Indian Language Identification using Semi-Supervised Learning

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Samarjit Karmakar

(Roll No. 167256)

under the guidance of

Dr. P. Radha Krishna

COMPUTER SCIENCE AND ENGINEERING
National Institute of Technology
Warangal – 500604

June 2020

BONAFIDE CERTIFICATE

This is to certify that the project titled **Indian Language Identification using Semi-Supervised Learning** is a bonafide record of the work done by

Samarjit Karmakar

(Roll No. 167256)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the **National Institute of Technology, Warangal**, during the year 2019-20.

Dr. P. Radha Krishna

Guide

Head of the Department

ABSTRACT

Identification of the language spoken from speech utterances is an interesting task because of the diversity associated with different languages and human voices. Indian languages have diverse origins and identifying them from speech utterances would help several language recognition, translation and relationship mining tasks. The current approaches for tackling the problem of languages identification in the Indian context heavily use feature engineering and classical speech processing techniques. This is a bottleneck for language identification systems, as we require to exploit necessary features in speech, required for machine identification, which are learnt by a probabilistic framework, rather than handcrafted feature engineering. In this paper, we tackle the problem of language identification using latent representations learnt from speech using Variational Autoencoders (VAEs) and leverage the representations learnt to train sequence models. VAEs are deep probabilistic models which learn semantically meaningful representation from data without any supervised training. They are highly inspired by Autoencoders which compress data to lower dimensional latent space and reconstruct it back using an encoder and a decoder. Our framework attains an accuracy of 89% in the identification of 8 well known Indian languages (namely Tamil, Telugu, Punjabi, Marathi, Gujarati, Hindi, Kannada and Bengali) from the CMU Indic Speech Database. The presented approach can be applied to several scenarios for speech processing by employing representation learning and leveraging them for sequence models.

TABLE OF CONTENTS

CHAPTER 1

Introduction

Information in the real world comes in several modalities. Several intelligent agents process information from various such modalities in order to perform various downstream tasks. We focus on learning representations from various such modalities and use them for several other learning tasks. The modalities of special interest are **images**, **text**, **speech** and **audio**. In order to perform multi-modal classification tasks, it is important to learn joint representations for all such modalities.

Language identification refers to the task of identifying the language being spoken when given a speech utterance. Several intelligent agents rely heavily on language identification systems for subsequent speech recognition and processing tasks. This problem is particularly interesting and important in the Indian context, given the diverse nature of Indian languages and the number of states the country has. Several Indian languages suffer regional bias and each language has its own dialect as one travels within each state (region) of the country. In this scenario, language identification would be a challenging task for traditional language identification systems which heavily rely on feature engineering from speech.

Several of the current Indian language identification systems rely on handcrafted features, which end up serving as a bottleneck to such systems. Such systems would greatly benefit by learning necessary features in speech utterances using a probabilistic framework and leveraging these representations for training deep sequence models. A few deep neural network based models have also been proposed. The authors in [1] perform posterior extraction using convolutional neural networks (CNNs), and an i-vector based system for subsequent language recognition. The authors in [2] use an end-to-end deep neural network with attention mechanism for Indian language identification.

The field of representation learning has matured with the rise of several deep learning models. Specifically, Autoencoders can process raw data and produce latent representations for the same by using an *encoder-decoder* framework for reconstruction. They compress raw data into a latent representation space and reconstruct the same data from those latent representations. In turn, they learn efficient latent representations for raw data in an unsupervised manner. The *encoder* encodes raw data into the latent representation space and the *decoder* decodes these latent representations, reconstructing the raw data from them. Variational autoencoders (VAEs) [3] are deep probabilistic generative models where we consider

both reconstruction along with the KL divergence between the encoder's distribution (latent distribution) and a prior distribution (multivariate standard Normal). This allows better feature learning in a stochastic low dimensional space and generation by sampling from the prior distribution.

