Highlights

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

Atanu Mandal,Santanu Pal,Indranil Dutta,Mahidas Bhattacharya,Sudip Kumar Naskar

- The proposed method uses Convolutional Neural Network (CNN), Convolutional Recurrent Neural Network (CRNN), and attention based CRNN for the task of Spoken Language Identification.
- Our proposed architecture provides state-of-the-art performance in languages that belong to the same language family as well as in noisy scenarios.

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

Atanu Mandal[a,*], Santanu Pal[b], Indranil Dutta[c], Mahidas Bhattacharya[c] and Sudip Kumar Naskar[a,*]

[a]Department of Computer Science and Engineering, Jadavpur University, 188, Raja S.C. Mallick Rd, Kolkata, 700 032, West Bengal, India
[b]Wipro AI Lab, Wipro India Limited, Doddakannelli, Sarjapur Road, Bangaluru, 560 035, Karnataka, India
[c]School of Languages and Linguistics, Jadavpur University, 188, Raja S.C. Mallick Rd, Kolkata, 700 032, West Bengal, India

ARTICLE INFO

Keywords:
Language Identification
Convolutional Neural Network
Convolutional Recurrent Neural Network
Attention
Indian Languages

ABSTRACT

Language Identification (LID), a recommended initial step to Automatic Speech Recognition (ASR), is used to detect a spoken language from audio specimens. In state-of-the-art systems capable of multilingual speech processing, however, users have to explicitly set one or more languages before using them. LID, therefore, plays a very important role in situations where ASR based systems cannot parse the uttered language in multilingual contexts causing failure in speech recognition. We propose an attention based convolutional recurrent neural network (*CRNN with Attention*) that works on Mel-frequency Cepstral Coefficient (MFCC) features of audio specimens. Additionally, we reproduce some state-of-the-art approaches, namely *Convolutional Neural Network* (*CNN*) and *Convolutional Recurrent Neural Network* (*CRNN*), and compare them to our proposed method. We performed extensive evaluation on thirteen different Indian languages and our model achieves classification accuracy over 98%. Our LID model is robust to noise and provides 91.2% accuracy in a noisy scenario. The proposed model is easily extensible to new languages.

1. Introduction

From the inception of research in Natural Language Processing (NLP), researchers have specifically rely on Convolution Neural Networks (CNN) as it utilizes local features effectively. Earlier Recurrent Neural Network (RNN) was effectively used in different NLP domains, but recent use of Transformer has shown promising results which outperforms all previous experimental results. Attention based models are capable of capturing the content-based global interactions.

Transformer in Question-Answering domain, researcher (Yamada et al., 2020) were able to outperform BERT (Devlin et al., 2019), SpanBERT (Joshi et al., 2020), XLNet (Yang et al., 2019), and ALBERT (Lan et al., 2020). In Machine Translation domain researcher (Takase and Kiyono, 2021; Gu et al., 2019; Chen and Heafield, 2020) used Transformer and were able to outperform other state-of-the-art (sota) algorithms. In other domain like Language Modelling, Text Classification, Topic Modelling, Emotion Classification, Sentiment Analysis, etc Transformer has widely used.

In this work we focused on using different approaches for Spoken Language Identification. Humans are capable of recognizing almost immediately the language being used by a speaker for voicing an utterance. The task of automatic language identification (LID) is to automatically classify the language used by a speaker in his/her speech. In the era of *Internet of Things*, smart and intelligent assistants (e.g., Alexa[1], Siri[2], Cortona[3], Google Assistant[4], etc.) can interact with humans with some default language settings (mostly in English) and these smart assistants rely heavily on Automatic Speech Recognition (ASR). However, these virtual assistants fail to provide any assistance in multilingual

*Corresponding author
atanumandal0491@gmail.com (A. Mandal); santanu.pal2@wipro.com (S. Pal); indranildutta.lnl@jadavpuruniversity.in (I. Dutta); languagemahib@gmail.com (M. Bhattacharya); sudipkumar.naskar@jadavpuruniversity.in (S.K. Naskar)
ORCID(s): 0000-0002-9385-5897 (A. Mandal); 0000-0003-3079-6903 (S. Pal); 0000-0001-8522-5302 (I. Dutta); 0000-0003-1588-4665 (S.K. Naskar)

[1]https://developer.amazon.com/en-US/alexa/alexa-voice-service
[2]https://www.apple.com/in/siri/
[3]https://www.microsoft.com/en-in/windows/cortana
[4]https://assistant.google.com/

contexts. To make such smart assistants robust, LID can be used so that the smart assistants can automatically recognize the speaker's language and change its language setting accordingly.

Our approach of identifying spoken language is limited to Indian Languages only because India is world second populated and seventh largest country in landmass and also have dynamic culture. Currently, India has 28 states and 8 Union Territories, where each states and Union Territories has its own language, but none of the language is recognised as the national language of the country. Only, English and Hindi is used as official language according to the Constitution of India Part XVII Chapter 1 Article 343[5]. Currently, only 22 languages have been accepted as regional languages.

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 and Kashmir
5	Gujarati	Indo-Aryan	Gujrat, Dadra and Nagar Haveli and Daman and Diu
6	Hindi	Indo-Aryan	Andaman and Nicobar Islands, Bihar, Chhattisgarh, Dadra and Nagar Haveli and Daman and Diu, Delhi, Haryana, Himachal Pradesh, Jammu and Kashmir, Jharkhand, Ladakh, Madhya Pradesh, Mizoram, Rajasthan, Uttar Pradesh, Uttarakhand
7	Kannada	Dravidian	Karnataka
8	Kashmiri	Indo-Aryan	Jammu and Kashmir
9	Konkani	Indo-Aryan	Dadra and Nagar Haveli and Daman and Diu, Goa
10	Maithili	Indo-Aryan	Jharkhand
11	Malayalam	Dravidian	Kerala, Lakshadweep, Puducherry
12	Marathi	Indo-Aryan	Dadra and Nagar Haveli and Daman and Diu, Goa, Maharashtra
13	Meitei	Sino-Tibetan	Manipur
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 and Kashmir, Jharkhand, Telangana, Uttar Pradesh

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

Table 1 describes the 22 languages designated as Official language according to the Eighth Schedule of the Constitution of India, as of 1 December 2007. Most of the Indian languages originated from Indo-Aryan and Dravidian language family.

[5]https://www.mea.gov.in/Images/pdf1/Part17.pdf

It can be seen from the Table 1 that different languages are spoken in different states, however, languages do not obey the geographical boundaries. Therefore, many of these languages, particularly in the neighboring regions, have multiple dialects which are amalgamation of two or more languages.

Such enormous linguistic diversity makes it difficult for the citizens to communicate in different parts of the country. Bilingualism and multilingualism are the norm in India. In this context, an LID system becomes a crucial component for any speech based smart assistant. The biggest challenge and hence an area of active innovation for 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. 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 purpose of our approach is yet to predict the correct spoken language regardless of the above-mentioned constraints.

