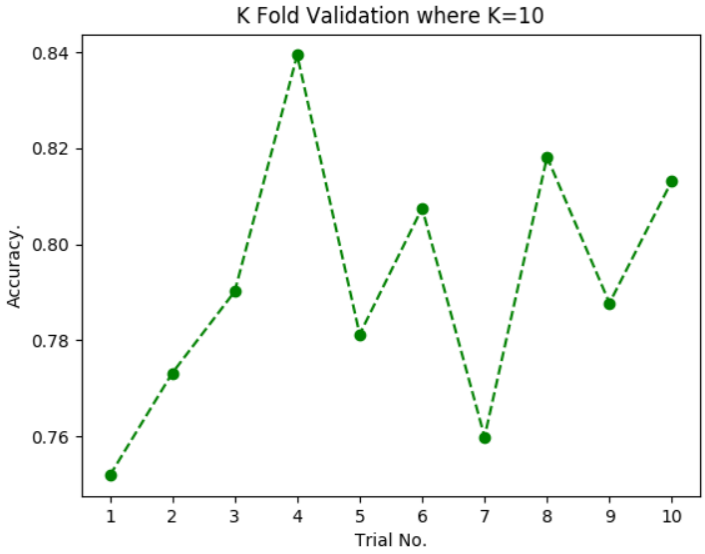

Fig **36** Results of the K Fold Validation of the language detection system. The average accuracy is 79.22% and a low standard deviation of 2.59 in the percentage accuracy is indicative of a stable system.

Comparison of Hybrid Neural Network against the Baseline SGD Classifier

The proposed model was tested against a baseline model which was selected to be a SGD Classifier. SVM based models have shown promising results in Spoken Language Identification, thus we compared our model to the SGD Classifier and achieved a much superior performance. This improvement in performance showed the promise of using DNN's for Spoken Language Identification and provided sufficient proof that the approach taken to solve the problem was much better than the previous approaches in the same field.

	Baseline SGD Classifier	Hybrid Neural Network
Accuracy	72.60%	79.20%
Precision	62.26%	68.87%
Recall	58.90%	68.50%

Fig **37** Comparison with the Baseline Model

GUI for visualization of the Hybrid Neural Network for any audio speech sample

The final model for the language prediction was integrated with a clean GUI to allow the user to test analyze audio speech sample. The GUI shows the frame wise prediction for the audio, which makes it suitable for a mixture of languages in a single sample. The predictions are color coded for better visualization. The GUI has a menu to select any .mp3 file for testing and displays the frame wise prediction as well the final prediction for the language.