1.1 Objective

The main objective of this work is to exploit the representation learnt by a VAE trained on speech segments to train sequence models. There are several widely used for natural language processing. These models use distributed word-representations (word embeddings) [4] which model the words in a continuous vector space. We use the latent feature representations learnt by a VAE in the stochastic low dimensional latent representation space in a manner similar to how distributed word representations, obtained by pre-training large corpuses in an unsupervised manner, are used in natural language processing to train sequence models.

1.2 Problem Statement

We model the task of Language Identification (LID) as a classification problem.

Let $D = \{x_i, y_i\}_{i=1,2,...,|D|}$ be some finite amount of data where each x_i is a speech utterance and y_i is the corresponding language label of x_i. Let $F : X \times M \rightarrow Y$ be a parameterized function where X is a set of all possible speech utterances, M is a set of all possible model parameters and Y is a set of all possible language labels. Let $J : Y \times Y \rightarrow \mathbb{R}^+$ be some performance measure (or distance) given two language labels. Let $y_1, y_2 \in Y$, then the lower the value of $J(y_1, y_2)$, the closer the language labels y_1 and y_2 are. The task is to find the optimum $\theta \in M$ such that, $J(F(x_i, \theta), y_i)$ is minimized for all $i \in \{1, 2, ..., |D|\}$.

In this work, we present a framework for Indian language identification by pre-training a probabilistic framework for representation learning and leveraging these representation for training sequence models for classification.

The contributions of this work are as follows:

1. Exploration of the VAE architecture on Indian speech utterances for representation learning. The architecture considers the property of translational invariance in speech. The VAE encoder produces a sequence of representation vectors for each speech utterance.

2. Learning temporal characteristics of Indian speech utterances using state-of-the-art sequence models used in natural language processing which directly work on the representation vectors learnt by the VAE.

1.3 Organization of the Thesis

The thesis is organized as follows.

- In Chapter 2, we present the related work.
- Chapter 3 discusses some major components which are helpful for our framework and a detailed description of it.
- Chapter 4 shows both quantitative and qualitative description of the experiments and results.
- Chapter 5 gives concluding remarks and notes on future endeavours.

1.4 Publication based on the Thesis

- S. Karmakar, P. Radha Krishna, "Leveraging Latent Representations of Speech for Indian Language Identification", in ICON 2020 : 17th International Conference on Natural Language Processing at IIT Patna.

CHAPTER 2

Related Work

Language identification from speech has been deeply studied by various research communities. Prosodic, phoenetic and phonotactic feature based approaches for identifying language is studied in [5] and [6]. Many such classical feature engineering based methods require a lot of domain knowledge. With the rise of deep learning and neural networks, automatic feature and representation learning has greatly outperformed all such methods. In these, the representational features are automatically learnt from data and hence, the requirement of domain knowledge is very less.

Language identification using Deep Convolutional Recurrent Neural Networks is studied in [7]. They use non-overlapping segments of Mel spectrograms of speech which are passed through a Convolutional Neural Network and then the features maps are passed through a Long Short Term Memory (LSTM). The final hidden state of the LSTM is used for classification. The CNN captures the spatial features, whereas the LSTM captures the temporal features.

For long speech utterances, Recurrent Neural Networks, can capture the temporal aspect of speech utterances and this was considered in [8]. Specifically, Long Short Term Memory (LSTM) networks [9] perform well on long speech utterances as they have a specific gated memory structure to counter the temporal length of input sequences.

The authors in [10] give an attention based 1D-CNN for the task of language identification directly from raw audio. This attention greatly enhances the performance of neural network based approaches.

Time Delay Neural Networks (TDNN) on Mel spectrograms of speech is studied in [11]. An elaborate analysis is given in the paper regarding the performance of deep architectures in comparison to shallow ones.

