Is Attention always needed? A Case Study on Language Identification from Speech

Atanu Mandal\textsuperscript{1*}, Santanu Pal\textsuperscript{2}, Indranil Dutta\textsuperscript{3}, Mahidas Bhattacharya\textsuperscript{3}, and Sudip Kumar Naskar\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Engineering, Jadavpur University, Kolkata
\textsuperscript{2}Wipro AI Lab, Wipro India Limited, Bengaluru
\textsuperscript{3}School of Languages and Linguistics, Jadavpur University, Kolkata
\*Corresponding author. E-mails: atanumandal0491@gmail.com, santanu.pal2@wipro.com, indranildutta.lnl@jadavpuruniversity.in, languagemahib@gmail.com, sudipkumar.naskar@jadavpuruniversity.in

Competing interests: The author(s) declare none

(Received xx xxx xxx; revised xx xxx xxx; accepted xx xxx xxx)

Abstract

Language Identification (LID) is a crucial preliminary process in the field of Automatic Speech Recognition (ASR) that involves the identification of a spoken language from audio samples. Contemporary systems that can process speech in multiple languages require users to expressly designate one or more languages prior to utilization. The LID task assumes a significant role in scenarios where ASR systems are unable to comprehend the spoken language in multilingual settings, leading to unsuccessful speech recognition outcomes. The present study introduces convolutional recurrent neural network (CRNN) based LID, designed to operate on the Mel-frequency Cepstral Coefficient (MFCC) characteristics of audio samples. Furthermore, we replicate certain state-of-the-art methodologies, specifically the Convolutional Neural Network (CNN) and Attention-based Convolutional Recurrent Neural Network (CRNN with attention), and conduct a comparative analysis with our CRNN-based approach. We conducted comprehensive evaluations on thirteen distinct Indian languages and our model resulted in over 98% classification accuracy. The LID model exhibits high-performance levels ranging from 97% to 100% for languages that are linguistically similar. The proposed LID model exhibits a high degree of extensibility to additional languages and demonstrates a strong resistance to noise, achieving 91.2% accuracy in a noisy setting when applied to a European Language (EU) dataset.

1. Introduction

In the era of the Internet of Things, smart and intelligent assistants (e.g., Alexa\textsuperscript{a}, Siri\textsuperscript{b}, Cortana\textsuperscript{c}, Google Assistant\textsuperscript{d}, etc.) can interact with humans with some default language settings (mostly in English) and these smart assistants rely heavily on ASR. The motivation for our work stems from the inadequacy of virtual assistants in providing support in multilingual settings. In order to enhance the durability of intelligent assistants, LID can be implemented to enable automatic recognition of the speaker’s language, thereby facilitating appropriate language setting adjustments. Psychological behaviour exhibits that Humans have an inherent capability to determine

\textsuperscript{a}https://developer.amazon.com/en-US/alexa/alexa-voice-service
\textsuperscript{b}https://www.apple.com/in/siri/
\textsuperscript{c}https://www.microsoft.com/en-in/windows/cortana
\textsuperscript{d}https://assistant.google.com/
the language of a statement nearly instantly. Automatic LID seeks to classify a speaker’s language usage from their speech utterances.

We focus our study of LID on Indian Languages since India is the world’s second most populated and seventh largest country in landmass and a linguistically diverse country. Currently, India has 28 states and 8 Union Territories, where each state and Union Territory has its own language, but none of the languages is recognised as the national language of the country. Only, English and Hindi are used as official languages according to the Constitution of India Part XVII Chapter 1 Article 343. Currently, the Eighth Schedule of the Constitution consists of 22 languages. Table I describes the recognised 22 languages according to the Eighth Schedule of the Constitution of India, as of 1 December 2007.

Table 1. : List of languages as per the Eighth Schedule of the Constitution of India, as of 1 December 2007 with their language family & states spoken in.

| Sl. No. | Language | Family            | Spoken in                                                                 |
|--------|----------|-------------------|---------------------------------------------------------------------------|
| 1      | Assamese | Indo-Aryan        | Assam                                                                     |
| 2      | Bengali  | Indo-Aryan        | Assam, Jharkhand, Tripura, West Bengal                                   |
| 3      | Bodo     | Sino-Tibetan      | Assam                                                                     |
| 4      | Dogri    | Indo-Aryan        | Jammu & Kashmir                                                          |
| 5      | Gujarati | Indo-Aryan        | Gujrat, Dadra & Nagar Haveli & Daman & Diu                               |
| 6      | Hindi    | Indo-Aryan        | Andaman & Nicobar Islands, Bihar, Chhattisgarh, Dadra & Nagar Haveli & Daman & Diu, Delhi, Haryana, Himachal Pradesh, Jammu & Kashmir, Jharkhand, Ladakh, Madhya Pradesh, Mizoram, Rajasthan, Uttar Pradesh, Uttarakhand |
| 7      | Kannada  | Dravidian         | Karnataka                                                                 |
| 8      | Kashmiri | Indo-Aryan        | Jammu & Kashmir                                                          |
| 9      | Konkani  | Indo-Aryan        | Dadra & Nagar Haveli & Daman & Diu, Goa                                   |
| 10     | Maithili | Indo-Aryan        | Jharkhand                                                                 |
| 11     | Malayalam| Dravidian         | Kerala, Lakshadweep, Puducherry                                          |
| 12     | Manipuri | Sino-Tibetan      | Manipur                                                                   |
| 13     | Marathi  | Indo-Aryan        | Dadra & Nagar Haveli & Daman & Diu, Goa, Maharashtra                      |
| 14     | Nepali   | Indo-Aryan        | Sikkim, West Bengal                                                       |
| 15     | Odia     | Indo-Aryan        | Jharkhand, Odisha                                                         |
| 16     | Punjabi  | Indo-Aryan        | Delhi, Haryana, Punjab                                                   |
| 17     | Sanskrit | Indo-Aryan        | Himachal Pradesh                                                          |
| 18     | Santali  | Austroasiatic     | Jharkhand                                                                 |
| 19     | Sindhi   | Indo-Aryan        | Rajasthan                                                                  |
| 20     | Tamil    | Dravidian         | Tamil Nadu                                                                 |
| 21     | Telugu   | Dravidian         | Andhra Pradesh, Puducherry, Telangana                                    |
| 22     | Urdu     | Indo-Aryan        | Bihar, Delhi, Jammu & Kashmir, Jharkhand, Telangana, Uttar Pradesh        |

https://www.mea.gov.in/Images/pdf1/Part17.pdf
Most of the Indian languages originated from the Indo-Aryan and Dravidian language families. It can be seen from Table 1 that different languages are spoken in different states, however, languages do not obey geographical boundaries. Therefore, many of these languages, particularly in the neighbouring regions, have multiple dialects which are amalgamations of two or more languages.

Such enormous linguistic diversity makes it difficult for citizens to communicate in different parts of the country. Bilingualism and multilingualism are the norms in India. In this context, a LID system becomes a crucial component for any speech-based smart assistant. The biggest challenge and hence an area of active innovation for the Indian language is the reality that most of these languages are under-resourced.

Every spoken language has its underlying lexical, speaker, channel, environment, and other variations. The likely differences among various spoken languages are in their phoneme inventories, frequency of occurrence of the phonemes, acoustics, the span of the sound units in different languages, and intonation patterns at higher levels. The overlap between the phoneme set of two or more familial languages makes it a challenge for recognition. The low-resource status of these languages makes the training of machine learning models doubly difficult. The idea behind our methodology is interesting on account of the aforementioned limitations. Our methodology involves forecasting the accurate spoken language, irrespective of the limitations mentioned earlier.

CNN has been heavily utilized by Natural Language Processing (NLP) researchers from the very beginning due to their efficient use of local features. While Recurrent Neural Networks (RNNs) have been shown to be effective in a variety of NLP tasks in the past, recent work with Attention-based methods have outperformed all previous models and architectures because of their ability to capture global interactions. Yamada et al. (2020) were able to achieve better results than BERT (Devlin et al. 2019), SpanBERT (Joshi et al. 2020), XLNet (Yang et al. 2019), and ALBERT (Zhenzong et al. 2020) using their Attention-based methods in the Question-Answering domain. Researchers (Takase et al. 2021; Gu et al. 2019; Chen et al. 2020) have employed Attention-based methods to achieve state-of-the-art (SOTA) performance in Machine Translation. Transformers (Vaswani et al. 2017), which utilize a self-attention mechanism, have found extensive application in almost all fields of NLP such as language modelling, text classification, topic modelling, emotion classification, sentiment analysis, etc., and produced SOTA performance.