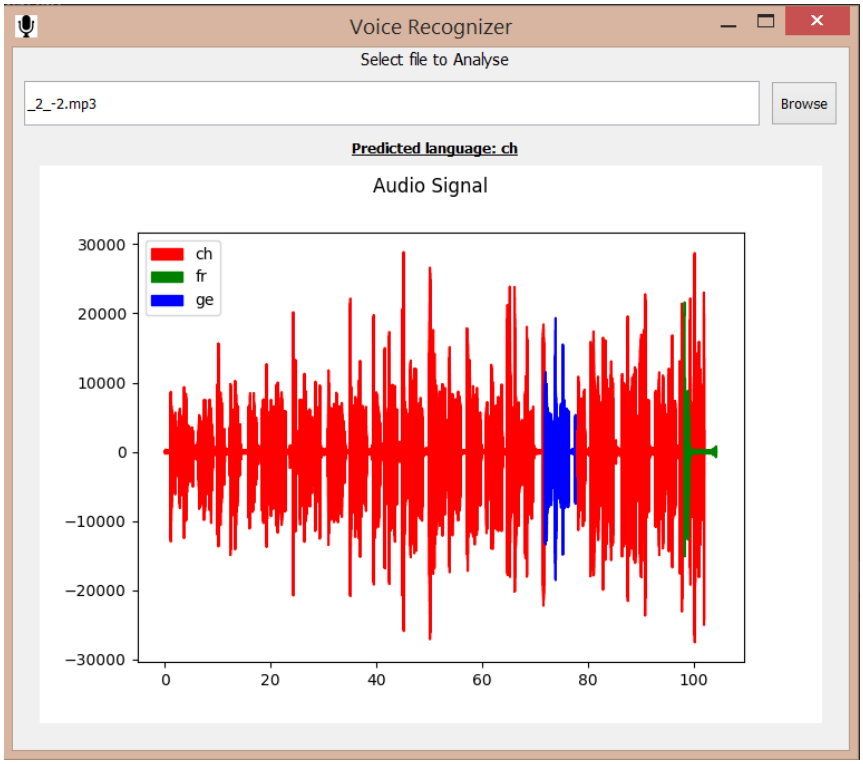

Fig **38** GUI for the Hybrid Neural Network

Real Time Analysis

All the parameters of the Hybrid Neural Network could be saved and that allowed for a fast prediction time. This fast prediction time paved the way for the system to be used for real time prediction. Threads were used to gather the input from the primary audio source of the device and pass that to the model for prediction. In this way a prediction was made for each second of the input audio. The processing time for each 1s audio sample was observed to be around 0.2s which was fast enough for real time processing. Real time prediction could be improved if the past second(s) audio was also considered for making the prediction at the current time, but that would even deteriorate the performance if the audio's language varied rapidly.

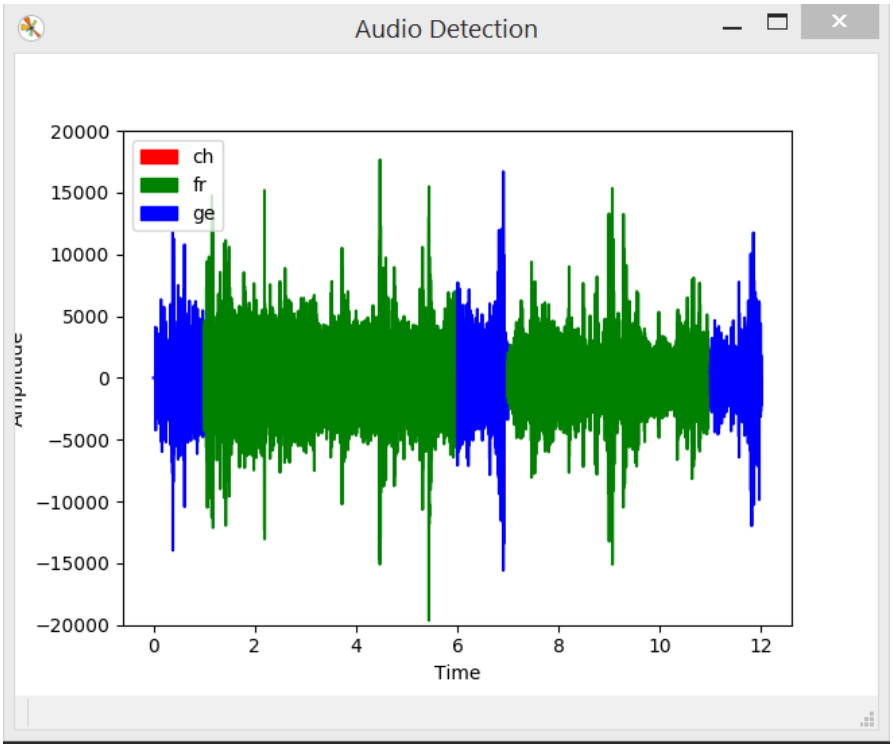

Fig **39** Real Time Analysis

Chapter 6: Conclusion and Future Work

Conclusion

Accuracy of the classification process by the deep neural network architecture suggests that the designed architecture is sufficient and powerful enough to be able to identify various languages. After trying various implementation of our project, we conclude that our two stage classification model is definitely an improvement over single stage model for language identification. Our main inspiration for adopting such a model comes from our early observations that languages are better resolved when considered pairwise. That is, we noticed that a feature which is useful in distinguishing some languages was not irrelevant for some pair of languages. Our observations were finally confirmed when we got significant improvement in accuracy of our model when we added a second binary classifier stage.