Indian language identification using deep learning based models have been studied in [12], [2], [13] and [14]. Deep neural network based systems take in the speech utterances at each frame, classification performed frame-wise, and this may be considered as a drawback. A deep neural network with attention mechanism was considered in [2]. This architecture applies attention to specific parts of the input sequence, whilst memorizing important features in long temporal sequences. A 39-dimensional MFCC is considered by the authors, each for 5 second chunks of the input sequence, which are passed through a regular DNN to compute hidden

layer representations. An attention mechanism is applied over this to memorize the temporal aspect and summarize the features in the whole speech utterance, giving a single context vector and this vector is subsequently passed to a classifier. Attention based Residual-Time Delay Neural Network (RES-TDNN) is studied in [15], which further improves over trying to capture the long range temporal dependencies.

Most of the works try to tackle either the task of feature extraction before classification, using classical speech processing or DNN based methods, and try to capture long range dependencies using RNNs or attention mechanisms. Indian language identification suffer from another major caveat. Data is scarce and languages are diverse. If our models have a large number of parameters, it becomes difficult to optimize them.

Hence, we first perform representation learning on speech using unsupervised learning using a Convolutional Variational Autoencoder (VAE). Then, we perform supervised learning of sequence models on the representations learnt by the VAE. As a results, we have to deal with less parameters at each optimization step. The unsupervised representation learning parameters are optimized in a different step from when the supervised sequence learning parameters are optimized.

CHAPTER 3

Underlying Components and Proposed Framework

Our goal is to learn latent representations from speech and use these representations to train sequence models for classification. We use Variational Autoencoders (VAEs) for representation learning on small segments of Mel spectrograms of speech utterances (40 Mel-scale filter banks).

The Mel spectrogram is obtained by taking the Fourier transform of the signal, followed by mapping the powers of the obtained spectrum onto the Mel scale. The Mel-frequency scale resembles the resolution of the human auditory system. The segmentation is performed along the time axis. The model is trained in a similar manner adopted in [16]. After pre-training the VAE, the encoder's latent distribution is able to encode Mel spectrograms of speech segments into a latent representations space. We use a sequence of such latent representation for each segmented Mel spectrogram of speech utterance as input to sequence models. The VAE captures important representational features for each segment of the speech utterance and the sequence model captures the temporal aspect of each speech utterance. Figure 3.1 shows a high-level view of our language identification system.

3.1 Preprocessing

The Mel spectrogram of a speech signal is obtained by applying a non-linear transform (Mel scaling) to the frequency axis of Short-Time Fourier Transform (STFT).

Suppose $x(t)$ be the input raw audio signal and w be a window.

The $STFT$ function is given by,

$$STFT\{x(t), w\} = \int_{-\infty}^{+\infty} x(t)w(t-\tau)e^{-j\omega t}dt \tag{3.1}$$

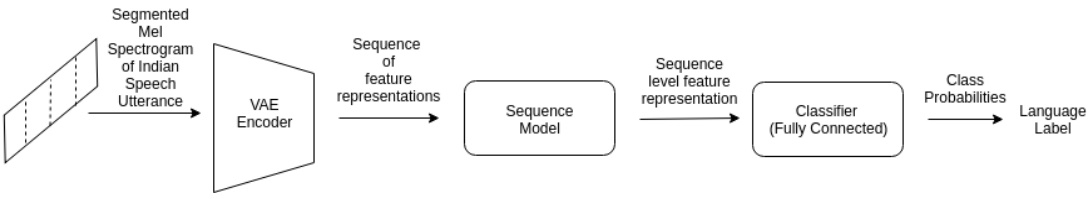


Figure 3.1: A framework for Indian language identification.