In this work, we present LID for Indian languages using a combination of CNN, RNN, and Attention-based methods. Our LID methods cover 13 Indian languages Additionally, our method is language agnostic. The main contributions of this work can be summarized as follows:

- We carried out exhaustive experiments using CNN, CRNN, and attention-based CRNN for the LID task on 13 Indian languages and achieved state-of-the-art results.
- The model exhibits exceptional performance in languages that are part of the same language family, as well as in diverse language sets under both normal and noisy conditions.
- We empirically proved that CRNN framework achieves better or similar results compared to CRNN with Attention framework although CRNN without Attention requires less computational overhead.

2. Related Works

Extraction of language-dependent features for example prosody and phonemes was widely used to classify spoken languages (Zissman 1996; Martinez et al. 2020; Ferrer et al. 2010). Following

\footnote{The study was limited to the number of Indian languages for which datasets were available}
the success of speaker verification systems, identity vectors (i-vectors) have also been used as features in various classification frameworks. Use of i-vectors requires significant domain knowledge (Dehak et al. 2011; Martinez et al. 2020). In recent trends, researchers rely on neural networks for feature extraction and classification (Lopez-Moreno et al. 2014; Ganapathy et al. 2014). Researcher Revay and Teschke (2019) used the ResNet50 (He et al. 2016) framework for classifying languages by generating the log-Mel spectra for each raw audio. The framework uses a cyclic learning rate where the learning rate increases and then decreases linearly. The maximum learning rate for a cycle is set by finding the optimal learning rate using fastai (Howard et al. 2020).

Researcher Gazeau et al. (2018) established the use of a Neural Network, Support Vector Machine, and Hidden Markov Model (HMM) to identify different languages. Hidden Markov models convert speech into a sequence of vectors and are used to capture temporal features in speech. Established LID systems (Dehak et al. 2011; Martinez et al. 2020; Plchot et al. 2016; Zazo et al. 2016) are based on identity vector (i-vectors) representations for language processing tasks. In Dehak et al. (2011), i-vectors are used as data representations for a speaker verification task and fed to the classifier as the input. Dehak et al. (2011) applied Support Vector Machines (SVM) with cosine kernels as the classifier, while Martinez et al. (2020) used logistic regression for the actual classification task. Recent years have found the use of feature extraction with neural networks, particularly with Long Short Term Memory (LSTM) (Zazo et al. 2016; Gelly et al. 2016; Lozano-Diez et al. 2015). These neural networks produce better accuracy while being simpler in design compared to the conventional LID methods (Dehak et al. 2011; Martinez et al. 2020; Plchot et al. 2016). Recent trends in developing LID systems are mainly focused on different forms of LSTMs with DNNs. Plchot et al. (2016) used a 3-layered CNN where i-vectors were the input layer and softmax activation function was the output layer. Zazo et al. (2016) used MFCC with Shifted Delta Coefficient features as information to a unidirectional layer that is directly connected to a softmax classifier. Gelly et al. (2016) used audio transformed to Perceptual Linear Prediction (PLP) coefficients and their 1st and 2nd order derivatives as information for a Bidirectional LSTM in forward and backward directions. The forward and backward sequences generated from the Bidirectional LSTM were joined together and used to classify the language of the input samples. Lozano-Diez et al. (2015) used CNNs for their LID system. They transformed the input data into an image containing MFCCs with Shifted Delta Coefficient features. The image represents the time domain for the x-axis and frequency bins for the y-axis.

Lozano-Diez et al. (2015) used CNN as the feature extractor for the identity vectors. They achieved better performance when combining both the CNN features and identity vectors. Revay and Teschke (2019) used ResNet (He et al. 2016) framework for language classification by generating spectrograms of each audio. Cyclic Learning (Smith 2018) was used where the learning rate increases and decreases linearly. Venkatesan et al. (2018) utilised MFCCs to infer aspects of speech signals from Kannada, Hindi, Tamil, and Telugu. They obtained an accuracy of 76% and 73% using Support Vector Machines and Decision Tree classifiers, respectively, on 5 hours of training data. Mukherjee et al. (2019) used CNNs for language identification in German, Spanish, and English. They used Filter Banks to extract features from frequency domain representations of the signal. Aarti et al. (2017) experimented with several auditory features in order to determine the optimal feature set for a classifier to detect Indian Spoken Language. Sisodia et al. (2020) evaluated Ensemble Learning models for classifying spoken languages such as German, Dutch, English, French, and Portuguese. Bagging, Adaboosting, random forests, gradient boosting, and additional trees were used in their ensemble learning models.

Heracleous et al. (2018) presented a comparative study of Deep Neural Networks (DNN) and CNNs for Spoken LID, with Support Vector Machines (SVM) as the baseline. They also presented the performance of the fusion of the mentioned methods. The NIST 2015 i-vector Machine Learning Challenge dataset was used to assess the system’s performance with the goal of detecting 50 in-set languages. Bartz et al. (2017) tackled the problem of Language Identification in the image domain rather than the typical acoustic domain. A hybrid CRNN is employed for this,
which acts on spectrogram images of the provided audio clips. Draghichi et al. (2020) tried to solve the task of Language Identification while using Mel-spectrogram images as input features. This strategy was employed in CNNs and CRNN in terms of performance. This work is characterized by a modified training strategy that provides equal class distribution and efficient memory utilisation. Ganapathy et al. (2014) reported how they used bottleneck features from a CNN for the LID task. Bottleneck features were used in conjunction with conventional acoustic features, and performance was evaluated. Experiments revealed that when a system with bottleneck features is compared to a system without them, average relative improvements of up to 25% are achieved. Zazo et al. (2016) proposed an open-source, end-to-end, LSTM-RNN system that outperforms a more recent reference i-vector system by up to 26% when both are tested on a subset of the NIST Language Recognition Evaluation with 8 target languages.

Our research differs from the previous works on LID in the following aspects:

• Comparison of performance of CNN, CRNN, as well as CRNN with Attention.
• Extensive experiments with our proposed model show its applicability both for close language as well as noisy speech scenarios.

3. Model Framework

Our proposed framework consists of three models.

• CNN-based framework
• CRNN-based framework
• CRNN with Attention-based framework

We made use of the capacity of CNNs to capture spatial information to identify languages from audio samples. In a CNN-based framework, our network uses four convolution layers, where each layer is followed by the ReLU (Nair and Geoffrey 2010) activation function and max pooling with a stride of 3 and a pool size of 3. The kernel sizes and the number of filters for each convolution layer are (3, 512), (3, 512), (3, 256), and (3, 128), respectively.

Figure[] provides a schematic overview of the framework. The CRNN framework passes the output of the Convolutional Module to a Bi-Directional LSTM consisting of a single LSTM with 256 output units. The LSTM’s activation function is \( \text{tanh} \), and its recurrent activation is \( \text{sigmoid} \). The Attention Mechanism used in our framework is based on Hierarchical Attention Networks (Yang et al. 2016). In the Attention Mechanism, contexts of features are summarized with a bidirectional LSTM by going forward and backwards.

\[
\begin{align*}
\hat{a}_n &= \overrightarrow{\text{LSTM}}(a_n), \ n \in [1, L] \\
\tilde{a}_n &= \overleftarrow{\text{LSTM}}(a_n), \ n \in [L, 1] \\
a_i &= \left[ \hat{a}_n, \tilde{a}_n \right]
\end{align*}
\]

In equation [], \( L \) is the number of audio specimens. \( a_n \) is the input sequence for the LSTM network. \( \hat{a}_n \) and \( \tilde{a}_n \) provide the learned vectors from LSTM forward direction and backward direction, respectively. The vector, \( a_i \), builds the base for the attention mechanism. The goal of the attention mechanism is to learn the model through training with randomly initialized weights and biases. The layer also ensures with the \( \text{tanh} \) function that the network does not stall. The function keeps the input values between -1 and 1 and maps zeros to near-zero values. The layer with \( \text{tanh} \) function is again multiplied by trainable context vector \( u_i \). The trainable context vector refers to a vector learned during the training process and used as a fixed-length representation of the entire input document. In our framework, the attention mechanism is used to compute a weighted sum of
Figure 1: The figure presents our CRNN framework consisting of a Convolution Block and LSTM Block denoted in different blocks. The convolution block extracts feature from the input audio. The output of the final convolution layer is provided to the Bi-Directional LSTM network as the input which is further connected to a Linear Layer with softmax classifier.

the sequences for each speech utterance, where the weights are learned based on the relevance of each sequence to the speech utterances. This produces a fixed-length vector for each utterance that captures the most salient information in the sequences. The context weight vector $u_i$ is randomly initiated and jointly learned during the training process. Improved vectors are represented by $a_i'$ as shown in equation (2).

$$a_i' = \tanh(a_i \cdot w_i + b_i) \cdot u_i$$

Context vectors are finally calculated by providing a weight to each $W_i$ by dividing the exponential values of the previously generated vectors with the summation of all exponential values.
of previously generated vectors as shown in equation 3. To avoid division by zero, an epsilon is added to the denominator.

\[
W_i = \frac{\exp(a_i')}{\sum_i \exp(a_i') + \epsilon}
\]  

(3)

The sum of these importance weights concatenated with the previously calculated context vectors is fed to a linear layer with 13 output units serving as a classifier for the 13 languages.