In this work we proposed Language Identification method for Indian Languages using different approaches. Our LID methods cover 13 Indian languages[6]. Additionally our method is language agnostic. The main contributions of this work can be summarized as follows:

- The method uses Convolutional Neural Network (CNN), Convolutional Recurrent Neural Network (CRNN), and attention based CRNN for the task of LID. We tested 13 Indian languages achieving state-of-the-art accuracy.
- Our model also provides state-of-the-art performance in languages that belong to the same language family as well as in noisy scenarios.

2. Related Works

Extraction of language dependent features for example prosody and phonemes was widely used to classify spoken languages (Zissman, 1996; Martínez et al., 2011; Ferrer et al., 2010). Following the success in speaker verification systems, identity vectors (i-vectors) have also been used as features in various classification architectures. Use of i-vectors requires significant domain knowledge (Dehak et al., 2011b; Martínez et al., 2011). 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 (2019a) used the ResNet50 (He et al., 2016) architecture for classifying languages by generating the log-Mel spectra for each raw audio. The architecture uses cyclic learning rate where learning rate increases and then decreases linearly. Maximum learning rate for a cycle is set by finding the optimal learning rate using fastai (Howard and Gugger, 2020).

Researcher Gazeau and Varol (2018) established the use of Neural Network, Support Vector Machine, and Hidden Markov Model (HMM) to identify different languages. Hidden Markov models converts speech into a sequence of vectors and was used to capture temporal features in speech. Established LID systems (Dehak et al., 2011a; Martínez et al., 2011; Plchot et al., 2016; Zazo et al., 2016) are based on identity vector (i-vectors) representations for language processing tasks. In Dehak et al. (2011a), i-vectors are used as data representations for a speaker verification task and fed to the classifier as the input. Dehak et al. (2011a) applied Support Vector Machines (SVM) with cosine kernels as the classifier, while Martínez et al. (2011) 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., 2011a; Martínez et al., 2011; 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 Convolutional Neural Network where i-vectors were the input layer and softmax activation function as the output layer. Zazo et al. (2016) used Mel Frequency Cepstral Coefficients (MFCCs) with Shifted Delta Coefficient features as information to a unidirectional layer which is directly connected to a softmax

[6]The study was limited to the number of Indian languages for which datasets were available

classifier. Gelly et al. (2016) used audio transformed to Perceptual Linear Prediction (PLP) coefficients and their 1^{st} and 2^{nd} 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 Convolutional Neural Networks (CNNs) for their LID system. They transformed the input data as 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 (2019b) used ResNet (He et al., 2016) architecture 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 Mel-Frequency Cepstral Coefficients (MFCC) to infer aspects of speech signals from Kannada, Hindi, Tamil, and Telugu. They obtained 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 Convolutional Neural Networks (CNN) For language identification on German, Spanish, and English. They used Filter Banks to extract features from frequency domain representations of the signal. Aarti and Kopparapu (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 Convolutional Neural Networks (CNN) for Spoken Language Identification (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 Convolutional Recurrent Neural Network (CRNN) is employed for this, which acts on spectrogram images of the provided audio clips. Draghici et al. (2020) tried to solve the task of Language Identification while using Mel-spectrogram images as input features. This strategy was employed in Convolutional Neural Network (CNN) and Convolutional Recurrent Neural Network (CRNN) in terms of performance. This work is characterized by a modified training strategy which provides equal class distribution and efficient memory utilisation. Ganapathy et al. (2014) reported how they used bottleneck features from a Convolutional Neural Network 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 shows its applicability both for close language as well as noisy speech scenarios.

3. Model Architecture

Our proposed architecture consists of three models.

- CNN based architecture
- CRNN based architecture
- CRNN with Attention based architecture

We made use of the capacity of CNNs to capture spatial information to identify languages from audio samples. In CNN based architecture our network uses four convolution layers, where each layer is followed by ReLU (Nair and Hinton, 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 1 provides a schematic overview of the network architecture. In CRNN based architecture the Bi-Directional LSTM consisting of a single LSTM with 256 output units was used after the Convolution module. The Attention

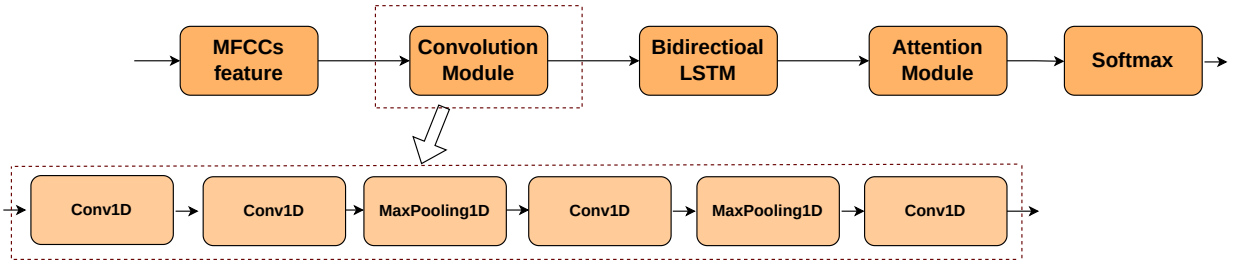

Figure 1: The figure presents our architecture consisting of Convolution module, Bi-Directional LSTM, and Attention Mechanism denoted in different blocks. Convolution module extracts features from the input audio. The output of the final convolution layer is provided to the Bi-Directional LSTM network as the input which is further connected to the Attention module for adequate learning. The Attention module is finally connected to a softmax classifier.

Mechanism used in our architecture 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 backward.

$$\begin{aligned} \overrightarrow{a_n} &= \overrightarrow{LSTM}(a_n), n \in [1, L] \\ \overleftarrow{a_n} &= \overleftarrow{LSTM}(a_n), n \in [L, 1] \\ a_i &= [\overrightarrow{a_n}, \overleftarrow{a_n}] \end{aligned} \tag{1}$$

$$a_i^{'} = tanh(a_i \cdot w_i + b_i) \cdot u_i \tag{2}$$

In equation 1, L is the the number of audio specimen. 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 *tanh* function that the network does not stall. The function keeps the input values between -1 and 1, and also maps zeros to near-zero values. The layer with *tanh* function are again multiplied by trainable context vector u_i. 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.

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.

$$W_i = \frac{exp(a_i^{'})}{\sum_i exp(a_i^{'}) + \epsilon} \tag{3}$$

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.