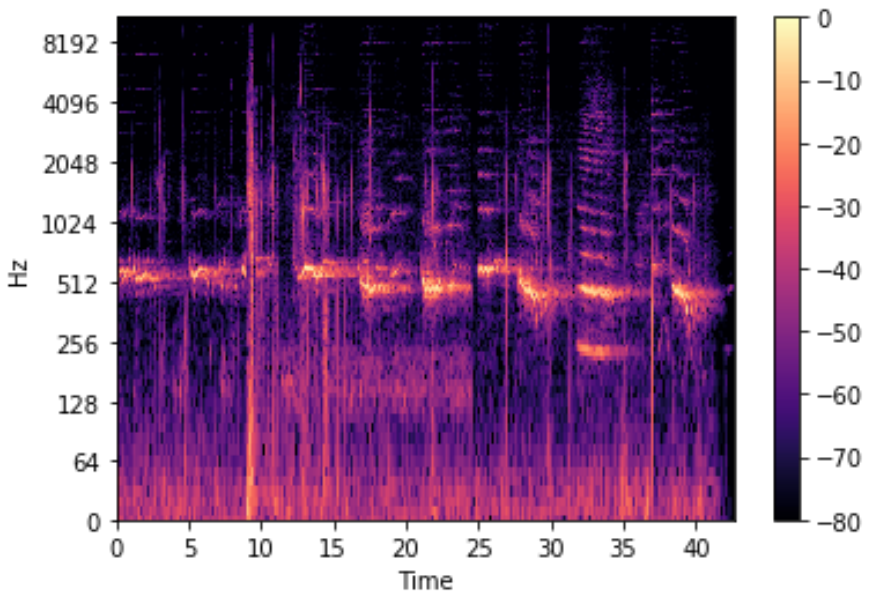


Figure 3.2: An example of a Mel Spectrogram for an audio sample. The depiction along the right axis is the Mel scale.

The Mel function for a frequence f is given by,

$$Mel(f) = \frac{1000}{\log 2}\left(1 + \frac{f}{1000}\right) \tag{3.2}$$

The function $Mel(\cdot)$ is applied along the frequency axis of the STFT magnitude, $|STFT\{x(t), w\}|$.

3.2 Variational Autoencoder

A Variational Autoencoder (VAE) comprises of two neural networks, the encoder $q_\theta(z|x)$ and the decoder $p_\phi(x|z)$. The encoder, parameterized by θ, takes in the input observation (x) and encodes it into a representation (z) sampled stochastically from the distribution of μ and σ (Gaussian parametric layers of the encoder). The decoder, parameterized by ϕ, takes in the representation (z) and decodes it back into the input observation (x). The loss function (L_{vae}) minimizes a joint objective of two losses: reconstruction loss and KL divergence loss.

$$L_{vae} = -\mathbb{E}_{q_\theta(z|x)}(p_\phi(x|z)) + \mathbb{KL}(q_\theta(z|x)||p(z)) \tag{3.3}$$

Here, $p(z)$ is the prior distribution (multivariate standard Normal).

Let us develop some insight into a VAE to fully appreciate it and understand how L_{vae} comes into being.

A VAE maps an input x into a latent distribution $p(z)$. Knowing the parameters

of this distribution helps us sample from it to generate a new data point. Hence, a VAE can be exactly given by $p(z)$, $p(z|x)$ and $p(x|z)$. $p(z)$ can be a chosen prior. Hence, the parameters for $p(z|x)$ and $p(x|z)$ must be learnt.

We are given real samples from dataset D. The optimum parameters, $\hat{\phi}$, can be calculated by the following.

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}} \sum_{x \in D} \log p_\phi(x) = \underset{\phi}{\operatorname{argmax}} \sum_{x \in D} \log \left(\int_z p_\phi(x|z) p_\phi(z) dz \right) \tag{3.4}$$

Calculating this integral is intractable as it involves integration over the all values of z. This is approximated in a VAE. We use a probabilistic encoder, $q_\theta(z|x)$, in order to approximate $p_\phi(z|x)$, and restricting ourselves rather than spanning the whole latent space.