Figure 2: Schematic diagram of the Attention Module.

Figure 2 presents the schematic diagram of the Attention Module where \(a_i\) is the input to the module and output of the Bi-Directional LSTM layers.

4. Experiments

4.1 Feature Extraction

For feature extraction of spoken utterances, we used MFCCs. For calculating MFCCs we used pre-emphasis, frame size represented as \(f\text{\_size}\), frame stride represented as \(f\text{\_stride}\), N-point Fast Fourier transform represented as \(N\text{\_FFT}\), low-frequency mel represented as \(lf\), the number of filters represented as \(n\text{\_filt}\), the number of cepstral coefficients represented as \(n\text{\_coef}\) and cepstral lifter represented \(lifter\) of values 0.97, 0.025 (25ms), 0.015 (15ms overlapping), 512, 0, 40, 13, and 22, respectively. We used a frame size of 25 ms as typically frame sizes in the speech processing domain use 20ms to 40ms with 50% (in our case 15ms) overlapping between consecutive frames.

\[
h_f = 2595 \times \log_{10}(1 + \frac{0.5 \times sr}{700})
\]  

(4)

We used low-frequency mel (lf) as 0 and high-frequency mel (hf) is calculated using the equation 4. If and hf are used to generate the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies.

\[
\text{emphasized\_signal} = [\text{sig}[0], \text{sig}[1:] - \text{pre\_emphasis} \times \text{sig}[: -1]]
\]  

(5)

As shown in equation 5 emphasized signal is calculated by using a pre-emphasis filter applied on the signal (sig) using the first-order filter. The number of frames is calculated by taking the ceiling value of the division of the absolute value of the difference between signal length (sig\_len) and product of filter size (f\_size) and sample rate (sr) with the product of frame stride (f\_stride) and sample rate (sr) as shown in equation 6. Signal length is the length of emphasized\_signal calculated in equation 5.

\[
n\_frames = \lceil \frac{\text{sig\_len} - (f\text{\_size} \times sr)}{(f\text{\_stride} \times sr)} \rceil
\]  

(6)

Using equation 7 pad\_signal is generated from concatenation of emphasized\_signal and zero value array of dimension \((\text{pad\_signal\_length} - \text{signal\_length}) \times 1\), where, pad\_signal\_length is
calculated by $n_{frames} \times (f_{stride} \times sr) + (f_{size} \times sr)$.

$$pad\_signal = [\text{emphasized}\_signal; 0]^{(n_{frames} \times (f_{stride} \times sr) + (f_{size} \times sr) - \text{sig\_len}) 	imes 1}$$  \hspace{1cm} (7)

Frames are calculated as shown in equation 8 from the pad_signal elements where elements are the addition of an array of positive natural numbers from 0 to $f_{size} \times sr$ repeated $n_{frames}$ and the transpose of the array of size of num_frames where each element is the difference of $(f_{stride} \times sr)$.

$$frames = pad\_signal[(\{x \in Z^+: 0 < x < (f_{size} \times sr)\}_{n=0}^{n_{frames}},1) + ((\{r: r = (f_{stride} \times sr) \times (i - 1), i \in \{0, \ldots, n_{frames} \times (f_{stride} \times sr)\}\}_{n=0}^{(f_{size} \times sr),1})^T]$$  \hspace{1cm} (8)

Power frames shown in equation 9 are calculated as the square of the absolute value of the Discrete Fourier Transform (DFT) of the product of hamming window and frames of each element with NFFT.

$$pf = \frac{|DFT((frames \times (0.54 - (\sum_{N=0}^{(f_{size} \times sr) - 1} 0.46 \times \cos(\frac{2\pi N}{(f_{size} \times sr) - 1}))) \times \text{NFFT})|^2}{\text{NFFT}}$$  \hspace{1cm} (9)

$$mel\_points = \{r: r = lf + \frac{hf - lf}{(n_{filt} + 2) - 1} \times i, i \in \{lf, \ldots, hf\}\}$$  \hspace{1cm} (10)

Mel points are the array where elements are calculated as shown in the equation 10 where i is the values belonging from lf to hf.

$$bins = \left\lfloor \frac{(\text{NFFT} + 1) \times (700 \times (10^{\frac{\text{mel\_points}}{2595}} - 1))}{\text{sample\_rate}} \right\rfloor$$  \hspace{1cm} (11)

From equation 11 bins are calculated where the floor value of the elements are taken which is the product of hertz points and NFFT + 1 divided by the sample rate. Hertz points are calculated by multiplying 700 by subtraction of 1 from 10 power of $\frac{\text{mel\_points}}{2595}$.

$$f_{bank}_{m}(k) = \begin{cases} 0 & k < bins(m - 1) \\ \frac{k - bins(m - 1)}{bins(m) - bins(m - 1)} & bins(m - 1) \leq k \leq bins(m) \\ \frac{bins(m) - k}{bins(m + 1) - bins(m)} & bins(m) \leq k \leq bins(m + 1) \\ 0 & k > bins(m + 1) \end{cases}$$  \hspace{1cm} (12)

Bins calculated from equation 11 are used to calculate filter banks shown in equation 12. Each filter in the filter bank is triangular, with a response of 1 at the central frequency and a linear drop to 0 till it meets the central frequencies of the two adjacent filters, where the response is 0.

Finally, mfcc is calculated shown in equation 13 by decorrelating the filter bank coefficients using Discrete Cosine Transform (DCT) to get a compressed representation of the filter banks. Sinusoidal liftering is applied to the mfcc to de-emphasize higher mfccs which improves classification in noisy signals.

$$mfcc = DCT(20 \log_{10}(pf \cdot f_{bank}^T)) \times \left[1 + \frac{\text{lifter}}{2} \sin\left(\frac{\pi \odot n: n \in Z^+, n \leq n_{coefficient}}{\text{lifter}}\right)\right]$$  \hspace{1cm} (13)

MFCCs features of shape $(1000, 13)$ generated from equation 13 is provided as input to the neural network which expects the same dimension followed by convolution layers as mentioned in section 3. Raw speech signal cannot be provided input to the framework as it contains lots of noise data therefore extracting features from the speech signal and using it as input to the model will produce better performance than directly considering raw speech signal as input. Our
motivation to use MFCC features as the feature count is small enough to force us to learn the information of the sample. Parameters are related to the amplitude of frequencies and provide us with frequency channels to analyze the speech specimen.

### 4.2 Data

#### 4.2.1 Benchmark Data

The Indian language (IL) dataset was acquired from the Indian Institute of Technology, Madras. The dataset includes 13 widely used Indian languages. Table 2 presents the statistics of this dataset which we used for our experiments.