Future Works

The most noticeable scope of improvement is having an effective noise filtering system to provide clean human voice to our processing model. Our current real time analysis suffers from this problem which is highly noisy as opposed to clean audio samples on which our network was trained.

Accuracy can be further improved by using Shifted Delta Cepstral(SDC) features[9-10][27]. The use of the Shifted Delta Cepstral Feature Vectors allows for a pseudo-prosodic feature vector to be computed without having to explicitly find or model the prosodic structure of the speech signal. A shifting delta operation is applied to frame based acoustic feature vectors in order to create the new combined feature vectors for each frame.

Moreover, we can improve our implementation of model by providing incremental training facility. That is whenever new language is needed to be added, the model need not to be trained again, instead can only be trained on new language and merging it with existing trained model. It can further be extended to detect if the speech sample does not belong to the set of languages the system is trained for, so as to reduce the systems false positives.

Chapter 7: References

1. Y.K Mumusamy, E. Barnard, and R.A. Cole, "Reviewing Automatic Language Identification," IEEE Signal Process. Mag. II, 33 (Oct. 1994).

2. M. Zissman, "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech," IEEE Trans. Acoustic., Speech, Signal Processing, vol. 4, no.1, pp. 31–44, 1996.

3. N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and Reda Dehak, "Language Recognition via i-vectors and Dimensionality Reduction.," in INTERSPEECH. 2011, pp. 857–860, ISCA.

4. L. Riek, W. Mistrerra, and D. Morgan, "Experimenters in Language Identification," Technical Report SPCOT-91-002 (Lockheed-Sandets, Nashua, NH, Dec. 1991).

5. S. Nakagawa, Y. Ueda, and T. Seino, "Speaker-Independent, Text-Independent Language Identification by HMM," ICSLP '92 Proc. 2, Banff,Alberta, Canada, 12-16Oct. 1992, p. 1011.

6. M.A Zissman, "Automatic Language Identification Using Gaussian Mixture and Hidden Markov Models," ICASSP '93 Proc. 2, Minneapolis, 27-30Apr. 1993, p. 399.

7. A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," J Royal Statistical Society 39, 1 (1977).

8. L.E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," Inequalities 3, 1 (1972).

9. P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. J.R.Deller, "Approaches to language identification using gaussian mixture models and shifted delta cepstral features, "*INTERSPEECH*, 2002.

10. M.A.Kohler and M.Kennedy, "Language identification using shifted delta cepstra," *IEEE*, 2002.

11. Ignacio Lopez-Moreno, "AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS", 2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP),2014 (context windows)

12. M. A. Zissman, E. Singer, "Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling", ICASSP '94 Proceedings, vol. 1, pp. 305-308, 1994-April.

13. Kshirod Sarmah, “GMM based Language Identification using MFCC and SDC Features”, International Journal of Computer Applications (0975 – 8887) Volume 85 – No 5, January 2014

14. Pedro A. Torres-Carrasquillo, “Language Identification using Gaussian Mixture Model Tokenization”, Lincoln Laboratory, Massachusetts Institute of Technology

15. Julien De Mori, “Spoken Language Classification” , CS 229 – Machine Learning(Stanford)

16. G. Montavon. ” Deep learning for spoken language identification,” Machine Learning Group, Berlin Institute of Technology.

17. Machine Learning, Andrew Ng, Associate Professor, Stanford University; Chief Scientist, Baidu; Chairman and Co-founder, Coursera

18. Theano: http://deeplearning.net/software/theano/

19. Keras: https://keras.io/

20. scikit-learn: http://scikit-learn.org/stable/

21. pyAudioAnalysis: https://github.com/tyiannak/pyAudioAnalysis

22. LibROSA: https://librosa.github.io/

23. PyQt, https://wiki.python.org/moin/PyQt

24. Numpy, http://www.numpy.org/

25. Matplotlib,https://matplotlib.org/

26. Audio Lingua: https://www.audio-lingua.eu/

27. H. Wang, C.-C. Leung, T. Lee, B. Ma, H. Li, "Shifted-delta MLP features for spoken language recognition", IEEE Signal Process. Lett., vol. 20, no. 1, pp. 15-18, 2013.