KL Divergence (also known as relative entropy) is a measure of how distribution p diverges from distribution q, denoted by $\mathbb{KL}(p||q)$. The following equation gives an expression for KL Divergence between two distributions.

$$\mathbb{KL}(p||q) = \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx = \mathbb{E}_{x \sim p} \left[\log \left(\frac{p(x)}{q(x)} \right) \right] \tag{3.5}$$

The KL Divergence between $q_\theta(z|x)$ and $p_\phi(z|x)$ must be minimized. Now, by some algebra, we may show the following,

$$\begin{aligned}
&\mathbb{KL}(q_\theta(z|x)||p_\phi(z|x)) \\
&= \int_z \left(q_\theta(z|x) \log \left(\frac{q_\theta(z|x)}{p_\phi(z|x)} \right) \right) dz \\
&= \int_z \left(q_\theta(z|x) \log \left(\frac{q_\theta(z|x) p_\phi(x)}{p_\phi(x|z) p_\phi(z)} \right) \right) dz \quad \text{(By Bayes Theorem)} \\
&= \int_z q_\theta(z|x) \log p_\phi(x) dz + \int_z q_\theta(z|x) \log \left(\frac{q_\theta(z|x)}{p_\phi(x|z) p_\phi(z)} \right) dz \\
&= \log p_\phi(x) + \int_z q_\theta(z|x) \log \left(\frac{q_\theta(z|x)}{p_\phi(x|z) p_\phi(z)} \right) dz \quad (\because \int_z q_\theta(z|x) dz = 1) \\
&= \log p_\phi(x) + \int_z q_\theta(z|x) \log \left(\frac{q_\theta(z|x)}{p_\phi(x|z) p_\phi(z)} \right) dz \\
&= \log p_\phi(x) + \mathbb{E}_{z \sim q_\theta(z|x)} \left[\log \left(\frac{q_\theta(z|x)}{p_\phi(z)} \right) \right] - \mathbb{E}_{z \sim q_\theta(z|x)} \left[\log p_\phi(x|z) \right] \\
&= \log p_\phi(x) + \mathbb{KL}(q_\theta(z|x)||p_\phi(z)) - \mathbb{E}_{z \sim q_\theta(z|x)} \left[\log p_\phi(x|z) \right]
\end{aligned} \tag{3.6}$$

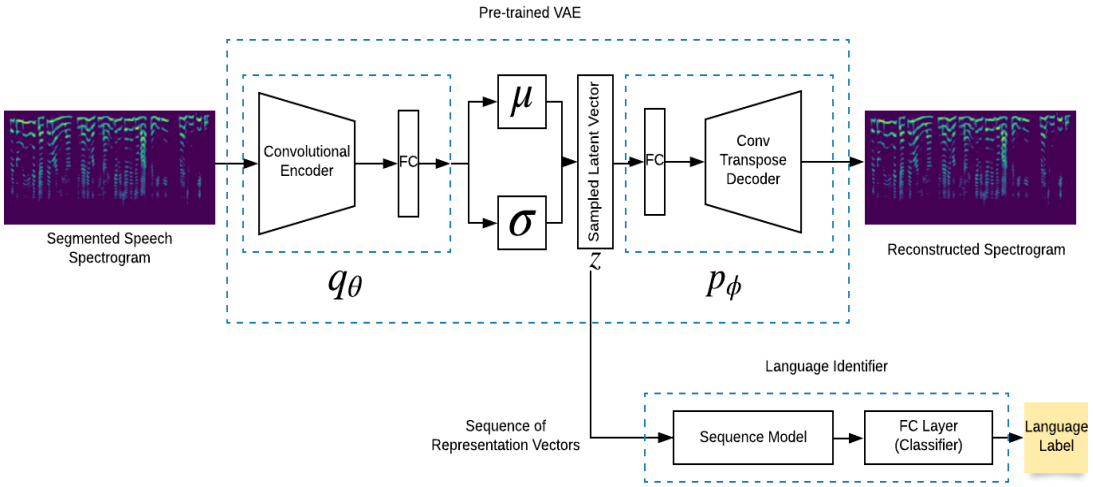


Figure 3.3: A view of the VAE architecture with the language identifier.

