EAI Endorsed Transactions
on Scalable Information System
Research Article EAI.EU

Deep Learning Approaches for English-Marathi Code-Switched Detection

Shreyash Bhimanwar[1], Onkar Viralekar[2], Koustubh Anturkar[3], Ashwini Kulkarni[4]

[1, 2, 3, 4] Department of Electronics and Telecommunication Engineering, COEP Technological University, Pune, India

Abstract

During a conversation, speakers in multilingual societies frequently switch between two or more spoken languages. A linguistic action known as "code-switching" particularly alters or merges two or more languages. The development of software or tools for detecting code-switching has received very little attention. This paper proposes a Deep Learning based methods for detecting code-switched English-Marathi data. These suggested methods can be applied to various applications, including phone call merging, Intelligent AI assistants, Intelligent travelling systems to assist travellers in navigation and reservations, call centres to handle customer service issues, etc. To create a system for code switch detection, our study demonstrates a detailed analysis of extracting several audio features such as the Mel-Spectrogram, Mel-frequency Cepstral Coefficient (MFCC), and Perceptual Linear Predictive coefficients (PLP). Our team's English-Marathi code-switched dataset served as the testing ground for our methodologies. Our model's accuracy was 92.99%, with 40 MFCC coefficients having energy coefficient serving as the zeroth coefficient.

Keywords: Code-Switching, Deep Learning, Log-Mel Spectrogram, Long Short-Term Memory (LSTM), Mel Frequency Cepstral Coefficients (MFCC), Neural Network, Perpetual Linear Prediction (PLP), Spoken Language Identification, Speech Recognition

Received on 03 June 2023, accepted on 30 August 2023, published on 25 September 2023

doi: 10.4108/eetsis.3972

*Corresponding author. Email: bhimanwarss19.extc@coep.ac.in

1. Introduction

In Code-Switching (CS), multilingual speakers switch between languages during communication. Therefore, speaking in two or more languages facilitates interaction with individuals from different geographical areas and cultural backgrounds. The main reason for code-switching is because the people belong to bilingual communities and because of the non-availability of some words in the vocabulary of the native language. According to a survey, 70.34% of Maharashtrians speak Marathi regularly and employ the code-switch action.

There are two types of code-switching. Type-1: Inter sentential CS occurs when the switch of the language is done at sentence boundaries, e.g. She is the CEO **आणि ती इथे दोन दिवसासाठी आली आहे**. Type-2: Intra-sentential CS, the words, and the phrases of one language are embedded into other, e.g., **मला माझा** current account balance check **करायचा आहे**. However, Due to a dearth of high-quality training data, little progress has been made in creating Deep-Learning based techniques to identify the code switch.

Code-switching is mainly found in motivational speeches, social media, navigation instructions, instructions to AI assistants, and phone calls. In many commercial contexts, such as advertising, healthcare, education, and entertainment, processing code-switched communication would improve user experience [21]. It helps educators to analyse code-switching patterns to learn more about language use and the difficulties that bilingual learners experience. Code-switching detection can be extremely important in call centres or customer service settings when bilingual agents answer calls in both English and Marathi. Call centre systems can route calls to the most qualified agents, offer language-specific instructions or resources, and enable successful communication between consumers and agents who are

knowledgeable in both languages by spotting code-switched utterances. Code-switched detection also helps navigation systems and AI assistants dynamically adapt their instructions to match the user's language preferences. This helps in improving their navigation experience.

Automatic Speech Recognition (ASR) systems mainly focus on converting spoken language into written text or symbolic representations, i.e., transcribing speech to text. Automatic Speech Recognition (ASR) systems can identify a few words of a constant language but cannot handle considerable code-switching in data [20]. Also, it is quite challenging due to the pronunciation changes from region to region. Hence in this paper, we are primarily focusing on developing a language identification system that can handle code-switching as well and classify them. Our research has noticed that there is no such kind of code-switch dataset for English and Marathi to carry out the research. To get around this restriction, we developed a code-switch dataset with enough variance in accent, age, gender, and other characteristics.