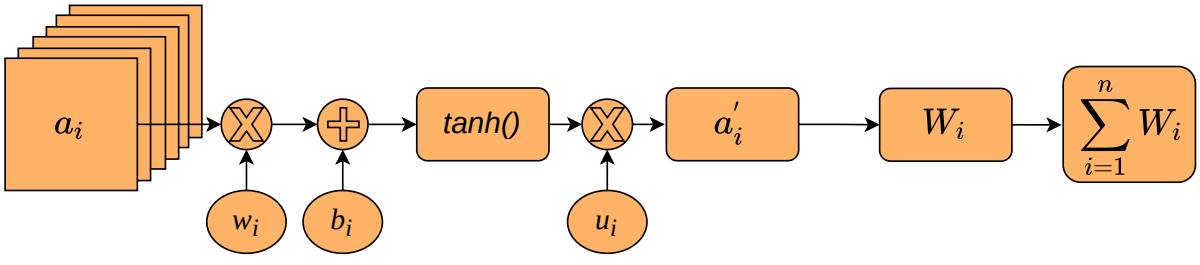

Figure 2: Schematic diagram of the Attention Module.

Figure 2 presents the schematic diagram of the Attention Module where a_i is the input to the 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_size, frame stride represented as f_stride, N-point Fast Fourier transform represented as $NFFT$, low frequency mel represented as lf, number of filter represented as $nfilt$, number of cepstral coefficients represented as $ncoef$, 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 frame size of 25 ms as typically frame sizes in speech processing domain uses 20ms to 40ms with 50% (in our case 15ms) overlapping between consecutive frames.

$$hf = 2595 \times \log_{10}(1 + \frac{0.5 \times sr}{700}) \tag{4}$$

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

$$emphasized\_signal = [sig[0], sig[1:] - pre\_emphasis * sig[:-1]] \tag{5}$$

As shown in equation 5 emphasized signal is calculated by using pre-emphasis filter applied on signal using first order filter. Number of frames is calculated by taking the ceiling value of division of absolute value of 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.

$$n\_frames = \lceil \frac{|sig\_len - (f\_size \times sr)|}{(f\_stride \times sr)} \rceil \tag{6}$$

Using equation 7 pad_signal is generated from concatenation of $emphasized\_signal$ and zero value array of dimension $(pad\_signal\_length - signal\_length)\times 1$, where, pad_signal_length is calculated by $n\_frames \times (f\_stride \times sr) + (f\_size \times sr)$.

$$pad\_signal = [emphasized\_signal, [0]_{((n\_frames\times(f\_stride\times sr)+(f\_size\times sr))-sig\_len)\times 1}] \tag{7}$$

Frames are calculated as shown in equation 8 from the pad_signal elements where elements are the addition of array of positive natural number from 0 to $f\_size \times sr$ repeated n_frames and transpose of array of size of num_frames where each element is the difference of $(f\_stride \times sr)$.

$$\begin{aligned} frames = pad\_signal[(\{x \in Z^+ : 0 < x < (f\_size \times sr)\}_n)_{n=0}^{(n\_frames,1)} + \\ ((\{r : r = (f\_stride \times sr) \times (i-1), i \in \{0, \dots, n\_frames \times (f\_stride \times sr)\}\}_n)_{n=0}^{((f\_size\times sr),1)})^\intercal] \end{aligned} \tag{8}$$

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

$$pf = \frac{|DFT((frames \times (0.54 - (\sum_{N=0}^{(f\_size\times sr)-1} 0.46 \times \cos \frac{2\pi N}{(f\_size\times sr)-1}))), NFFT)|^2}{NFFT} \tag{9}$$

$$mel\_points = \{r : r = lf + \frac{hf - lf}{(nfilt + 2) - 1} \times i, i \in \{lf, \dots, hf\}\} \tag{10}$$

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

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

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

$$fbank_m(k) = \begin{cases} 0 & k < bins(m-1) \\ \frac{k - bins(m-1)}{bins(m) - bins(m-1)} & bins(m-1) \leq k \leq bins(m) \\ \frac{bins(m+1) - k}{bins(m+1) - bins(m)} & bins(m) \leq k \leq bins(m+1) \\ 0 & k > bins(m+1) \end{cases} \tag{12}$$

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 decorrelate 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 to classify in noisy signals.

$$mfcc = DCT(20\log_{10}(pf \cdot fbank^{\intercal})) \times \left[1 + \frac{lifter}{2}\sin\frac{\{\pi \odot n : n \in Z^{+}, n \leq ncoef\}}{lifter}\right] \tag{13}$$

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 architecture 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 MFCCs 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

In the past two decades, development of LID methods has been largely fostered through NIST Language Evaluations (LREs). As a result, the most popular benchmarks for evaluating new LID models and methods are 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 European Language Dataset Bartz et al. (2017) which is open sourced. The European language (*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 2.

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

Table 2
Statistics of the European Language (*EU*) Dataset

4.2.2. Experimental Data

The Indian language (*IL*) dataset was acquired from the Indian Institute of Technology, Madras[7]. The dataset includes 13 widely used Indian languages. Table 3 presents the statistics of this dataset which we used for our experiments.

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		

Table 3
Statistics of the Indian Language (*IN*) Dataset

4.3. Environment

We implemented our architecture 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 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 and Ba, 2015) 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. Batch size of 64 with "Sparse Categorical Crossentrop" as the loss function were used.

4.4. 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 to each test sample which has a strong audible presence but retains the identity of the language.

[7]https://www.iitm.ac.in/donlab/tts/database.php

	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

Table 4
Comparative evaluation results (in terms of Accuracy) of our model and the model of Bartz et al. (2017) on the YouTube News (EU) dataset

Table 4 compares the results of our models on the EU dataset with state-of-the-art models presented by Bartz et al. (2017). Proposed model by Bartz et al. (2017) consists of CRNN and uses Google's Inception-v3 architecture (Szegedy et al., 2016). They experimented in four different environments – No Noise, White Noise, Cracking Noise, and Background Noise. All our evaluation results are rounded up to 3 digit after decimal point.

The CNN model failed to achieve competitive results; it provided accuracy of 0.948/0.871 in No Noise/White Noise. In CRNN architecture, our model provides accuracy of 0.967/0.912 on 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 No Noise scenario, however it does not perform well on White Noise

4.5. Result on Indian Language Dataset

We evaluated our proposed architecture with the proposed architecture by Kulkarni et al. (2022) on the same datasets. Our two architectures i.e. CRNN and CRNN with Attention was able to outperform the accuracy obtained by Kulkarni et al. (2022) as shown in Table 5. They used 6 Linear layers where units are 256, 256, 128, 64, 32, 13, respectively in the CNN architecture, whereas the DNN architecture uses 3 LSTM layers having units 256, 256, 128, respectively followed by a dropout layer followed by 3 Time Distributed layer followed by Linear layer of 13 as units.

	Accuracy
DNN (Kulkarni et al., 2022)	98.34%
RNN (Kulkarni et al., 2022)	98.43%
CNN	98.3%
CRNN	**98.7%**
CRNN with Attention	**98.7%**

Table 5
Comparative evaluation results (in terms of Accuracy) of our model and the model of Kulkarni et al. (2022) on the Indian Language dataset

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 case for the Indian Language Dataset as shown in Table 3 in which Kannada, Marathi and particularly Bodo have limited amount of data compared to the rest of the languages. To alleviate this data imbalance problem, we used class weight balancing as a dynamic method using scikit-learn (Pedregosa et al., 2011).