Table 2. : Statistics of the Indian Language (IN) Dataset

| Language | Label | Gender | Samples | Total Samples | Average Duration (in seconds) |
|----------|-------|--------|---------|---------------|-----------------------------|
| Assamese | as    | F      | 8,713   | 17,654        | 5.587                       |
|          |       | M      | 8,941   |               |                             |
| Bengali  | bn    | F      | 3,253   | 9,440         | 5.743                       |
|          |       | M      | 6,187   |               |                             |
| Bodo     | bd    | F      | 571     | 571           | 25.219                      |
| Gujarati | gu    | F      | 2,396   | 5,684         | 13.459                      |
|          |       | M      | 3,288   |               |                             |
| Hindi    | hi    | F      | 2,318   | 4,636         | 8.029                       |
|          |       | M      | 2,318   |               |                             |
| Kannada  | kn    | F      | 1,289   | 2,578         | 10.264                      |
|          |       | M      | 1,289   |               |                             |
| Malayalam| ml    | F      | 5,650   | 11,300        | 5.699                       |
|          |       | M      | 5,650   |               |                             |
| Manipuri | mn    | F      | 9,487   | 17,917        | 4.169                       |
|          |       | M      | 8,430   |               |                             |
| Marathi  | mr    | F      | 2,448   | 2,448         | 7.059                       |
| Odia     | or    | F      | 3,578   | 7,151         | 4.4                         |
|          |       | M      | 3,573   |               |                             |
| Rajasthani| rj   | F      | 4,346   | 9,125         | 7.914                       |
|          |       | M      | 4,779   |               |                             |
| Tamil    | ta    | F      | 3,243   | 6,960         | 10.516                      |
|          |       | M      | 3,717   |               |                             |
| Telugu   | te    | F      | 4,043   | 6,524         | 15.395                      |
|          |       | M      | 2,481   |               |                             |

[1]https://www.iitm.ac.in/donlab/tts/database.php
4.2.2 Experimental Data
In the past two decades, the development of LID methods has been largely fostered through NIST Language Evaluations (LREs). As a result, the most popular benchmark for evaluating new LID models and methods is the NIST LRE evaluation dataset (Sadjadi et al. 2018). The NIST LREs dataset mostly contains narrow-band telephone speech. Datasets are typically distributed by the Linguistic Data Consortium (LDC) and cost thousands of dollars. For example, the standard Kaldi (Povey et al. 2011) recipe for LRE072 relies on 18 LDC SLR datasets that cost $15000 (approx) to LDC non-members. This makes it difficult for new research groups to enter the academic field of LID. Furthermore, the NIST LRE evaluations focus mostly on telephone speech.

As the NIST LRE dataset is not freely available we used the EU Dataset (Bartz et al. 2017) which is open source. The (EU) dataset contains YouTube News data for 4 major European languages – English (en), French (fr), German (de) and Spanish (es). Statistics of the dataset are given in Table 3.

| Language | Label | Total Samples | Average Duration (in seconds) |
|----------|-------|---------------|------------------------------|
| English  | en    | 43,269        | 684.264                      |
| French   | fr    | 67,689        | 492.219                      |
| German   | de    | 48,454        | 1,152.916                   |
| Spanish  | es    | 57,869        | 798.169                      |

4.3 Environment
We implemented our framework using Tensorflow (Abadi et al. 2016) backend. We split the Indian language dataset into training, validation, and testing set, containing 80%, 10%, and 10% of the data, respectively, for each language and gender.

For regularization, we apply dropout (Srivastava et al. 2014) after the Max-Pooling layer and Bi-Directional LSTM layer. We use the rate of 0.1. A $l_2$ regularization with $10^{-6}$ weight is also added to all the trainable weights in the network. We train the model with Adam (Kingma et al. 2014) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$ and learning rate schedule (Vaswani et al. 2017), with 4k warm-up steps and peak learning rate of $0.05/\sqrt{d}$ where $d$ is 128. A batch size of 64 with “Sparse Categorical Crossentropy” as the loss function was used.

4.4 Result on Indian Language Dataset
The proposed framework was assessed against Kulkarni et al. (2022) using identical datasets. Both CRNN and CRNN with Attention exhibited superior performance compared to the results reported by Kulkarni et al. (2022), as shown in Table 4. They used 6 Linear layers where units are 256, 256, 128, 64, 32, and 13, respectively in the CNN framework, whereas the DNN framework uses 3 LSTM layers having units 256, 256, and 128, respectively followed by dropout layer followed by 3 Time Distributed layer followed by a Linear layer of 13 as units.

We evaluated system performance using the following evaluation metrics – Recall (TPR), Precision (PPV), f1-score, and Accuracy. Since one of our major objectives was to measure the accessibility of the network to new languages, we introduced Data Balancing of training data for each class, as the number of samples available for each class may vary drastically. This is the
Table 4: Comparative evaluation results (in terms of Accuracy) of our model and the model of Kulkarni et al. (2022) on the Indian Language dataset

| Model                  | Accuracy |
|------------------------|----------|
| DNN Kulkarni et al 2022| 0.9834   |
| RNN Kulkarni et al 2022| 0.9843   |
| CNN                    | 0.983    |
| CRNN                   | 0.987    |
| CRNN with Attention    | 0.987    |

Table 5: Experimental Results for Indian Languages

| Language | CRNN with Attention | CRNN | CNN |
|----------|---------------------|------|-----|
|          | PPV | TPR | f1  | Score | Accuracy | PPV | TPR | f1  | Score | Accuracy | PPV | TPR | f1  | Score | Accuracy |
|          |     |     |     |       |          |     |     |     |       |          |     |     |     |       |          |
| as       | 0.989 | 0.998 | 0.993 | 0.995 | 0.990 | 0.992 | 0.991 | 0.998 | 0.999 | 0.993 | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 |
| bn       | 1    | 0.9 | 0.948 | 1    | 0.904 | 0.949 | 1    | 0.888 | 0.941 | 1    | 0.888 | 0.941 | 1    | 0.888 | 0.941 |
| bd       | 0.966 | 1    | 0.983 | 0.966 | 1    | 0.983 | 0.966 | 1    | 0.983 | 1    | 0.983 | 1    | 0.983 | 1    | 0.983 |
| gu       | 0.997 | 0.997 | 0.997 | 0.951 | 0.998 | 0.974 | 0.996 | 0.991 | 0.984 | 0.951 | 0.991 | 0.984 | 0.951 | 0.991 | 0.984 |
| hi       | 0.987 | 0.991 | 0.989 | 0.961 | 0.991 | 0.989 | 0.961 | 0.974 | 0.983 | 0.961 | 0.974 | 0.983 | 0.961 | 0.974 | 0.983 |
| kn       | 0.977 | 0.996 | 0.987 | 0.996 | 0.996 | 0.996 | 0.973 | 0.992 | 0.983 | 0.996 | 0.992 | 0.983 | 0.973 | 0.992 | 0.983 |
| ml       | 0.996 | 0.988 | 0.992 | 0.997 | 0.99 | 0.994 | 0.99 | 0.993 | 0.991 | 0.99 | 0.993 | 0.991 | 0.99 | 0.993 | 0.991 |
| nn       | 0.987 | 0.999 | 0.993 | 0.973 | 0.999 | 0.986 | 0.972 | 0.998 | 0.985 | 0.972 | 0.998 | 0.985 | 0.972 | 0.998 | 0.985 |
| mr       | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |
| or       | 0    | 1    | 1    | 1    | 0.999 | 0.999 | 0.996 | 0.999 | 0.992 | 0.996 | 0.999 | 0.992 | 0.996 | 0.999 | 0.992 |
| rj       | 0.999 | 0.993 | 0.996 | 0.995 | 1    | 0.977 | 0.996 | 0.992 | 0.991 | 0.996 | 0.992 | 0.991 | 0.996 | 0.992 | 0.991 |
| ta       | 0.929 | 0.991 | 0.959 | 0.975 | 0.997 | 0.986 | 0.946 | 0.989 | 0.967 | 0.946 | 0.989 | 0.967 | 0.946 | 0.989 | 0.967 |
| te       | 0.979 | 0.998 | 0.989 | 0.982 | 1    | 0.991 | 0.975 | 0.998 | 0.986 | 0.975 | 0.998 | 0.986 | 0.975 | 0.998 | 0.986 |

Table 6: Confusion matrix for CRNN with Attention framework

| Actual | Predicted | as | bn | bd | gu | hi | kn | ml | nn | mr | or | rj | ta | te |
|--------|-----------|---|----|----|----|----|----|----|----|----|----|----|----|----|
| as     | 1762      | 0 | 0  | 0  | 0  | 0  | 4  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| bn     | 10        | 0 | 0  | 0  | 0  | 0  | 10 | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| bd     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| gu     | 1         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| hi     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| kn     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| ml     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| nn     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| mr     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| or     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| rj     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| ta     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| te     | 0         | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