In this paper, we proposed a methodology for the task of language identification for code-switched English-Marathi data. For data preparation, we used YouTube and self-recorded data. To tackle the problem of low resources of data, we used speed perturbation-based data augmentation. It acts as a regularization technique by introducing variations in the data and reducing data bias. To increase the robustness of the data, we also included Room impulse responses in the utterances. By applying RIR augmentation, we simulate the effect of different acoustic environments on the speech signal. Also, the models can learn to extract meaningful speech features even in the presence of reverberation, echoes. We proposed different approaches for our task by converting this pre-processed audio into a) Log Mel spectrogram and applying CNN based model, b) MFCC coefficients and PLP coefficients and applying LSTM-based models. This methodology is important because it deals with low-resource data and improves its robustness. We further provide insights into the proposed deep learning-based approaches and compare them in the results section.

2. Related Works

A few code-switching ASRs and Spoken language identification (LID) reports cover different Native and non-native language combinations in the literature below. We Briefly reviewed the code-switching reports and summarized their silent attributes used by them as follows.A study was conducted on building automatic speech recognition of Frisian speech containing code-switches to Dutch using multilingual DNN [1]. Audio data can be converted into Mel-Spectrograms, and CNN and LSTM architectures can be applied for the classification. The spectrogram was discretised using a Hann window along with 129 frequency bins. This also proposes CNN-LSTM architecture to train grayscale images of the spectrogram[3].

Recurrent Neural networks are found to be effective in learning the relevant features for the language classification[2][3][19]. The feature extraction techniques like Log-Mel-Spectrogram, Mel-Frequency cepstral coefficients (MFCC), Perceptual Linear Prediction (PLP), Relative spectral transform Perceptual Linear Prediction (RASTA-PLP) and Linear Predictive coding (LPC) coefficients can be used for Automatic Speech Recognition (ASR)[4]. The ideal overlapping window length for converting audio to spectrogram is between 10-30 ms, with left /right Neighbours to be –20 and +5 ms. Data augmentation techniques include Noise Addition, Time Sifting, Frequency and Time masking, Pitch changing and Speed perturbation [2]. Suitable speed factors for varying speeds of audio signals are 0.9x, 1.0x, 1.1x[6]. Traditional Language identification systems use i-vector and DNN systems for language recognition. Still, in recent years, it has been found that Neural Networks are best suited for Language Identification (LID) tasks as they are more straightforward in design and provide high accuracy of about 90% [3]. Another study investigates the performance of acoustic feature methods, which includes MFCC, Linear Predictive cepstral coefficient (LPCC) and PLP using Hidden Markov's model (HMM) classifier [4]. The state-of-the-art LSTM architecture outperforms the traditional DNN architectures for speech recognition [9][10][11][12][13]. LSTM RNNs are well-suited for modelling sequence data due to their hidden layer with recurrent connections and memory blocks and gates for regulating information flow within the network. As a result, they are effective for modelling speech signals. Furthermore, research has indicated that utilizing MFCC features with energy coefficient can lead to improved outcomes [16].

3. Proposed Methodology

In this section, we present our proposed approach for code-switch detection. Our approach utilizes three distinct methods to effectively identify code-switched data: Log-Mel Spectrogram, Mel-frequency Cepstral Coefficients (MFCCs), and Perceptual Linear Prediction (PLP). Additionally, we apply various augmentation techniques to the dataset, such as Speed perturbation, Room Impulse. Responses (RIRs) and the addition of noise sampled at 16 kHz with 16-bit precision. These augmentation methods aim to enhance the robustness and diversity of the training data.

The first method involves the extraction of Log-Mel Spectrograms, which are representations of speech signals in the frequency domain. As Log-Mel Spectrograms can be treated as images, we employ a Convolutional Neural Network (CNN) to process and analyse these spectrograms.

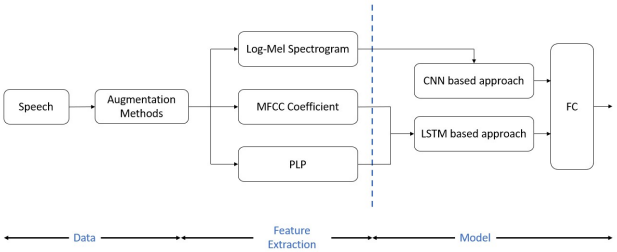

Figure 1. Schematic Flow of Proposed Method

The subsequent methods focus on extracting MFCCs and PLP coefficients from the speech utterances. These features capture essential characteristics of the audio signals, particularly related to the spectral and cepstral domains. Since the input data for these methods consists of ordered sequences of spoken language, we employ Long Short-Term Memory (LSTM) networks. The bidirectional nature of the network allows it to capture the context before and after code-switching points, enabling accurate detection of code-switching instances. LSTM's memory-like behaviour helps capture long-range dependencies, making them suitable for processing MFCCs and PLP coefficients. Detailed Methodology is discussed as follows:

3.1. Data Collection

Due to the unavailability of large Marathi and English-Marathi CS datasets, we decided to create our dataset for our experiment. In order to have diverse data, we applied two modes of Data Collection: a) Recording Apps and b) YouTube Videos like News, Speeches, Broadcasts, Podcasts etc. Their details and characteristics are as follows:

Self-Recorded Dataset: There are a number of Recording Apps available online to record audio in WAV format. The dataset collected from these apps has a sampling frequency of 44.1 kHz with 16 bits. The dataset consists of different Male and Female speakers, covering various aspects of variations of audio signals such as pronunciation, noise, pitch, gender, age etc.

YouTube Data Collection: We used YouTube data as the second source to obtain audio samples. We downloaded YouTube videos and extracted audio from them by splitting them into segments of 7 seconds each. We inspected every sample for CS data and cleaned the samples that contained a decent number of embedded words in a non-native language.

The obtained dataset contains many desired properties, such as High sound quality and a good mix of speakers. Further data augmentation and noise injection techniques are used to diversify our dataset. The statistics of our dataset are given in Table 1.

Table 1. Conv2D Neural Network Layers

Language	Label	Gender	Sample	Total Samples (Augmented)
Marathi	MAR	F	423	2914
		M	423	
English	ENG	F	430	2930
		M	423	
Code-Switch	ENG_MAR	F	440	2946
		M	439	

3.2 Data Preprocessing

The entire dataset has been converted into an uncompressed lossless WAV format. Initial sampling rates for audio files were 44100 Hz and 48000 Hz, down sampled to a 16 kHz sampling frequency to reduce computation time. In the process of preparing the dataset, we employed various augmentation techniques to enhance the diversity and robustness of the data, which include speed perturbation, room impulse response (RIR) and the addition of noise. This section describes how each technique was applied and its impact on the dataset.

Speed Perturbation: We applied speed perturbation to the speech utterances using librosa. time_stretch () library for accounting for natural variations in speech rate. It involves altering the temporal aspect of the speech signal by accelerating or decelerating it. We utilised two variations: increasing the speed of speech by 0.9 and 1.1. The application of speed perturbation served two primary purposes. Firstly, it introduced variability in speech rate, enabling the model to handle different speaking speeds. Secondly, it expanded the dataset by generating multiple versions of each utterance at varying speeds, thereby augmenting the training data with increased diversity.

Room Impulse Response (RIR): We incorporated room impulse response (RIR) augmentation by convolving it with speech data to simulate the effect of different acoustic environments. RIR captures the characteristics of sound propagation within a room, including reflections, echoes, and reverberation. We simulated different acoustic conditions by convolving the speech signals with RIRs generated for various room configurations. This augmentation aimed to improve the model's ability to handle real-world scenarios with diverse acoustic environments. Exposing the model to different room characteristics during training taught it to extract meaningful speech features even in the presence of reverberation and echoes, increasing its robustness.

Addition of Noise: To simulate realistic acoustic conditions, we introduced noise augmentation. The noise samples were sourced at a sampling rate of 16 kHz with 16-bit precision. The noise was carefully added by maintaining controlled signal-to-noise ratios (SNRs). The addition of noise has enhanced the model's ability to

handle speech signals corrupted by noise, as encountered in real-world environments. All these augmentation techniques also act as a regularization technique. This helps in preventing over-fitting of the model. By applying these techniques, we add a form of controlled randomness to the training process, encouraging the model to learn more robust and generalised representations.

3.3. Feature Extraction

Feature extraction is a sequence of feature vectors for input speech signals to classify the code-switch utterances. In this paper, several feature extraction methods have been performed and discussed to extract static parameters along with their first-derivative and second-derivative dynamic parameter coefficients. The above feature coefficients are extracted with 25ms window and 10 ms hop lengths. Delta and Double Delta Features: The speech recognition performance can significantly be enhanced by computing the time derivative of static parameters (given in eq. (1)). The trend of speech signals depends on the time axis's frame-by-frame analysis. The time derivatives (Delta) and accelerations (Double Delta) represent the speech rate and acceleration of the speech, respectively. The dynamic parameters can be computed using the following equation,

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \qquad (1)$$

where d_t is Delta coefficient at time t in the term of static coefficient C_{t+n} to C_{t-n}. Again, the same formula is applied to Delta's coefficient to get acceleration parameters. These features are essential for a speech recognition system to become independent of speakers. In the following sections, we will see more about these features and their detailed analysis.