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	0.989	0.992		0.991	0.988	0.989	
bn	1	0.9	0.948		1	0.904	0.949		1	0.888	0.941	
bd	0.966	1	0.983		0.966	1	0.983		0.966	1	0.983	
gu	0.997	0.997	0.997		0.951	0.998	0.974		0.996	0.991	0.994	
hi	0.987	0.991	0.989		0.991	0.991	0.991		0.991	0.974	0.983	
kn	0.977	0.996	0.987		0.996	0.996	0.996		0.973	0.992	0.983	
ml	0.996	0.988	0.992	0.987	0.997	0.99	0.994	0.987	0.99	0.993	0.991	0.983
mn	0.987	0.999	0.993		0.973	0.999	0.986		0.972	0.998	0.985	
mr	1	1	1		1	1	1		1	0.996	0.998	
or	1	1	1		1	0.999	0.999		0.986	0.999	0.992	
rj	0.999	0.993	0.996		0.995	1	0.997		0.986	0.996	0.991	
ta	0.929	0.991	0.959		0.975	0.997	0.986		0.946	0.989	0.967	
te	0.979	0.998	0.989		0.982	1	0.991		0.975	0.998	0.986	

Table 6
Experimental Results for Indian Languages

PPV, TPR, f1-score and Accuracy scores are reported in Table 6 for the three architectures - CNN, CRNN, and CRNN with Attention. From Table 6 it is clearly visible that both CRNN architecture and CRNN with Attention provide competitive results of 0.987 accuracy. Table 7, 8, and 9 shows the confusion matrix for CNN, CRNN, and CRNN with Attention.

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**1762**	0	0	0	0	0	0	4	0	0	0	0	0	0.989	0.998	0.993
	bn	10	**850**	0	1	0	0	0	18	0	0	0	53	12	1	0.9	0.948
	bd	0	0	**57**	0	0	0	0	0	0	0	0	0	0	0.966	1	0.983
	gu	1	0	0	**566**	0	0	0	0	0	0	0	0	1	0.997	0.997	0.997
	hi	0	0	0	0	**460**	0	4	0	0	0	0	0	0	0.987	0.991	0.989
	kn	0	0	1	0	0	**257**	0	0	0	0	0	0	0	0.977	0.996	0.987
	ml	0	0	1	0	6	5	**1116**	1	0	0	0	0	1	0.996	0.988	0.992
	mn	0	0	0	1	0	0	0	**1790**	0	0	0	0	0	0.987	0.999	0.993
	mr	0	0	0	0	0	0	0	0	**245**	0	0	0	0	1	1	1
	or	0	0	0	0	0	0	0	0	0	**716**	0	0	0	1	1	1
	rj	4	0	0	0	0	1	1	0	0	0	**906**	0	0	0.999	0.993	0.996
	ta	5	0	0	0	0	0	0	0	0	0	1	**690**	0	0.929	0.991	0.959
	te	0	0	0	0	0	0	0	1	0	0	0	0	**652**	0.979	0.998	0.989

Table 7
Confusion matrix for CRNN with Attention architecture

From Table 7, 8, and 9 it can be observed that Assamese gets confused with Meitei; Bengali gets confused with Assamese, Meitei, Tamil and Telugu; and Hindi gets confused with Malayalam.

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**1746**	0	0	0	0	0	0	19	0	0	0	1	0	0.995	0.989	0.992
	bn	8	**853**	0	28	0	0	0	28	0	0	0	17	10	1	0.904	0.949
	bd	0	0	**57**	0	0	0	0	0	0	0	0	0	0	0.966	1	0.983
	gu	0	0	0	**567**	0	0	0	0	0	0	0	0	1	0.951	0.998	0.974
	hi	0	0	0	0	**460**	0	3	0	0	0	1	0	0	0.991	0.991	0.991
	kn	0	0	1	0	0	**257**	0	0	0	0	0	0	0	0.996	0.996	0.996
	ml	0	0	1	0	4	1	**1119**	1	0	0	4	0	0	0.997	0.99	0.994
	mn	0	0	0	1	0	0	0	**1789**	0	0	0	0	1	0.973	0.999	0.986
	mr	0	0	0	0	0	0	0	0	**245**	0	0	0	0	1	1	1
	or	0	0	0	0	0	0	0	1	0	**715**	0	0	0	1	0.999	0.999
	rj	0	0	0	0	0	0	0	0	0	0	**912**	0	0	0.995	1	0.997
	ta	1	0	0	0	0	0	0	1	0	0	0	**694**	0	0.975	0.997	0.986
	te	0	0	0	0	0	0	0	0	0	0	0	0	**653**	0.982	1	0.991

Table 8
Confusion matrix for CRNN

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**1744**	0	1	0	0	3	1	13	0	0	0	4	0	0.991	0.988	0.989
	bn	9	**838**	0	1	1	0	0	35	0	8	5	34	13	1	0.888	0.941
	bd	0	0	**57**	0	0	0	0	0	0	0	0	0	0	0.966	1	0.983
	gu	2	0	0	**563**	0	0	0	0	0	1	0	0	2	0.996	0.991	0.994
	hi	0	0	0	0	**452**	0	10	0	0	0	2	0	0	0.991	0.974	0.983
	kn	0	0	0	0	1	**256**	0	0	0	0	1	0	0	0.973	0.992	0.983
	ml	0	0	1	0	2	2	**1122**	0	0	0	3	0	0	0.99	0.993	0.991
	mn	1	0	0	0	0	0	0	**1788**	0	0	0	1	1	0.972	0.998	0.985
	mr	0	0	0	0	0	0	0	0	**244**	1	0	0	0	1	0.996	0.998
	or	0	0	0	0	0	0	0	1	0	**715**	0	0	0	0.986	0.999	0.992
	rj	1	0	0	0	0	2	1	0	0	0	**908**	0	0	0.986	0.996	0.991
	ta	3	0	0	1	0	0	0	1	0	0	2	**688**	1	0.946	0.989	0.967
	te	0	0	0	0	0	0	0	1	0	0	0	0	**652**	0.975	0.998	0.986

Table 9
Confusion matrix for CNN

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 family but share 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, Meitei and Assamese share similar phonemes. On close study we observed that Hindi and Malayalam have also similar phoneme set as both the languages borrowed most of the vocabularies 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 10 shows the most common classification errors encountered during evaluation.

Assamese	→	Meitei
Bengali	→	Assamese
Bengali	→	Meitei
Bengali	→	Tamil
Hindi	→	Malayalam

Table 10
Most common errors

4.6. 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 be differentiated.

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

Bodo and Meitei are phonetically very much distant from any of the rest of the languages, thus they form singleton clusters. We carried out separate experiments for identification of the cluster internal languages for Cluster 1, 2 and 3, and the experimental results are presented in Table 11.