PPV, TPR, f1-score, and Accuracy scores are reported in Table 5 for the three frameworks - CNN, CRNN, and CRNN with Attention. From Table 5 it is clearly visible that both CRNN framework and CRNN with Attention provide competitive results of 0.987 accuracy. Table 6, Table 7, and Table 8 shows the confusion matrix for CNN, CRNN, and CRNN with Attention.
Table 7. : Confusion matrix for CRNN

| Actual | Predicted | PPV | TPR | F1 Score |
|--------|-----------|-----|-----|----------|
| as     | 1776      | 0   | 0   | 0.995    |
| bn     | 1746      | 0   | 0   | 0.989    |
| bd     | 0         | 0   | 1710| 0.949    |
| gu     | 0         | 0   | 0   | 0.989    |
| hi     | 0         | 0   | 0   | 0.982    |
| kn     | 0         | 1   | 0   | 0.975    |
| ml     | 0         | 0   | 0   | 0.995    |
| mn     | 0         | 0   | 0   | 0.995    |
| mr     | 0         | 0   | 0   | 0.995    |
| or     | 0         | 0   | 0   | 0.995    |
| rj     | 0         | 0   | 0   | 0.995    |
| te     | 0         | 0   | 0   | 0.995    |

Table 8. : Confusion matrix for CNN

| Actual | Predicted | PPV | TPR | F1 Score |
|--------|-----------|-----|-----|----------|
| as     | 1744      | 0   | 0   | 0.991    |
| bn     | 1744      | 0   | 0   | 0.988    |
| bd     | 0         | 0   | 100 | 0.988    |
| gu     | 0         | 0   | 0   | 0.988    |
| hi     | 0         | 0   | 2   | 0.988    |
| kn     | 0         | 0   | 0   | 0.988    |
| ml     | 0         | 0   | 0   | 0.988    |
| mn     | 0         | 0   | 0   | 0.988    |
| mr     | 0         | 0   | 0   | 0.988    |
| or     | 0         | 0   | 0   | 0.988    |
| rj     | 0         | 0   | 0   | 0.988    |
| te     | 0         | 0   | 0   | 0.988    |

From Table 6, Table 7, and Table 8 it can be observed that Assamese gets confused with Manipuri; Bengali gets confused with Assamese, Manipuri, Tamil, and Telugu; and Hindi gets confused with Malayalam.

Assamese and Bengali have originated from the same language family and they share approximately the same phoneme set. However, Bengali and Tamil are from different language families but share a similar phoneme set. For example, in Bengali *cigar* is *churut* and *star* is *nakshatra* while *cigar* in Tamil is *charuttu* and *star* in Tamil is *natsattira*, which is quite similar. Similarly, Manipuri and Assamese share similar phonemes. On close study, we observed that Hindi and Malayalam have also similar phoneme sets as both languages borrowed most of the vocabulary from Sanskrit. For example, ‘arrogant’ is *Ahankar* in Hindi and *Ahankaram* in Malayalam. Similarly, *Sathyu* or commonly spoken as *Satya* in Hindi means ‘Truth’, which is *Sathyam* in Malayalam. Also, the word *Sundar* in Hindi is *Sundaram* in Malayalam, which means ‘beautiful’.

Table 9. : Most common errors

| Assamese | Manipuri |
|-----------|----------|
| Bengali   | Assamese |
| Bengali   | Manipuri |
| Bengali   | Tamil    |
| Hindi     | Malayalam |
4.5 Result on same language families on Indian Language Dataset

A deeper study into these 13 Indian languages led us to define five clusters of languages based on their phonetic similarity. Cluster internal languages are phonetically similar, close, and geographically contiguous, hence difficult to differentiate.

- **Cluster 1**: Assamese, Bengali, Odia
- **Cluster 2**: Gujarati, Hindi, Marathi, Rajasthani
- **Cluster 3**: Kannada, Malayalam, Tamil, Telugu
- **Cluster 4**: Bodo
- **Cluster 5**: Manipuri

Bodo and Manipuri are phonetically very much distant from any of the rest of the languages, thus they form singleton clusters. We carried out separate experiments for the identification of the cluster internal languages for Cluster 1, 2 and 3, and the experimental results are presented in Table 10.

| Cluster | Language | CRNN with Attention | CRNN | CNN |
|---------|----------|---------------------|------|-----|
|         | PPV      | TPR     | F1 Score | PPV      | TPR     | F1 Score | PPV      | TPR     | F1 Score |
| 1       | as 0.962 | 1       | 0.981    | 0.953    | 1       | 0.976    | 0.953    | 1       | 0.976    |
|         | bn 1     | 0.926   | 0.961    | 1        | 0.907   | 0.951    | 1        | 0.934   | 0.944    |
|         | or 1     | 1       | 1        | 1        | 1       | 1        | 1        | 0.982   | 0.991    |
| 2       | gu 1     | 0.906   | 0.999    | 0.999    | 1        | 0.998   | 0.999    | 1        | 0.996   | 0.998    |
|         | hi 1     | 1       | 1        | 1        | 0.998   | 0.999    | 0.998    | 0.991   | 0.994    |
|         | mr 1     | 1       | 1        | 1        | 0.998   | 0.999    | 0.995    | 0.998   | 0.996    |
|         | rj 0.999 | 1       | 0.999    | 0.999    | 0.996    | 1        | 0.999    | 0.996    | 0.996    |
| 3       | kn 1     | 0.906   | 0.998    | 0.996    | 1        | 1       | 1        | 0.992   | 0.988   | 0.99     |
|         | ml 0.999 | 1       | 0.999    | 0.996    | 1        | 1       | 1        | 0.996   | 0.996   | 0.996    |
|         | ta 1     | 1       | 1        | 1        | 0.996   | 0.997   | 0.996    | 0.997   | 0.996    |
|         | te 1     | 1       | 1        | 1        | 0.995   | 0.997   | 0.996    | 0.997   | 0.996    |

It can be clearly observed from Table 10 that both CRNN framework and CRNN with Attention provide competitive results for every language cluster. For **cluster-1** CRNN framework and CRNN with Attention provides an accuracy of 0.98/0.974, for **cluster-2** 0.999/0.999, and for **cluster-3** 0.999/1, respectively. CNN framework also provides comparable results to the other two frameworks.

Table 11, Table 12, and Table 13 presents the confusion matrix for cluster 1, cluster 2, and cluster 3, respectively. From Table 11, we observed that Bengali gets confused with Assamese and Odia, which is quite expected since these two languages are spoken in neighbouring states and both of them share almost the same phonemes. For example, in Odia, rice is pronounced as bhata whereas in Bengali pronounced as bhat, similarly fish in odia as machha whereas in Bengali it is machh. Both CRNN and CRNN with Attention perform well to discriminate between Bengali and Odia. It can be observed from Table 13 that CNN creates a lot of confusion when discriminating between these four languages. Both CRNN and CRNN with Attention prove to be better at discriminating among these languages. From the results in Table 10, 11, 12 and 13 it is pretty clear that CRNN (Bi-Directional LSTM over CNN) and CRNN with Attention are more effective for Indian language identification and they perform almost at par. Another important observation is that it is harder to classify the languages in cluster 1 than in the other two clusters.
Table 11. : Confusion matrix for Cluster 1

| CRNN and Attention | Predicted | PPV | TPR | f1 Score |
|--------------------|-----------|-----|-----|----------|
| Actual             |           |     |     |          |
| as                 | 1766      | 0   | 0   | 0.962    | 1       | 0.981  |
| bn                 | 70        | 874 | 0   | 1       | 0.926   | 0.964  |
| or                 | 0         | 0   | 716 | 1       | 1       | 1      |

Table 12. : Confusion matrix for Cluster 2

| CRNN and Attention | Predicted | PPV | TPR | f1 Score |
|--------------------|-----------|-----|-----|----------|
| Actual             |           |     |     |          |
| as                 | 1766      | 0   | 0   | 0.953    | 1       | 0.976  |
| bn                 | 88        | 856 | 0   | 1       | 0.907   | 0.951  |
| or                 | 0         | 0   | 716 | 1       | 1       | 1      |

Table 13. : Confusion matrix for Cluster 3

| CRNN and Attention | Predicted | PPV | TPR | f1 Score |
|--------------------|-----------|-----|-----|----------|
| Actual             |           |     |     |          |
| as                 | 1766      | 0   | 0   | 0.953    | 1       | 0.976  |
| bn                 | 87        | 844 | 13  | 1       | 0.894   | 0.944  |
| or                 | 0         | 0   | 716 | 0.982   | 1       | 0.991  |

4.6 Results on European Language

We evaluated our model in two environments – No Noise and White Noise. According to our intuition, in real-life scenarios during prediction of language chances of capturing background noise of chatter and other sounds may happen. For the white noise evaluation setup, we mixed white noise into each test sample which has an audible solid presence but retains the identity of the language.