Rearranging, we get,

$$-\log p_\phi(x) + \mathbb{KL}(q_\theta(z|x)||p_\phi(z|x)) = -\mathbb{E}_{z\sim q_\theta(z|x)}\Big[\log p_\phi(x|z)\Big] + \mathbb{KL}(q_\theta(z|x)||p_\phi(z)) \tag{3.7}$$

We see that minimizing the left hand side is exactly our objective. The right hand side exactly corresponds to the loss function used in VAE. Hence, we may clearly define our objective as,

$$\hat{\theta}, \hat{\phi} = \underset{\theta,\phi}{\operatorname{argmin}} \left(-\mathbb{E}_{z\sim q_\theta(z|x)}\Big[\log p_\phi(x|z)\Big] + \mathbb{KL}(q_\theta(z|x)||p_\phi(z)) \right) \tag{3.8}$$

Now, $z \sim q_\theta(z|x)$ is a stochastic operation and computing gradients to backpropagate through such an operating is not possible. Hence, to be able to backpropagate up to $q_\theta(z|x)$ to facilitate end to end training, the re-parameterization trick was introduced. Suppose, we choose a Gaussian prior. We may write z as $z = \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$. Hence, $q_\theta(z|x)$ would give us the deterministic parameter for z, namely μ and σ, and stochastic parameter, ϵ, is sampled.

We use similar hyper-parameters as used in [16]. The encoder contains 3 convolutional layers, followed by a fully connected layer and two Gaussian parametric layer (one for mean and another for log variance). The decoder contains an initial fully connected layer which takes in z, followed by another fully connected layer and 3 transpose-convolutional layers.

Figure 3.3 shows a view of the VAE architecture along with the language identifier (successor to the VAE pre-training phase).

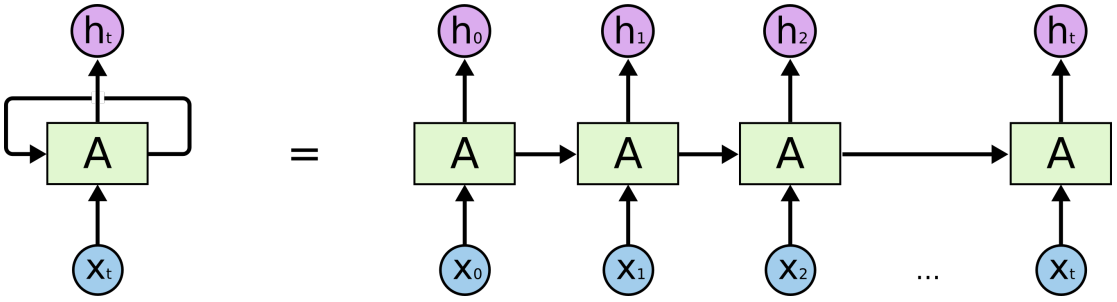


Figure 3.4: An unwinded Recurrent Neural Network (RNN).

3.3 Sequence Models

We experiment with four sequence models in this paper. The sequence models capture the temporal aspect of the speech utterance from the given sequence of representation vectors.

For each segment of the Mel spectrogram of speech utterance, the VAE encoder produces a vector in $\mathbb{R}^n$, where n is the dimension of the representation space. We pass the sequence of these vectors for each segmented speech utterance to sequence models for classification. The input to the sequence models are a sequence representation vectors of size 128 units, i.e the dimension of the representation space of the VAE. We compute the maximum length of the sequences produced on segmentation of each speech utterance, and apply zero-padding vectors to each sequence to produce uniform length sequences.

The sequence models considered in this work are:

1. **Bi-directional LSTM with Self-Attention**
2. **Bi-directional LSTM with Soft-Aligned Attention**
3. **Recurrent Convolutional Neural Networks**
4. **Transformer**

All the models are trained separately on the same representation vectors obtained from the pre-trained VAE encoder.

3.3.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNN) are special classes of Neural Networks built to handle sequence information. Let $x = \{x_t\}_{t=0,1,2,..}$ define an input sequence. An RNN maintains a hidden state, $h = \{h_t\}_{t=0,1,2,..}$, which gets updated at each time step, t, based on some form of non-linear combination of the input at current time step, x_t, and the hidden state at previous time step, h_{t-1}.