It can be clearly observed from Table 11 that both CRNN architecture and CRNN with Attention provide competitive results for every language cluster. For **cluster-1** CRNN architecture and CRNN with Attention provides accuracy of 0.98/0.974, for **cluster-2** 0.999/0.999, and for **cluster-3** 0.999/1, respectively. CNN architecture also provides comparable results to the other two architectures.

Cluster	Language	CRNN with Attention				CRNN				CNN			
		PPV	TPR	f1 Score	Accuracy	PPV	TPR	f1 Score	Accuracy	PPV	TPR	f1 Score	Accuracy
1	as	0.962	1	0.981	**0.98**	0.953	1	0.976	**0.974**	0.953	1	0.976	**0.971**
	bn	1	0.926	0.961		1	0.907	0.951		1	0.894	0.944	
	or	1	1	1		1	1	1		0.982	1	0.991	
2	gu	1	0.998	0.999	**0.999**	1	0.998	0.999	**0.999**	1	0.993	0.996	**0.996**
	hi	1	1	1		1	0.998	0.999		0.991	0.996	0.994	
	mr	1	1	1		1	1	1		1	0.996	0.998	
	rj	0.999	1	0.999		0.998	1	0.999		0.995	0.998	0.996	
3	kn	1	0.996	0.998	**0.999**	1	1	1	**1**	0.992	0.988	0.99	**0.996**
	ml	0.999	1	0.999		1	1	1		0.996	0.996	0.996	
	ta	1	1	1		1	1	1		0.996	0.997	0.996	
	te	1	1	1		1	1	1		0.995	0.997	0.996	

Table 11
Experimental Results of LID for close languages.

Table 12, Table 13 and Table 14 presents the confusion matrix for cluster 1, cluster 2 and cluster 3, respectively.

CRNN and Attention

		Predicted			PPV	TPR	f1 Score
		as	bn	or			
Actual	as	**1766**	0	0	0.962	1	0.981
	bn	70	**874**	0	1	0.926	0.961
	or	0	0	**716**	1	1	1

CRNN

		Predicted			PPV	TPR	f1 Score
		as	bn	or			
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

CNN

		Predicted			PPV	TPR	f1 Score
		as	bn	or			
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

Table 12
Confusion matrix for Cluster 1

From Table 12, 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.

CRNN and Attention								
		Predicted				PPV	TPR	f1 Score
		gu	hi	mr	rj			
Actual	gu	**567**	0	0	1	1	0.998	0.999
	hi	0	**464**	0	0	1	1	1
	mr	0	0	**245**	0	1	1	1
	rj	0	0	0	**912**	0.999	1	0.999
CRNN								
		Predicted				PPV	TPR	f1 Score
		gu	hi	mr	rj			
Actual	gu	**567**	0	0	1	1	0.998	0.999
	hi	0	**463**	0	1	1	0.998	0.999
	mr	0	0	**245**	0	1	1	1
	rj	0	0	0	**912**	0.998	1	0.999
CNN								
		Predicted				PPV	TPR	f1 Score
		gu	hi	mr	rj			
Actual	gu	**564**	2	0	2	1	0.993	0.996
	hi	0	**462**	0	2	0.991	0.996	0.994
	mr	0	0	**244**	1	1	0.996	0.998
	rj	0	2	0	**910**	0.995	0.998	0.996

Table 13
Confusion matrix for Cluster 2

It can be observed from Table 14 that CNN makes a lot of confusion discriminating among these four languages. Both CRNN and CRNN with Attention prove to be better at discriminating among these languages.

CRNN and Attention								
		Predicted				PPV	TPR	f1 Score
		kn	ml	ta	te			
Actual	kn	**257**	1	0	0	1	0.996	0.998
	ml	0	**1130**	0	0	0.999	1	0.999
	ta	0	0	**696**	0	1	1	1
	te	0	0	0	**653**	1	1	1
CRNN								
		Predicted				PPV	TPR	f1 Score
		kn	ml	ta	te			
Actual	kn	**258**	0	0	0	1	1	1
	ml	0	**1130**	0	0	1	1	1
	ta	0	0	**696**	0	1	1	1
	te	0	0	0	**653**	1	1	1
CNN								
		Predicted				PPV	TPR	f1 Score
		kn	ml	ta	te			
Actual	kn	**255**	3	0	0	0.992	0.988	0.99
	ml	1	**1125**	2	2	0.996	0.996	0.996
	ta	1	0	**694**	1	0.996	0.997	0.996
	te	0	1	1	**651**	0.995	0.997	0.996

Table 14
Confusion matrix for Cluster 3

From the results in Table 11, 12, 13 and 14, it is quite 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 the other two clusters.

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, 65 of the models. We found that performance decreases with larger kernel sizes, as shown in Table 15. On comparing the accuracy upto second decimal places kernel size 3 performs better than rest.

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

Table 15
Ablation study on convolution kernel sizes

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 6. We study the efficacy of the architectures 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, 17 and 18, respectively.

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	**0.883**	0.923	0.96	0.941	**0.932**	0.774	0.82	0.796	**0.72**
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		0.943	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 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	0.941	0.96	0.95		1	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	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	0.962	1	0.98	**0.975**	0.893	1	0.943	**0.971**	0.896	0.86	0.878	**0.883**
mn	0.979	0.92	0.948		0.907	0.98	0.942		0.754	0.92	0.829	
mr	0.98	0.98	0.98		0.98	0.96	0.97		0.956	0.86	0.905	
or	0.98	1	0.99		1	1	1		0.941	0.96	0.95	
rj	0.96	0.96	0.96		1	0.96	0.98		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 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	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		1	1	1	
gu	1	1	1		0.983	1	0.991		0.982	0.931	0.956	
hi	0.983	0.983	0.983		1	1	1		0.893	0.862	0.877	
kn	1	1	1		1	1	1		0.903	0.966	0.933	
ml	1	0.966	0.982	**0.988**	0.983	1	0.991	**0.985**	0.914	0.914	0.914	**0.945**
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	0.931	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	

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

The objective of the study was to observe the performance of the architectures on increasing the sample size. Since Bodo language has the minimum data (571 samples) among all the languages in the dataset, we perform our experiments till 571 samples.

A comparison of the results in Table 16, 17, and 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.

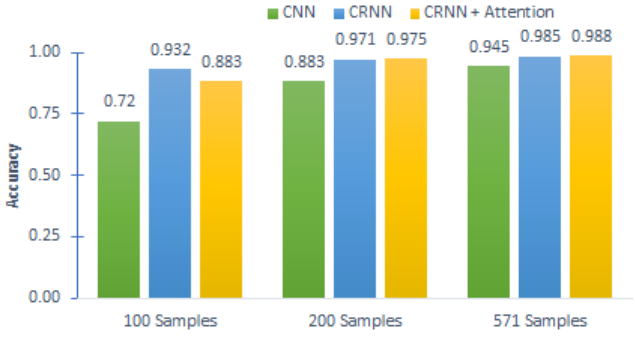

Figure 3: Comparison of model results for varying dataset size.