Keras sequential layer for Log-Mel-Spectrogram: The Log Mel-spectrogram is extracted, which involves converting the spectrograms to the log-Mel scale by applying Mel filter banks (Equation 2) and converting power to decibel form [18]. During our experiment, we found that training time increases drastically because of the steps involved in finding the log Mel-spectrogram for each audio file. Due to the above constraints, the Keras sequential model with four layers is used to reduce the training time as it uses real-time GPU. The first layer computes a short-time Fourier transform (STFT) on the original audio waveform. The second layer computes the magnitude of STFT. Mel-filter banks are applied at the 3rd one. And finally, 4th layer converts these magnitudes to decibels (db).

The Mel-Frequency Cepstral Coefficient (MFCC): The Mel-Frequency Cepstral Coefficient (MFCC) is a popular feature extraction approach used in speech recognition. It is based on the frequency domain using the Mel scale, similar to the human ear scale. MFCC coefficients are robust to changes caused by the speakers and the recording environment. This method extracts speech characteristics that are comparable to those utilised by people to hear speech while, at the same time, de-emphasising all other information. The first step in the MFCC computation is the speech signal's windowing to divide it into overlapping frames. After windowing, the Fast Fourier Transform (FFT) is used to determine each frame's power spectrum. The logarithmic Mel-Scale filter bank is applied to the Fourier-transformed frame. This scale is approximately linear up to 1 kHz and logarithmic at greater frequencies. The relationship between the Mel-scale and the frequency of the signal is given by:

$$\text{Mel}\,(f) = 2595 * \log\left(1 + \frac{f}{700}\right) \qquad (2)$$

Where, mel(f) is the frequency (mels) and f is the frequency (Hz).

The final step is the computation of the Discrete cosine transform (DCT) of output from the filter bank. MFCCs and their Delta and Double Delta features were also extracted for improvement in accuracy. Fig 1 depicts the MFCC workflow.

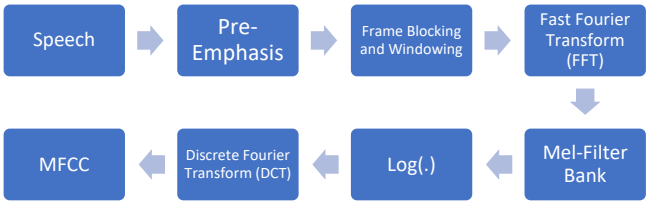

Figure 2. MFCC computation steps

Perceptual Linear Prediction (PLP): Perceptual Linear Prediction is a frequency-based feature extraction method for speech recognition. These features are characterized to match the human auditory system using an auto-regressive all-pole model [15]. The general flow diagram is shown in Fig 2. Equation 3 converts frequency warp into bark scale to find frequencies with low resolution.

$$Bark(f) = 13\arctan(0.00076f) + 3.5\arctan\left(\left(\frac{f}{700}\right)^2\right) \qquad (3)$$

The advantage of using PLP features over MFCC is that PLP coefficients are Noise immune. Also, it discards irrelevant audio information, which is unnecessary for speech identification. The number of PLP features extracted was 13, along with its Delta and double delta features were also extracted for improvement in accuracy.

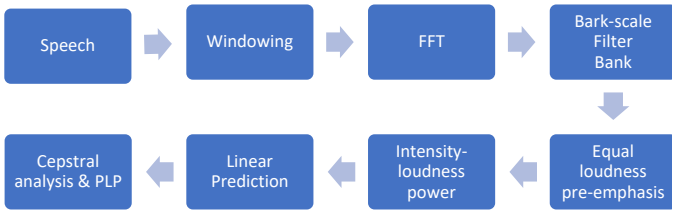

Figure 3. PLP computation steps

3.1. Models

Conv2D: The following design principles are applied to our model.