Table 14 compares the results of our models on the EU dataset with state-of-the-art models presented by Bartz et al. (2017). The model proposed by Bartz et al. (2017) consists of CRNN and uses Google’s Inception-v3 framework (Szegedy et al. 2016). The feature extractor performs convolutional operations on the input image through multiple stages, resulting in the production of a feature map that possesses a height of one. The feature map is partitioned horizontally along the x-axis, and each partition is employed as a temporal unit for the subsequent Bidirectional LSTM network. The network employs a total of five convolutional layers, with each layer being succeeded by the ReLU activation function, Batch Normalization, and 2 × 2 max pooling with a stride of 2. The convolutional layers in question are characterized by their respective kernel sizes and the number of filters, which are as follows: (7 × 7, 16), (5 × 5, 32), (3 × 3, 64), (3 × 3,
Table 14. Comparative evaluation results (in terms of Accuracy) of our model and the model of Bartz et al. (2017) on the EU dataset

| Model                        | No Noise | White Noise |
|------------------------------|----------|-------------|
| CRNN (Bartz et al. 2017)    | 0.91     | 0.63        |
| Inception-v3 CRNN (Bartz et al. 2017) | 0.96 | 0.91        |
| CNN                         | 0.948    | 0.871       |
| CRNN                         | 0.967    | 0.912       |
| CRNN with Attention         | 0.966    | 0.888       |

128), and (3 × 3, 256). The Bidirectional LSTM model comprises a pair of individual LSTM models, each with 256 output units. The concatenation of the two outputs is transformed into a 512-dimensional vector, which is then input into a fully-connected layer. The layer has either 4 or 6 output units, which function as the classifier. They experimented in four different environments – No Noise, White Noise, Cracking Noise, and Background Noise. All our evaluation results are rounded to 3 digits after the decimal point.

The CNN model failed to achieve competitive results; it provided an accuracy of 0.948/0.871 in No Noise/White Noise. In the CRNN framework, our model provides an accuracy of 0.967/0.912 on the No Noise/White Noise scenario outperforming the state-of-the-art results of Bartz et al. (2017). Use of Attention improves over the Inception-v3 CRNN in the No Noise scenario, however, it does not perform well on White Noise.

4.7 Ablation Studies

4.7.1 Convolution Kernel Size
To study the effect of kernel sizes in the convolution layers, we sweep the kernel size with 3, 7, 17, 32, and 65 of the models. We found that performance decreases with larger kernel sizes, as shown in Table 15. On comparing the accuracy up to the second decimal place kernel size 3 performs better than the rest.

Table 15. Ablation study on convolution kernel sizes

| Kernel size | Accuracy |
|-------------|----------|
| 3           | 98.7%    |
| 7           | 98.68%   |
| 17          | 98.65%   |
| 32          | 98.13%   |
| 65          | 93.56%   |

4.7.2 Automatic Class Weight vs Manual Class Weight
Balancing the data using class weights gives better accuracy for CRNN with Attention (98.7%) and CRNN (98.7%), compared to CNN (98.3%) shown in Table 5. We study the efficacy of the frameworks by manually balancing the datasets using 100 samples, 200 samples, and 571 samples...
drawn randomly from the dataset and the results of these experiments are presented in Table 16, Table 17 and Table 18 respectively.

Table 16. Experimental Results for Manually Balancing the Samples for each category to 100.

| Language | CRNN with Attention | CRNN | CNN |
|----------|---------------------|------|-----|
|          | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy |
| as       | 0.766 | 0.72 | 0.742 | 0.839 | 0.94 | 0.887 | 0.617 | 0.58 | 0.598 |
| bn       | 0.875 | 0.7 | 0.778 | 0.957 | 0.9 | 0.928 | 0.816 | 0.8 | 0.808 |
| bd       | 1     | 1 | 1 | 0.962 | 1 | 0.98 | 0.843 | 0.86 | 0.851 |
| gu       | 0.943 | 1 | 0.971 | 1 | 0.98 | 0.99 | 0.731 | 0.76 | 0.745 |
| hi       | 0.959 | 0.94 | 0.95 | 0.957 | 0.9 | 0.928 | 0.778 | 0.7 | 0.737 |
| kn       | 0.961 | 0.98 | 0.97 | 0.94 | 0.94 | 0.94 | 0.725 | 0.74 | 0.733 |
| ml       | 0.958 | 0.92 | 0.939 | 0.923 | 0.96 | 0.941 | 0.774 | 0.82 | 0.796 |
| mn       | 0.878 | 0.72 | 0.791 | 0.935 | 0.86 | 0.896 | 0.691 | 0.76 | 0.724 |
| mr       | 0.906 | 0.96 | 0.932 | 0.98 | 0.96 | 0.97 | 0.857 | 0.84 | 0.848 |
| or       | 0.959 | 0.94 | 0.949 | 0.941 | 1 | 0.971 | 0.811 | 0.86 | 0.835 |
| rj       | 0.782 | 0.86 | 0.819 | 0.894 | 0.84 | 0.866 | 0.605 | 0.52 | 0.559 |
| ta       | 0.677 | 0.88 | 0.765 | 0.898 | 0.88 | 0.889 | 0.564 | 0.62 | 0.590 |
| te       | 0.878 | 0.86 | 0.869 | 0.906 | 0.96 | 0.932 | 0.532 | 0.5 | 0.515 |

Table 17. Experimental Results for Manually Balancing the Samples for each Category to 200.

| Language | CRNN with Attention | CRNN | CNN |
|----------|---------------------|------|-----|
|          | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy |
| as       | 0.941 | 0.96 | 0.95 | 0.98 | 0.94 | 0.969 | 0.8 | 0.88 | 0.838 |
| bn       | 0.909 | 1 | 0.952 | 1 | 0.96 | 0.98 | 0.92 | 0.92 | 0.92 |
| bd       | 0.98 | 0.96 | 0.97 | 0.98 | 0.98 | 0.98 | 0.94 | 0.94 | 0.94 |
| gu       | 1 | 1 | 1 | 1 | 1 | 0.918 | 0.9 | 0.909 |
| hi       | 1 | 0.98 | 0.99 | 1 | 0.98 | 0.99 | 0.956 | 0.86 | 0.905 |
| kn       | 1 | 0.98 | 0.99 | 1 | 0.98 | 0.99 | 0.878 | 0.86 | 0.869 |
| ml       | 0.979 | 0.92 | 0.946 | 0.907 | 0.98 | 0.942 | 0.754 | 0.92 | 0.828 |
| mn       | 0.98 | 0.98 | 0.98 | 0.98 | 0.96 | 0.97 | 0.956 | 0.86 | 0.903 |
| mr       | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.97 | 0.878 | 0.86 | 0.878 |
| or       | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.941 | 0.96 | 0.95 |
| rj       | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.86 | 0.86 | 0.86 |
| ta       | 1 | 0.96 | 0.98 | 0.904 | 0.94 | 0.922 | 0.784 | 0.8 | 0.792 |
| te       | 1 | 0.98 | 0.99 | 0.979 | 0.94 | 0.959 | 0.935 | 0.86 | 0.896 |

Table 18. Experimental Results for Manually Balancing the Samples for each category to 571.

| Language | CRNN with Attention | CRNN | CNN |
|----------|---------------------|------|-----|
|          | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy | PPV | TPR | F1 Score | Accuracy |
| as       | 1 | 1 | 1 | 0.983 | 0.983 | 0.983 | 0.967 | 1 | 0.983 |
| bn       | 1 | 1 | 1 | 1 | 1 | 1 | 0.983 | 1 | 0.991 |
| bd       | 1 | 1 | 1 | 1 | 1 | 1 | 0.983 | 1 | 0.991 |
| gu       | 1 | 1 | 1 | 0.983 | 1 | 0.991 | 0.982 | 0.931 | 0.956 |
| hi       | 1 | 1 | 1 | 1 | 1 | 1 | 0.983 | 1 | 0.991 |
| kn       | 1 | 1 | 1 | 0.983 | 1 | 0.991 | 0.982 | 0.931 | 0.956 |
| ml       | 1 | 0.966 | 0.982 | 0.983 | 1 | 0.991 | 0.914 | 0.941 | 0.914 |
| mn       | 0.983 | 1 | 0.991 | 1 | 0.983 | 0.991 | 0.931 | 0.931 | 0.931 |
| mr       | 1 | 1 | 1 | 0.982 | 1 | 0.991 | 0.965 | 0.982 | 0.973 |
| or       | 1 | 1 | 1 | 1 | 0.983 | 0.991 | 1 | 0.966 | 0.982 |
| rj       | 0.919 | 0.983 | 0.95 | 0.918 | 0.966 | 0.941 | 0.9 | 0.911 | 0.915 |
| ta       | 0.964 | 0.931 | 0.947 | 0.964 | 0.914 | 0.938 | 0.879 | 0.879 | 0.879 |
| te       | 1 | 0.983 | 0.991 | 1 | 0.983 | 0.991 | 0.982 | 0.931 | 0.956 |
The objective of the study was to observe the performance of the frameworks in increasing the sample size. Since the Bodo language has the minimum data (571 samples) among all the languages in the dataset, we performed our experiments on 571 samples.