Figure 3 graphically shows the performance improvement over increasing data samples. The confusion matrices for the three architectures 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.

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 architecture efficiently identifies the language both in close language and noisy scenarios. We carried out extensive experiments and our architecture produced state-of-the-art results. Through our experiments, we have also shown our architecture's robustness to noise and its extensible 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 some additional computational overhead does not always result in improvement over CRNN. In future, we would like to extend our work by increasing the language classes with speech specimen recorded in different environments. We would also like to extend our work to check the usefulness of the proposed architecture 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.

Declaration of Competing Interest

None

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].

A. CNN architecture

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**29**	2	0	1	0	3	0	7	0	0	3	2	3	0.617	0.58	0.598
	bn	2	**40**	0	0	0	1	1	1	1	0	0	3	1	0.816	0.8	0.808
	bd	0	0	**43**	1	0	2	0	0	2	0	1	1	0	0.843	0.86	0.851
	gu	1	0	0	**38**	4	0	0	0	0	0	0	3	4	0.731	0.76	0.745
	hi	2	0	0	2	**35**	3	3	0	0	2	0	2	1	0.778	0.7	0.737
	kn	0	0	1	0	1	**37**	6	1	0	0	1	0	3	0.725	0.74	0.733
	ml	0	1	0	1	0	3	**41**	0	0	0	2	1	1	0.774	0.82	0.796
	mn	2	2	0	0	0	0	1	**38**	0	1	0	1	5	0.691	0.76	0.724
	mr	0	0	1	0	0	0	0	0	**42**	3	4	0	0	0.857	0.84	0.848
	or	0	1	1	0	0	0	0	0	3	**43**	1	1	0	0.811	0.86	0.835
	rj	2	2	3	1	3	0	1	2	1	2	**26**	7	0	0.605	0.52	0.559
	ta	5	0	0	3	2	0	0	3	0	0	2	**31**	4	0.564	0.62	0.590
	te	4	1	2	5	0	2	0	3	0	2	3	3	**25**	0.532	0.5	0.515

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**44**	1	0	0	0	0	0	1	0	0	1	3	0	0.8	0.88	0.838
	bn	1	**46**	0	0	0	0	0	1	0	0	0	2	0	0.92	0.92	0.92
	bd	0	0	**47**	0	0	1	0	1	0	0	1	0	0	0.94	0.94	0.94
	gu	1	1	0	**45**	1	0	0	0	0	0	0	0	2	0.918	0.9	0.909
	hi	0	1	0	1	**43**	1	1	2	0	0	1	0	0	0.956	0.86	0.905
	kn	0	0	1	1	0	**43**	2	1	0	0	0	1	1	0.878	0.86	0.869
	ml	1	0	0	0	0	1	**43**	2	1	2	0	0	0	0.896	0.86	0.878
	mn	2	0	0	0	0	0	1	**46**	0	0	0	1	0	0.754	0.92	0.829
	mr	0	0	2	0	0	0	1	1	**43**	1	2	0	0	0.956	0.86	0.905
	or	0	0	0	0	1	0	0	0	0	**48**	1	0	0	0.941	0.96	0.95
	rj	2	1	0	0	0	1	0	1	1	0	**43**	1	0	0.86	0.86	0.86
	ta	2	0	0	1	0	2	0	4	0	0	1	**40**	0	0.784	0.8	0.792
	te	2	0	0	1	0	0	0	1	0	0	0	3	**43**	0.935	0.86	0.896

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**58**	0	0	0	0	0	0	0	0	0	0	0	0	0.967	1	0.983
	bn	0	**58**	0	0	0	0	0	0	0	0	0	0	0	0.983	1	0.991
	bd	0	0	**56**	0	0	0	0	0	0	0	0	0	0	1	1	1
	gu	0	0	0	**54**	4	0	0	0	0	0	0	0	0	0.982	0.931	0.956
	hi	0	0	0	0	**50**	0	2	1	0	0	0	5	0	0.893	0.862	0.877
	kn	0	0	0	0	0	**56**	2	0	0	0	0	0	0	0.903	0.966	0.933
	ml	0	0	0	0	0	4	**53**	0	0	0	1	0	0	0.914	0.914	0.914
	mn	1	0	0	0	0	1	0	**54**	0	0	0	1	1	0.931	0.931	0.931
	mr	0	0	0	0	0	0	0	0	**55**	0	1	0	0	0.965	0.982	0.973
	or	0	0	0	0	1	0	0	0	1	**56**	0	0	0	1	0.966	0.982
	rj	1	1	0	0	0	0	1	1	0	0	**54**	0	0	0.9	0.931	0.915
	ta	0	0	0	1	1	0	0	1	1	0	3	**51**	0	0.879	0.879	0.879
	te	0	0	0	0	0	1	0	1	0	0	1	1	**54**	0.982	0.931	0.956

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

B. CRNN architecture

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**47**	0	0	0	0	0	0	1	0	0	1	1	0	0.839	0.94	0.887
	bn	0	**45**	0	0	0	1	0	0	0	0	0	3	1	0.957	0.9	0.928
	bd	0	0	**50**	0	0	0	0	0	0	0	0	0	0	0.962	1	0.98
	gu	0	0	0	**49**	0	0	0	0	0	0	0	0	1	1	0.98	0.99
	hi	0	0	0	0	**45**	2	2	0	0	0	0	0	1	0.957	0.9	0.928
	kn	1	0	0	0	0	**47**	2	0	0	0	0	0	0	0.94	0.94	0.94
	ml	0	0	0	0	1	0	**48**	0	0	0	1	0	0	0.923	0.96	0.941
	mn	1	2	0	0	0	0	0	**43**	0	1	1	0	2	0.935	0.86	0.896
	mr	0	0	0	0	0	0	0	0	**48**	0	2	0	0	0.98	0.96	0.97
	or	0	0	0	0	0	0	0	0	0	**50**	0	0	0	0.943	1	0.971
	rj	4	0	1	0	0	0	0	0	1	1	**42**	1	0	0.894	0.84	0.866
	ta	2	0	0	0	1	0	0	2	0	1	0	**44**	0	0.898	0.88	0.889
	te	1	0	1	0	0	0	0	0	0	0	0	0	**48**	0.906	0.96	0.932