- Each convolution layer is followed by a MaxPooling layer which helps to reduce the variance, and computation complexity, e.g. (2*2 Max/Avg Pooling) reduces 75% of data. Sometimes average pooling can't extract the good features as it considers the average value, which may or may not be necessary. In contrast to this, MaxPooling helps in extracting the sharpest features.
- The rectifier linear unit (ReLU) activation layer is applied after each convolution layer. Layer normalisation is applied to the input layer, which is the output of the Log Mel-spectrogram Keras sequential layer. This helps in speeding up the training of neural networks.
- The model ends with two dense layers, out of which one is the final output layer with three neurons and SoftMax activation.

Table 2. Conv2D Neural Network Layers

Legend	Output Shape	Filter size	Units
Mel-Spectrogram Layer	(None, 700,40,1)	-	-
Layer Normalization	(None, 700,40,1)	Axis=2	-
Conv2D-(Relu)	(None,700,40,8)	(7,7)	-
MaxPooling2D	(None, 350,20,8)	(2,2)	16
Conv2D-(Relu)	(None,350,20,16)	(5,5)	-
MaxPooling2D	(None,175,10,16)	(2,2)	16
Conv2D-(Relu)	(None,175,10,16)	(3,3)	-
MaxPooling2D	(None, 88,5,32)	(2,2)	32
Conv2D-(Relu)	(None, 88,5,32)	(3,3])	-
MaxPooling2D	(None,44,3,32)	(2,2)	32
Conv2D-(Relu)	(None,44,3,32)	(3,3)	-
Flatten	(None,4224)	-	-
Dense	(None,64)	-	64
Dense (Softmax)	(None,3)	-	3

Model details: As mentioned in section 3.1 audio signal has sampling rate and time duration of 16 kHz and 7 sec respectively. Hence, the model has an input size of (Batch-Size, 112000, 1). Table 2 represents the detailed layer-by-layer visualization of the model, including the hyper-parameters. Further hyper-parameters optimization is discussed in the Experiment Results section.

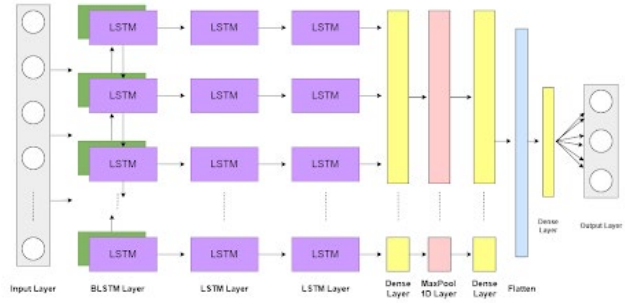

Figure 4. Layer By Layer Visualization of LSTM

Long short-term memory (LSTM): The following design principles have been applied to our model.

- We added a Bi-directional LSTM (BLSTM) layer with 256 units. The bidirectional nature of the network allows it to capture the context before and after code-switching points, enabling accurate detection of code-switching instances, while LSTM's memory-like behaviour helps in capturing long-range dependencies [16]. This is followed by two LSTM layers with 128 and 64 units, respectively.
- Three dense layers are added to the network and MaxPooling 1D and flattened layers are used.
- Dropouts and l2 regularizer are used to avoid overfitting problems. The dense layer with an l2 regularizer follows the dropout layer as an activity regularizer.

Table 3. LSTM Neural Network Layer

Layer	Output Layer	No. of Units
Input Layer	(None,699,39)	-
BLSTM	(None,699,512)	256
LSTM-1	(None,699,128)	64
LSTM-2	(None,699,64)	64
Dense-1	(None,699,64)	64
MaxPooling1D	(None,349,64)	-
Dense-2	(None,349,32)	32
Flatten	(None,11168)	-
Dropout	(None,11168)	0.25
Dense-3	(None,32)	32
Dense (Softmax)	(None,3)	3

Model details: Long Short-Term Memory Recurrent Neural are useful for analysing significant features in a sequence of inputs over a long distance and for capturing the relationship between elements in the acoustic signal over time. The audio data is sampled to 16 kHz by truncating their length to 7 sec. The model has an input shape of (batch_size, 699, 39). Table 3 and Fig 4 represent the model's detailed layer-by-layer visualization.

4. Experimental Results

This section contains the experimental setup and results of different techniques. All the details regarding the use of datasets, hyperparameter optimization and evaluation metrics of the different proposed approaches using various feature extraction techniques, including MFCC, Log-Mel Spectrogram, and PLP coefficients, are illustrated.