Appendix

Tool Installation

1) pyAudioAnalysis

Install dependencies:

```
pip install numpy matplotlib scipy sklearn hmmlearn simplejson eyed3 pydub
```

Clone the source of this library:

```
git clone https://github.com/tyiannak/pyAudioAnalysis.git
```

2) LibROSA

The latest stable release is available on PyPI, and you can install it by saying

```
pip install librosa
```

librosa is also available on Anaconda. You can install it by saying

```
conda install -c conda-forge librosa
```

3) NumPy

Windows does not have any package manager analogous to that in Linux, so installing one of the scientific Python distributions mentioned above is preferred. However, if that is not an option, Christoph Gohlke provides pre-built Windows installers for many Python packages, including all of the core SciPy stack, which work extremely well.

4) Theano

Installation of the requirements supported only through conda.

- Python == 2.7* or (>= 3.3 and < 3.6)
- NumPy >= 1.9.1 <= 1.12
- SciPy >= 0.14 < 0.17.1
- BLAS installation (with Level 3 functionality)

Assuming all the dependencies are already installed, theano can be installed as:
With conda

If you use conda, you can directly install both theano and pygpu. Libgpuarray will be automatically installed as a dependency.

conda install theano pygpu

With pip

If you use pip, you have to install Theano and libgpuarray separately.

theano

Install the latest stable version of Theano with:

<sudo> pip install <--user> Theano[test, doc]

libgpuarray

For the stable version of Theano you need a specific version of libgpuarray, that has been tagged v0.6.2. Download it with:

git clone https://github.com/Theano/libgpuarray.git

cd libgpuarray

git checkout tags/v0.6.2 -b v0.6.2

Bleeding-Edge Installation (recommended)

Install the latest, bleeding-edge, development version of Theano with:

<sudo> pip install <--user> <--no-deps>
git+https://github.com/Theano/Theano.git#egg=Theano

5) Keras

Keras uses the following dependencies:

- numpy, scipy
- yaml
- HDF5 and h5py (optional, required if you use model saving/loading functions)
- Theano

It is assumed that all these dependencies are installed

To install Keras, cd to the Keras folder and run the install command:

```
sudo python setup.py install
```

You can also install Keras from PyPI:

```
sudo pip install keras
```

6) scikit-learn

Scikit-learn requires:

- Python (>= 2.6 or >= 3.3),
- NumPy (>= 1.6.1),
- SciPy (>= 0.9).

Assuming these dependencies are satisfied,

The easiest way to install scikit-learn is using pip

pip install -U scikit-learn

or conda:

conda install scikit-learn

7) pyAudio (Real time analysis)

Install using pip:

python -m pip install pyaudio

8) PyQt

Building and Installing from Source:

Downloading SIP

SIP must be installed before building and using PyQt5. You can get the latest release of the SIP source code from https://www.riverbankcomputing.com/software/sip/download.

Downloading PyQt5

You can get the latest release of the GPL version of the PyQt5 source code from https://www.riverbankcomputing.com/software/pyqt/download5.

Configuring PyQt5

In order to configure the build of PyQt5 you need to run the configure.py script as follows:

python3 configure.py

This assumes that the Python interpreter is on your path. Something like the following may be appropriate on Windows:

c:\Python36\python configure.py

If you have multiple versions of Python installed, then make sure you use the interpreter for which you wish to build PyQt5 for.

Building PyQt5

The next step is to build PyQt5 by running your platform's make command. For example:

```
make
```

The final step is to install PyQt5 by running the following command:

```
make install
```

9) matplotlib

For standard Python installations, install matplotlib using pip:

```
python -m pip install -U pip setuptools
python -m pip install matplotlib
```