A comparison of the results in Table 16, Table 17, and Table 18 reveals the following observations.

- All the models perform consistently better with more training data.
- CRNN and CRNN with attention perform consistently better than CNN.
- CRNN is less data hungry among the 3 models and it performs the best in the lowest data scenario.

![Figure 3: Comparison of model results for varying dataset size.](image)

Figure 3 graphically shows the performance improvement over increasing data samples. The confusion matrices for the three frameworks for the 3 datasets are presented in Table A.1, A.2, A.3, B.1, B.2, B.3, C.1, C.2, C.3 in the Appendix.

### 4.7.3 Additional performance and parameter size analysis of our frameworks

Table 19: A comprehensive performance analysis of our various proposed frameworks.

| Framework                      | CNN  | CRNN | CRNN with Attention |
|--------------------------------|------|------|---------------------|
| Parameters                     | 1,355,917 | 2,094,477 | 2,357,645 |
| Indian Dataset                 |      |      |                     |
|                               | 0.983 | 0.987 | 0.987               |
| Close Language Cluster         |      |      |                     |
| Cluster 1                      | 0.971 | 0.974 | 0.980               |
| Cluster 2                      | 0.996 | 0.999 | 0.999               |
| Cluster 3                      | 0.996 | 1     | 0.999               |
| European Language Dataset      |      |      |                     |
| No Noise                       | 0.948 | 0.967 | 0.966               |
| White Noise                    | 0.871 | **0.912** | 0.888               |

Table 19 demonstrates that both CRNN and CRNN with attention perform better compared to the CNN-based framework. At the same time, CRNN itself produces better or equivalent performance compared to CRNN with an Attention-based mechanism. CRNN with Attention performs
better only for Cluster 1 of the Indian dataset; CRNN itself produces the best results in all other tasks, sometimes jointly with CRNN with Attention. This is despite the fact that the Attention-based framework has more parameters than the other models. The underlying intuition is that the attention-based framework generally suffers from overfitting problems due to its additional parameter count. An attention-based framework needs to learn how to assign importance to different parts of the input sequence, which may require a large number of training instances to produce a generalized performance. Thus, CRNN with Attention makes the experimental set-up time-consuming and resource-intensive, but still, it is not able to improve over CRNN.

5. Conclusion and future work
In this work, we proposed a language identification method using CRNN that works on MFCC features of speech signals. Our framework efficiently identifies the language both in close language and noisy scenarios. We carried out extensive experiments and our framework produced state-of-the-art results. Through our experiments, we have also shown our framework’s robustness to noise and its extensibility to new languages. The model exhibits the overall best accuracy of 98.7% which improves over the traditional use of CNN (98.3%). CRNN with attention performs almost at par with CRNN, however, the attention mechanism which incurs additional computational overhead does not result in improvement over CRNN in most cases.

In future, we would like to extend our work by increasing the language classes with speech specimens recorded in different environments. We would also like to extend our work to check the usefulness of the proposed framework on smaller time speech samples through which we can deduce the optimal time required to classify the languages with high accuracy. We would also like to test our method on language dialect identification.

Acknowledgements
This research was supported by the TPU Research Cloud (TRC) program, a Google Research initiative and funded by Rashtriya Uchchatar Shiksha Abhiyan 2.0 [grant number R-11/828/19].

References
B. Aarti and S. K. Kopparapu, "Spoken Indian language classification using artificial neural network — An experimental study," 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), 2017, pp. 424-430, doi: 10.1109/SPIN.2017.8049987.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: a system for large-scale machine learning. In Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI’16). USENIX Association, USA, 265–283.

Bartz, C., Herold, T., Yang, H., Meinel, C. (2017). Language Identification Using Deep Convolutional Recurrent Neural Networks. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S. (eds) Neural Information Processing. ICONIP 2017. Lecture Notes in Computer Science(), vol 10639. Springer, Cham. https://doi.org/10.1007/978-3-319-70136-3_93

Chen, Pinzhen & Heafield, Kenneth. (2020). Approaching Neural Chinese Word Segmentation as a Low-Resource Machine Translation Task.

N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 4, pp. 788-798, May 2011, doi: 10.1109/TASL.2010.2064307.

Dehak, N., Torres-Carrasquillo, P.A., Reynolds, D., Dehak, R. (2011) Language recognition via i-vectors and dimensionality reduction. Proc. Interspeech 2011, 857-860, doi: 10.21437/Interspeech.2011-328

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages
Sadjadi, S.O., Kheyrkhah, T., Greenberg, C., Singer, E., Reynolds, D., Mason, L., Hernandez-Cordero, J. (2018) Performance Analysis of the 2017 NIST Language Recognition Evaluation. Proc. Interspeech 2018, 1798-1802, doi: 10.21437/Interspeech.2018-69

D. S. Sisodia, S. Nikhil, G. S. Kiran and P. Sathvik, "Ensemble Learners for Identification of Spoken Languages using Mel Frequency Cepstral Coefficients," 2nd International Conference on Data, Engineering and Applications (IDEA), 2020, pp. 1-5, doi: 10.1109/IDEA49133.2020.9170720.

Smith, L.N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. ArXiv, abs/1803.09820.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

Takase, S., & Kiyono, S. (2021). Lessons on Parameter Sharing across Layers in Transformers. ArXiv, abs/2104.06022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS’17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

H. Venkatesan, T. V. Venkatasubramanian and J. Sangeetha, “Automatic Language Identification using Machine learning Techniques,” 2018 3rd International Conference on Communication and Electronics Systems (ICCES), 2018, pp. 583-588, doi: 10.1109/CESYS.2018.8724070.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6442–6454, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: general-ized autoregressive pretraining for language understanding. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, Article 517, 5753–5763.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Zazo R, Lozano-Diez A, Gonzalez-Dominguez J, T. Toledano D, Gonzalez-Rodriguez J (2016) Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks. PLOS ONE 11(1): e0146917. https://doi.org/10.1371/journal.pone.0146917

M. A. Zissman. "Comparison of four approaches to automatic language identification of telephone speech," in IEEE Transactions on Speech and Audio Processing, vol. 4, no. 1, pp. 31-, Jan. 1996, doi: 10.1109/TSA.1996.481450.
Appendix A. CNN framework

Table A.1: Confusion matrix of Manually Balancing the Samples for each category to 100 with CNN

| Actual | Predicted | PPV | TPR  | f1 Score |
|--------|------------|-----|------|----------|
| as     | 29         | 2   | 0    | 0.617    |
| bn     | 2          | 40  | 0    | 0.816    |
| bd     | 0          | 43  | 1    | 0.843    |
| gu     | 1          | 0   | 0    | 0.731    |
| hi     | 2          | 0   | 0    | 0.778    |
| kn     | 0          | 0   | 1    | 0.725    |
| ml     | 0          | 1   | 0    | 0.575    |
| mr     | 0          | 1   | 1    | 0.811    |
| or     | 0          | 1   | 1    | 0.711    |
| rj     | 2          | 2   | 1    | 0.778    |
| ta     | 5          | 0   | 0    | 0.843    |
| te     | 4          | 1   | 2    | 0.843    |