Dataset: Long The experiments were carried out using Self Recorded and YouTube datasets as discussed in section 2. To increase the dataset to a considerable amount, we applied the data augmentation technique of speed perturbation, which includes slowing the audio by 0.9 and fasting it by 1.1. We included Room Impulse Responses (RIRs) having different categories of small, large and medium rooms and noises sampled at 16 kHz 16-bit precision [14] into the speech utterances. The recorded 2581 utterances of the YouTube dataset and 1096 utterances of the Self-Recorded dataset have been augmented to 10324 and 4384, respectively by increasing speed to 1.1, reducing speed to 0.9 and convolution with RIRs' with addition of noise. For the experimentation purpose, we used a test dataset separated from training and validation. The recorded utterances are partitioned into training, validation, and testing sets with ratios of 0.6, 0.2 and 0.2. Analytical results are discussed in Table 4.

Evaluation Metrics: The evaluation metrics used to test the performance are F1 score and accuracy. The model's performance is evaluated using the following equations (4) and (7):

$$\text{Accuracy} = \frac{TP}{TP+FP+FN+TN} \quad (4)$$

$$\text{Precision(P)} = \frac{TP}{TP+FP} \quad (5)$$

$$Recall(R) = \frac{TP}{TP+TN} \quad (6)$$

$$F1Score = \frac{2*P*R}{P+R} \quad (7)$$

Where, TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

Hyperparameter Optimization: We used the Keras tuner for hyperparameter optimization, a grid-based or random search algorithm provided by the Keras framework, to find the optimum set of hyperparameters. Our observations for two models CNN and LSTM, are summarised as follows:

- Number of convolution layers: 5 convolution layers (specifically, CNN layers) were found to provide the best results among [3, 4, 5, 6] layer.
- Number of filters for CNN layer: The filters for convolution layers are chosen from [8, 16, 32] for the different layers.
- Kernel size: kernel sizes of the CNN are chosen from [3, 5, 7]. Convolution layers with the 1st and 2nd layers having filters (7,7) and (5,5), respectively, give us the best results. The size of filters is kept large to capture the suitable receptive field, i.e., the patterns in the input. The rest of the kernel sizes are kept as (3,3).
- Dropout and Regularizer: We experimented with different values such as [0.2, 0.25, 0.3, 0.35, 0.4]. Dropout value of 0.2 is used to avoid the development of co-dependency among each other.

Long Short-Term Network (LSTM):

- Number of units of BLSTM: We also experimented with different values such as [32, 64, 128, 256], but 256 worked well for the model.
- Number of units to LSTM: Different values such as [16, 32, 64, 128] have been experimented but 64 worked well for both layers of LSTM in the model.
- Dropout: We experimented with different values such as [0.25, 0.3, 0.35, 0.4, 0.5]. A dropout of 0.25 worked well for our dataset.

Performance Evaluation: We have done a feature-based performance evaluation, and the results are discussed in this section. We applied the K-fold cross-validation method with three folds to increase the significance of the results on the training and validation dataset. The results presented in the last column of the Table 4. are the average values of all folds. Second, third and fourth column shows the best f1 scores for the fold among three.

Log-Mel Spectrogram: As mentioned in section 3.1, all the audio files were fixed to a length of 7sec. Log-Mel spectrogram for each audio file is computed by keeping n_mels equal to 40, window length and hop length of 25ms and 10ms respectively. We generated the Log-Mel spectrogram using Keras sequential layers. Fig.5 represents the confusion matrix of the Testing dataset.

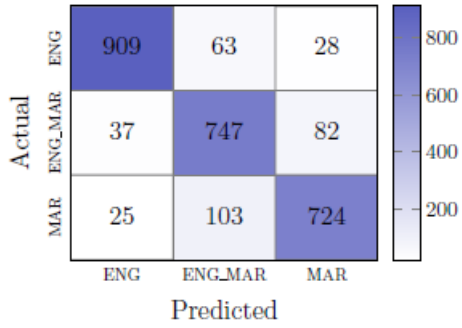

Figure 5. Log Mel Spectrogram

MFCC: The Mel-Frequency cepstral coefficient of each audio file is computed by varying the size of coefficients from 13 to 39. Again, the time derivatives (Delta) and accelerations (double Delta) are computed as mentioned in section 4. In the first step, 39 MFCC coefficients were applied to the model. In the next step, MFCC + Δ + ΔΔ (13+13+13) coefficients were applied, and Fig 6 and 7 represent the confusion matrix of the Testing dataset.