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**47**	0	0	0	0	0	0	2	0	0	0	1	0	1	0.94	0.969
	bn	0	**48**	0	0	0	0	0	0	0	0	0	2	0	1	0.96	0.98
	bd	0	0	**49**	0	0	0	1	0	0	0	0	0	0	0.98	0.98	0.98
	gu	0	0	0	**50**	0	0	0	0	0	0	0	0	0	1	1	1
	hi	0	0	0	0	**49**	0	1	0	0	0	0	0	0	1	0.98	0.99
	kn	0	0	0	0	0	**49**	1	0	0	0	0	0	0	1	0.98	0.99
	ml	0	0	0	0	0	0	**50**	0	0	0	0	0	0	0.893	1	0.943
	mn	0	0	0	0	0	0	1	**49**	0	0	0	0	0	0.907	0.98	0.942
	mr	0	0	1	0	0	0	0	1	**48**	0	0	0	0	0.98	0.96	0.97
	or	0	0	0	0	0	0	0	0	0	**50**	0	0	0	1	1	1
	rj	0	0	0	0	0	0	0	0	1	0	**48**	1	0	1	0.96	0.98
	ta	0	0	0	0	0	0	0	2	0	0	0	**47**	1	0.904	0.94	0.922
	te	0	0	0	0	0	0	2	0	0	0	0	1	**47**	0.979	0.94	0.959

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**57**	0	0	0	0	0	0	0	0	0	0	1	0	0.983	0.983	0.983
	bn	0	**58**	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	bd	0	0	**56**	0	0	0	0	0	0	0	0	0	0	1	1	1
	gu	0	0	0	**58**	0	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	1	1	1
	kn	0	0	0	0	0	**58**	0	0	0	0	0	0	0	1	1	1
	ml	0	0	0	0	0	0	**58**	0	0	0	0	0	0	0.983	1	0.991
	mn	0	0	0	0	0	0	0	**57**	0	0	1	0	0	1	0.983	0.991
	mr	0	0	0	0	0	0	0	0	**56**	0	0	0	0	0.982	1	0.991
	or	0	0	0	0	0	0	0	0	1	**57**	0	0	0	1	0.983	0.991
	rj	1	0	0	0	0	0	1	0	0	0	**56**	0	0	0.918	0.966	0.941
	ta	0	0	0	1	0	0	0	0	0	0	4	**53**	0	0.964	0.914	0.938
	te	0	0	0	0	0	0	0	0	0	0	0	1	**57**	1	0.983	0.991

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

C. CRNN with Attention architecture

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**36**	0	0	0	0	0	0	1	0	0	6	6	1	0.766	0.72	0.742
	bn	1	**35**	0	0	0	0	0	1	0	0	0	13	0	0.875	0.7	0.778
	bd	0	0	**50**	0	0	0	0	0	0	0	0	0	0	1	1	1
	gu	0	0	0	**50**	0	0	0	0	0	0	0	0	0	0.943	1	0.971
	hi	0	0	0	0	**47**	1	1	0	0	0	0	0	1	0.959	0.94	0.95
	kn	0	0	0	0	0	**49**	0	0	0	0	0	1	0	0.961	0.98	0.97
	ml	0	0	0	0	1	1	**46**	0	0	0	2	0	0	0.958	0.92	0.939
	mn	5	3	0	1	0	0	0	**36**	0	1	1	0	3	0.878	0.72	0.791
	mr	0	0	0	0	0	0	0	0	**48**	0	2	0	0	0.906	0.96	0.932
	or	0	0	0	0	0	0	0	0	3	**47**	0	0	0	0.959	0.94	0.949
	rj	2	0	0	0	0	0	1	0	2	1	**43**	1	0	0.782	0.86	0.819
	ta	2	1	0	0	1	0	0	0	0	0	1	**44**	1	0.677	0.88	0.765
	te	1	1	0	2	0	0	0	3	0	0	0	0	**43**	0.878	0.86	0.869

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**48**	2	0	0	0	0	0	0	0	0	0	0	0	0.941	0.96	0.95
	bn	0	**50**	0	0	0	0	0	0	0	0	0	0	0	0.909	1	0.952
	bd	0	0	**48**	0	0	0	0	0	1	0	1	0	0	0.98	0.96	0.97
	gu	0	0	0	**50**	0	0	0	0	0	0	0	0	0	1	1	1
	hi	0	0	0	0	**49**	0	1	0	0	0	0	0	0	1	0.98	0.99
	kn	0	0	0	0	0	**49**	1	0	0	0	0	0	0	1	0.98	0.99
	ml	0	0	0	0	0	0	**50**	0	0	0	0	0	0	0.962	1	0.98
	mn	3	0	0	0	0	0	0	**46**	0	1	0	0	0	0.979	0.92	0.948
	mr	0	0	1	0	0	0	0	0	**49**	0	0	0	0	0.98	0.98	0.98
	or	0	0	0	0	0	0	0	0	0	**50**	0	0	0	0.980	1	0.99
	rj	0	2	0	0	0	0	0	0	0	0	**48**	0	0	0.96	0.96	0.96
	ta	0	1	0	0	0	0	0	0	0	0	1	**48**	0	1	0.96	0.98
	te	0	0	0	0	0	0	0	1	0	0	0	0	**49**	1	0.98	0.99

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

		Predicted													PPV	TPR	f1 Score
		as	bn	bd	gu	hi	kn	ml	mn	mr	or	rj	ta	te			
Actual	as	**58**	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	bn	0	**58**	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	bd	0	0	**56**	0	0	0	0	0	0	0	0	0	0	1	1	1
	gu	0	0	0	**58**	0	0	0	0	0	0	0	0	0	1	1	1
	hi	0	0	0	0	**57**	0	0	0	0	0	0	1	0	0.983	0.983	0.983
	kn	0	0	0	0	0	**58**	0	0	0	0	0	0	0	1	1	1
	ml	0	0	0	0	1	0	**56**	0	0	0	1	0	0	1	0.966	0.982
	mn	0	0	0	0	0	0	0	**58**	0	0	0	0	0	0.983	1	0.991
	mr	0	0	0	0	0	0	0	0	**56**	0	0	0	0	1	1	1
	or	0	0	0	0	0	0	0	0	0	**58**	0	0	0	1	1	1
	rj	0	0	0	0	0	0	0	1	0	0	**57**	0	0	0.919	0.983	0.95
	ta	0	0	0	0	0	0	0	0	0	0	4	**54**	0	0.964	0.931	0.947
	te	0	0	0	0	0	0	0	0	0	0	0	1	**57**	1	0.983	0.991

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

References

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

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. Tensorflow: A system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, USA. p. 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.M. (Eds.), Neural Information Processing, Springer International Publishing, Cham. pp. 880–889.

Chen, P., Heafield, K., 2020. Approaching neural chinese word segmentation as a low-resource machine translation task. CoRR abs/2008.05348. URL: https://arxiv.org/abs/2008.05348, arXiv:2008.05348.

Dehak, N., Kenny, P.J., Dehak, R., Dumouchel, P., Ouellet, P., 2011a. Front-end factor analysis for speaker verification. IEEE Transactions on Audio, Speech, and Language Processing 19, 788–798. doi:10.1109/TASL.2010.2064307.

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

Devlin, J., Chang, M., Lee, K., Toutanova, K., 2019. BERT: pre-training of deep bidirectional transformers for language understanding, in: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics. pp. 4171–4186. URL: https://doi.org/10.18653/v1/n19-1423, doi:10.18653/v1/n19-1423.