Table A.2: Confusion matrix of Manually Balancing the Samples for each category to 200 with CNN

| Actual | Predicted | PPV | TPR  | f1 Score |
|--------|------------|-----|------|----------|
| as     | 44         | 1   | 1    | 0.688    |
| bn     | 1          | 46  | 0    | 0.92     |
| bd     | 0          | 47  | 1    | 0.94     |
| gu     | 1          | 0   | 0    | 0.918    |
| hi     | 0          | 1   | 0    | 0.956    |
| kn     | 0          | 1   | 0    | 0.878    |
| ml     | 1          | 0   | 1    | 0.86     |
| mn     | 2          | 0   | 0    | 0.754    |
| mr     | 0          | 1   | 2    | 0.956    |
| or     | 0          | 0   | 1    | 0.941    |
| rj     | 2          | 1   | 1    | 0.86     |
| ta     | 2          | 0   | 0    | 0.784    |
| te     | 2          | 0   | 0    | 0.935    |

Table A.3: Confusion matrix of Manually Balancing the Samples for each category to 571 with CNN

| Actual | Predicted | PPV | TPR  | f1 Score |
|--------|------------|-----|------|----------|
| as     | 58         | 0   | 0    | 0.967    |
| bn     | 0          | 58  | 0    | 0.983    |
| bd     | 0          | 56  | 0    | 0.983    |
| gu     | 0          | 54  | 4    | 0.983    |
| hi     | 0          | 50  | 2    | 0.983    |
| kn     | 0          | 56  | 2    | 0.903    |
| ml     | 0          | 53  | 0    | 0.914    |
| mn     | 1          | 0   | 0    | 0.914    |
| mr     | 0          | 0   | 0    | 0.914    |
| or     | 0          | 1   | 1    | 0.914    |
| rj     | 1          | 1   | 0    | 0.883    |
| ta     | 0          | 0   | 1    | 0.914    |
| te     | 0          | 0   | 0    | 0.914    |
# Appendix B. CRNN framework

Table B.1. : Confusion matrix of Manually Balancing the Samples for each category to 100 with CRNN

| Actual | Predicted | PPV | TPR | f1 Score |
|--------|-----------|-----|-----|----------|
| as | 47 0 0 0 0 0 0 1 0 0 1 0 0.839 | 0.94 | 0.847 |
| bn | 0 45 0 0 0 0 0 3 0 0 0 1 0.957 | 0.9 | 0.928 |
| bd | 0 0 50 0 0 0 0 0 0 0 0 1 0.962 | 1 | 0.98 |
| gu | 0 0 0 49 0 0 0 0 0 0 0 1 1 0.957 | 0.9 | 0.928 |
| hi | 0 0 0 0 45 2 0 0 0 0 0 0 0.962 | 1 | 0.98 |
| kn | 1 0 0 0 0 0 0 1 0 0 0 0 0.962 | 1 | 0.98 |
| ml | 0 0 0 0 0 0 0 0 0 0 0 0 0.962 | 1 | 0.98 |
| mn | 1 0 0 0 0 0 0 1 0 0 0 1 0.962 | 1 | 0.98 |
| mr | 0 0 0 0 0 0 0 0 0 0 0 0 0.962 | 1 | 0.98 |
| or | 0 0 0 0 0 0 0 1 0 0 0 0 0.962 | 1 | 0.98 |
| rj | 1 0 0 0 0 0 0 0 0 0 0 0 0.962 | 1 | 0.98 |
| ta | 0 0 0 0 0 0 0 0 0 0 0 0 0.962 | 1 | 0.98 |
| te | 0 0 0 0 0 0 0 0 0 0 0 0 0.962 | 1 | 0.98 |

Table B.2. : Confusion matrix of Manually Balancing the Samples for each category to 200 with CRNN

| Actual | Predicted | PPV | TPR | f1 Score |
|--------|-----------|-----|-----|----------|
| as | 47 0 0 0 0 0 0 1 0 0 1 0 0.839 | 0.94 | 0.969 |
| bn | 0 48 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| bd | 0 0 49 0 0 0 0 1 0 0 0 0 0.98 | 0.96 | 0.98 |
| gu | 0 0 0 50 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| hi | 0 0 0 0 49 1 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| kn | 0 0 0 0 49 1 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| ml | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| mn | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| mr | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| or | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| rj | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| ta | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |
| te | 0 0 0 0 0 0 0 0 0 0 0 0 0.98 | 0.96 | 0.98 |

Table B.3. : Confusion matrix of Manually Balancing the Samples for each category to 571 with CRNN

| Actual | Predicted | PPV | TPR | f1 Score |
|--------|-----------|-----|-----|----------|
| as | 57 0 0 0 0 0 0 1 0 0 1 0 0.983 | 0.983 | 0.983 |
| bn | 0 58 0 0 0 0 0 0 0 0 0 0 1 1 |
| bd | 0 0 56 0 0 0 0 0 0 0 0 0 0 1 1 |
| gu | 0 0 0 58 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| hi | 0 0 0 0 58 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| kn | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| ml | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| mn | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| mr | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| or | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| rj | 1 0 0 0 0 0 0 1 0 0 0 0 0.983 | 1 | 0.991 |
| ta | 0 0 0 1 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
| te | 0 0 0 0 0 0 0 0 0 0 0 0 0.983 | 1 | 0.991 |
Appendix C. CRNN with Attention framework

Table C.1: Confusion matrix of Manually Balancing the Samples for each category to 100 with CRNN and Attention

| Actual | Predicted | PPV   | TPR   | f1 Score |
|--------|-----------|-------|-------|----------|
| as     | 36        | 0.766 | 0.72  | 0.742    |
| bn     | 1         | 0.875 | 0.7   | 0.778    |
| bd     | 0         | 0.43  | 1     | 0.971    |
| gu     | 0         | 0.943 | 1     | 0.971    |
| hi     | 0         | 0.959 | 0.94  | 0.95     |
| kn     | 0         | 0.961 | 0.98  | 0.97     |
| ml     | 0         | 0.958 | 0.92  | 0.930    |
| mn     | 5         | 0.784 | 0.72  | 0.791    |
| mr     | 0         | 0.906 | 0.98  | 0.932    |
| or     | 0         | 0.959 | 0.94  | 0.949    |
| rj     | 2         | 0.782 | 0.86  | 0.819    |
| ta     | 2         | 0.677 | 0.88  | 0.765    |
| te     | 1         | 0.878 | 0.86  | 0.869    |

Table C.2: Confusion matrix of Manually Balancing the Samples for each category to 200 with CRNN and Attention

| Actual | Predicted | PPV   | TPR   | f1 Score |
|--------|-----------|-------|-------|----------|
| as     | 48        | 0.941 | 0.96  | 0.95     |
| bn     | 0         | 0.909 | 1     | 0.952    |
| bd     | 0         | 0.98  | 1     | 0.99     |
| gu     | 0         | 0.98  | 1     | 0.99     |
| hi     | 0         | 0.98  | 1     | 0.99     |
| kn     | 0         | 0.98  | 1     | 0.99     |
| ml     | 0         | 0.98  | 1     | 0.99     |
| mn     | 5         | 0.979 | 0.92  | 0.944    |
| mr     | 0         | 0.98  | 1     | 0.99     |
| or     | 0         | 0.96  | 0.96  | 0.96     |
| rj     | 2         | 0.96  | 1     | 0.99     |
| ta     | 0         | 0.98  | 1     | 0.98     |
| te     | 0         | 0.98  | 1     | 0.99     |

Table C.3: Confusion matrix of Manually Balancing the Samples for each category to 571 with CRNN and Attention

| Actual | Predicted | PPV   | TPR   | f1 Score |
|--------|-----------|-------|-------|----------|
| as     | 58        | 0.983 | 0.983| 0.993    |
| bn     | 0         | 0.983 | 0.983| 0.993    |
| bd     | 0         | 0.983 | 0.983| 0.993    |
| gu     | 0         | 0.983 | 0.983| 0.993    |
| hi     | 0         | 0.983 | 0.983| 0.993    |
| kn     | 0         | 0.983 | 0.983| 0.993    |
| ml     | 0         | 0.983 | 0.983| 0.993    |
| mn     | 5         | 0.983 | 0.983| 0.993    |
| mr     | 0         | 0.983 | 0.983| 0.993    |
| or     | 0         | 0.983 | 0.983| 0.993    |
| rj     | 0         | 0.983 | 0.983| 0.993    |
| ta     | 0         | 0.983 | 0.983| 0.993    |
| te     | 0         | 0.983 | 0.983| 0.993    |