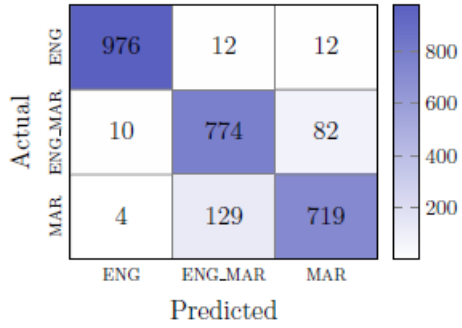

Figure 6. MFCC without energy

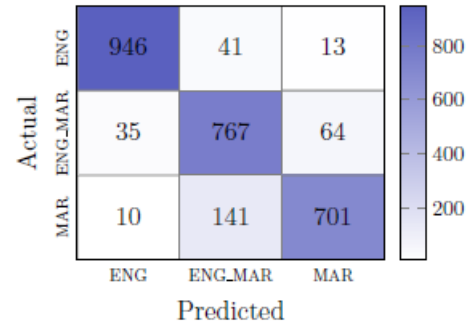

Figure 7. MFCC+Δ + ΔΔ without energy

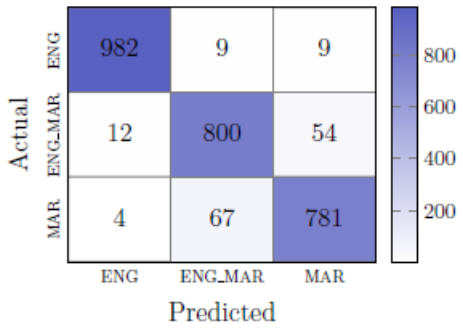

Figure 8. MFCC with energy

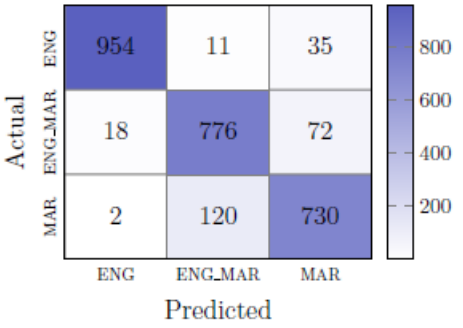

Figure 9. MFCC +Δ + ΔΔ with energy

PLP: The final analysis for the Predictive Linear Perceptual coefficients varies the number of features from five to twenty.

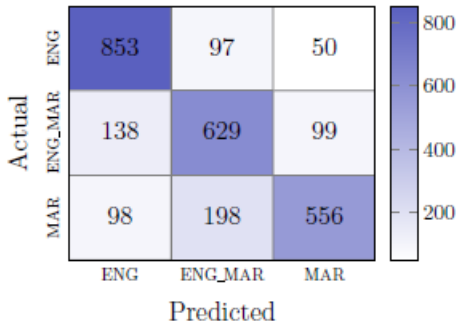

Figure 10. PLP coefficients

Finally, the performance comparison of the number of coefficients is discussed in Table 4.

Table 4. Results

Feature Extraction methods	Network	F1 Score of Best Fold			K-Fold Accuracy
		Eng	Mar	Eng-Mar	
MFCC (without energy)	LSTM	98.09	86.22	86.81	88.06
Log Mel Spectrogram	Conv2D	92.23	85.88	83.98	87.56
MFCC + Δ + ΔΔ (without energy)	LSTM	94.74	85.80	84.65	86.41
MFCC + Δ + ΔΔ (with energy)	LSTM	96.29	90.51	89.26	92.99
PLP	LSTM	81.54	71.25	70.17	74.00

Table 4 discusses the overall results of the test dataset with different feature extraction methods and different models. The Table shows that the use of MFCC coefficients with the addition of energy coefficient achieved better results than any other method.

5. Conclusion

In conclusion, this research paper explored three different methods for code-switch detection. The first method utilized log-mel spectrogram images and achieved an accuracy of 87.56% using a CNN model. However, this approach had limitations in identifying subtle variations in code-switched language.

The second method involved training the dataset on Bi-LSTM and LSTM networks using MFCC coefficients. Various variations were applied, including the addition of delta and double delta features and training with and without the energy coefficient. It was found that incorporating the energy coefficient significantly improved performance. The best accuracy of 92.99% was achieved using MFCC + delta and double delta features with the energy coefficient.