Draghici, A., Abeßer, J., Lukashevich, H., 2020. A Study on Spoken Language Identification Using Deep Neural Networks. Association for Computing Machinery, New York, NY, USA. p. 253–256. URL: https://doi.org/10.1145/3411109.3411123.

Ferrer, L., Scheffer, N., Shriberg, E., 2010. A comparison of approaches for modeling prosodic features in speaker recognition, in: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 4414–4417. doi:10.1109/ICASSP.2010.5495632.

Ganapathy, S., Han, K., Thomas, S., Omar, M., Segbroeck, M.V., Narayanan, S.S., 2014. Robust language identification using convolutional neural network features, in: Proc. Interspeech 2014, pp. 1846–1850. doi:10.21437/Interspeech.2014-419.

Gazeau, V., Varol, C., 2018. Automatic spoken language recognition with neural networks. International Journal of Information Technology and Computer Science (IJITCS) 10, 11–17.

Gelly, G., Gauvain, J.L., Le, V., Messaoudi, A., 2016. A Divide-and-Conquer Approach for Language Identification Based on Recurrent Neural Networks, in: Proc. Interspeech 2016, pp. 3231–3235. doi:10.21437/Interspeech.2016-180.

Gu, J., Wang, C., Zhao, J., 2019. Levenshtein transformer, in: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 11179–11189. URL: https://proceedings.neurips.cc/paper/2019/hash/675f9820626f5bc0afb47b57890b466e-Abstract.html.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Heracleous, P., Takai, K., Yasuda, K., Mohammad, Y., Yoneyama, A., 2018. Comparative study on spoken language identification based on deep learning, in: 2018 26th European Signal Processing Conference (EUSIPCO), pp. 2265–2269. doi:10.23919/EUSIPCO.2018.8553347.

Howard, J., Gugger, S., 2020. Fastai: A layered api for deep learning. Information 11. URL: https://www.mdpi.com/2078-2489/11/2/108, doi:10.3390/info11020108.

Joshi, M., Chen, D., Liu, Y., Weld, D.S., Zettlemoyer, L., Levy, O., 2020. Spanbert: Improving pre-training by representing and predicting spans. Trans. Assoc. Comput. Linguistics 8, 64–77. URL: https://transacl.org/ojs/index.php/tacl/article/view/1853.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1412.6980.

Kulkarni, R., Joshi, A., Kamble, M., Apte, S., 2022. Spoken language identification for native indian languages using deep learning techniques, in: Chen, J.I.Z., Wang, H., Du, K.L., Suma, V. (Eds.), Machine Learning and Autonomous Systems, Springer Singapore, Singapore. pp. 75–97.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2020. ALBERT: A lite BERT for self-supervised learning of language representations, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net. URL: https://openreview.net/forum?id=H1eA7AEtvS.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martinez, D., Gonzalez-Rodriguez, J., Moreno, P., 2014. Automatic language identification using deep neural networks, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5337–5341. doi:10.1109/ICASSP.2014.6854622.

Lozano-Diez, A., Zazo-Candil, R., Gonzalez-Dominguez, J., Toledano, D.T., Gonzalez-Rodriguez, J., 2015. An end-to-end approach to language identification in short utterances using convolutional neural networks, in: Proc. Interspeech 2015, pp. 403–407. doi:10.21437/Interspeech.2015-164.

Martínez, D., Plchot, O., Burget, L., Glembek, O., Matějka, P., 2011. Language recognition in ivectors space, in: Proc. Interspeech 2011, pp. 861–864. doi:10.21437/Interspeech.2011-329.

Mukherjee, S., Shivam, N., Gangwal, A., Khaitan, L., Das, A.J., 2019. Spoken language recognition using cnn, in: 2019 International Conference on Information Technology (ICIT), pp. 37–41. doi:10.1109/ICIT48102.2019.00013.

Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, Omnipress, Madison, WI, USA. p. 807–814.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Édouard Duchesnay, 2011. Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12, 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

Plchot, O., Matejka, P., Glembek, O., Fér, R., Novotný, O., Pesán, J., Burget, L., Brummer, N., Cumani, S., 2016. BAT system description for NIST LRE 2015, in: Rodríguez-Fuentes, L.J., Lleida, E. (Eds.), Odyssey 2016: The Speaker and Language Recognition Workshop, Bilbao, Spain, June 21-24, 2016, ISCA. pp. 166–173. URL: https://doi.org/10.21437/Odyssey.2016-24, doi:10.21437/Odyssey.2016-24.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al., 2011. The kaldi speech recognition toolkit, in: IEEE 2011 workshop on automatic speech recognition and understanding, IEEE Signal Processing Society.

Revay, S., Teschke, M., 2019a. Multiclass language identification using deep learning on spectral images of audio signals. CoRR abs/1905.04348. URL: http://arxiv.org/abs/1905.04348, arXiv:1905.04348.

Revay, S., Teschke, M., 2019b. Multiclass language identification using deep learning on spectral images of audio signals. URL: https://arxiv.org/abs/1905.04348, doi:10.48550/ARXIV.1905.04348.

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, in: Proc. Interspeech 2018, pp. 1798–1802. doi:10.21437/Interspeech.2018-69.

Sisodia, D.S., Nikhil, S., Kiran, G.S., Sathvik, P., 2020. Ensemble learners for identification of spoken languages using mel frequency cepstral coefficients, in: 2nd International Conference on Data, Engineering and Applications (IDEA), 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. URL: https://arxiv.org/abs/1803.09820, doi:10.48550/ARXIV.1803.09820.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826. doi:10.1109/CVPR.2016.308.

Takase, S., Kiyono, S., 2021. Lessons on parameter sharing across layers in transformers. CoRR abs/2104.06022. URL: https://arxiv.org/abs/2104.06022, arXiv:2104.06022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Venkatesan, H., Venkatasubramanian, T.V., Sangeetha, J., 2018. Automatic language identification using machine learning techniques, in: 2018 3rd International Conference on Communication and Electronics Systems (ICCES), pp. 583–588. doi:10.1109/CESYS.2018.8724070.

Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y., 2020. LUKE: deep contextualized entity representations with entity-aware self-attention, in: Webber, B., Cohn, T., He, Y., Liu, Y. (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, Association for Computational Linguistics. pp. 6442–6454. URL: https://doi.org/10.18653/v1/2020.emnlp-main.523, doi:10.18653/v1/2020.emnlp-main.523.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V., 2019. Xlnet: Generalized autoregressive pretraining for language understanding, in: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 5754–5764. URL: https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E., 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, Association for Computational Linguistics, San Diego, California. pp. 1480–1489. URL: https://aclanthology.org/N16-1174, doi:10.18653/v1/N16-1174.

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, 1–17. URL: https://doi.org/10.1371/journal.pone.

0146917, doi:10.1371/journal.pone.0146917.
Zissman, M., 1996. Comparison of four approaches to automatic language identification of telephone speech. IEEE Transactions on Speech and Audio Processing 4, 31–. doi:10.1109/TSA.1996.481450.