The third method employed PLP coefficients on an LSTM network, resulting in an accuracy of 74%. While this approach demonstrated satisfactory results, further enhancements can be made.

In summary, this research has provided valuable insights into code-switch detection. The findings highlight the importance of feature selection and data augmentation techniques in achieving higher accuracy. There is potential for improvement through data augmentation techniques such as Pitch Shifting, Time Shift, and other methods to increase the variability of available data. Future research can focus on refining the proposed approaches and exploring additional augmentation methods to further enhance performance in code-switch detection tasks.

References

[1] **“Code-switching detection using multilingual DNNS”** :E. Yılmaz, H. van den Heuvel and D. van Leeuwen, 2016 IEEE Spoken Language Technology Workshop(SLT), 2016, pp. 610-616

[2] **"Exploiting spectral augmentation for code-switched spoken language identification":** Rangan, Pradeep, Sundeep Teki, and Hemant Misra.

[3] **"Language identification using deep convolutional recurrent neural networks":** Bartz, Christian, Tom Herold, Haojin Yang, and Christoph Meinel.

[4] **"Performance Evaluation of Conventional and Hybrid Feature Extractions Using Multivariate HMM Classifier":** International Journal of Engineering Research and Applications 5, no. 4 (2015): 96-101. Këpuska, Veton Z., and Hussien A. Elharati.

[5] **Speech Recognition—Feature Extraction MFCC & PLP. [online] Medium. Hui, J., 2022 Available at:** https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9.

[6] **"Audio augmentation for speech recognition":** In Sixteenth annual conference of the international speech communication association. 2015. Ko, Tom, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur.

[7] **"Spoken Language Identification Using Deep Learning":** Computational Intelligence and Neuroscience 2021 (2021). Singh, Gundeep, Sahil Sharma, Vijay Kumar, Manjit Kaur, Mohammed Baz, and Mehedi Masud.

[8] **"Performance Evaluation of Conventional and Hybrid Feature Extractions Using Multivariate HMM Classifier":** International Journal of Engineering Research and Applications 5, no. 4 (2015): 96-101. Këpuska, Veton Z., and Hussien A. Elharati.

[9] **“Long short-term memory”:** Neural Computation, vol. 9, no. 8, pp. 1735– 1780, Nov. 1997. S. Hochreiter and J. Schmidhuber.

[10] **"Learning to forget: Continual prediction with LSTM":** Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000. F. A. Gers, J. Schmidhuber, and F. Cummins.

[11] **"Learning precise timing with LSTM recurrent networks":** Journal of Machine Learning Research, vol. 3, pp. F. A. Gers, N. N. Schraudolph, and J. Schmidhube. 115– 143, Mar. 2003.

[12] **"Hybrid speech recognition with deep bidirectional LSTM"**: in Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013, pp. 273–278. A. Graves, N. Jaitly, and A. Mohamed.

[13] **"Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition":** ArXiveprints, Feb. 2014 H. Sak, A. Senior, and F. Beaufays.

[14] **"A study on data augmentation of reverberant speech for robust speech recognition"**: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 5220-5224 T. Ko, V. Peddinti, D. Povey, M. L. Seltzer and S. Khudanpur.

[15] **“Perceptual linear predictive (PLP) analysis of speech, the Journal of the Acoustical Society”:** H. Hermansky.

[16] **"Bidirectional RNN":** Version 6, April 30. Accessed 2022-02-15. https://devopedia.org/bidirectional-rnn Devopedia. 2020.

[17] **“Feature Extraction Methods Proposed for Speech Recognition Are Effective on Road Condition Monitoring Using Smartphone Inertial Sensors.”**

Sensors. 19. 3481. 10.3390/s19163481. Cabral, Frederico&Fukai, Hidekazu& Tamura, Satoshi. (2019).

[18] **"Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras":** arXiv preprint arXiv:1706.05781 (2017). Choi, Keunwoo, DeokjinJoo, and Juho Kim.

[19] **"A review into deep learning techniques for spoken language identification. Multimed Tools Appl":** 81, 32593–32624 (2022). Thukroo, I.A., Bashir, R. & Giri, K.J.

[20] **"Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching":** Rallabandi, SaiKrishna and Sitaram, Sunayana and Black, Alan W (2018).

[21] **"A Survey of Code-switched Speech and Language Processing":** CoRR vol. abs/1904.00784, 2019. Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W. Black.