Let us suppose h is a sequence of vectors in $\mathbb{R}^m$, i.e. $h_t \in \mathbb{R}^m$, and x is a sequence

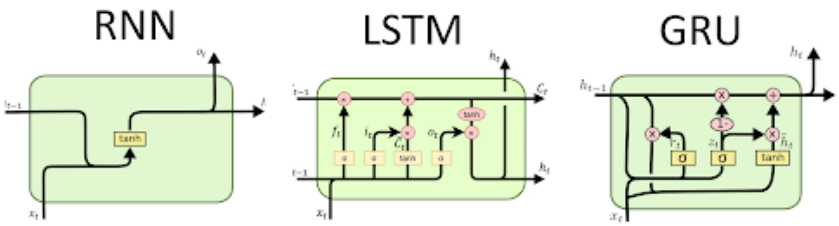


Figure 3.5: Gating mechanisms for RNN, LSTM and GRU.

of vectors in $\mathbb{R}^n$, i.e. $x_t \in \mathbb{R}^n$. Let W be a matrix in $\mathbb{R}^{m \times n}$, U be a matrix in $\mathbb{R}^{m \times m}$ and b be a bias vector in $\mathbb{R}^m$.
A Vanilla RNN can be updated using the following equation:

$$h_t = \sigma(W x_t + U h_{t-1} + b) \tag{3.9}$$

The parameters (weight matrices: W and U, bias vector: b) must be learnt by backpropagation through time [17]. An unwinded RNN is shown in Figure 3.4.
A shortcoming of Vanilla RNNs is preserving "long-term dependencies" primarily due to the "vanishing gradient" problem. Gated Recurrent Units (GRUs) and Long Short Term Memory (LSTM) Networks were created to solve several of the shortcomings of Vanilla RNNs.

3.3.2 Long Short Term Memory (LSTM) Networks

Long Short Term Memory (LSTM) networks are sequence models which learn temporal characteristics and contextual information from sequences. They are able to preserve long-term dependencies better than Vanilla RNNs. An LSTM is updated using the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.10}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.11}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3.12}$$

$$c_t = c_{t-1} \odot f_t + \tanh(W_u x_t + U_u h_{t-1} + b_u) \odot i_t \tag{3.13}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.14}$$

Here, i, f, c and o are the activations of input gate, forget gate, cell state and output state respectively. x is the input sequence, W and U are weight matrices and b is the bias. Each vector is sub-scripted by t to denote the t'th time step. The descriptive notations for inputs and parameters are similar to the RNN defined in

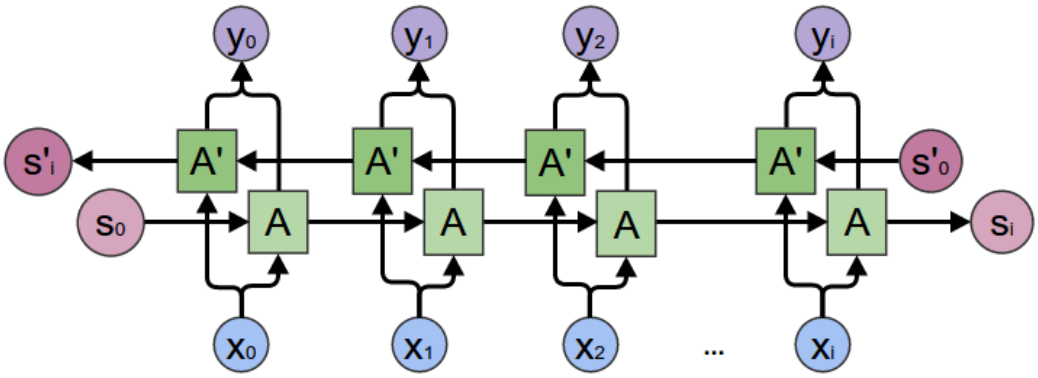


Figure 3.6: A schematic diagram for understanding bidirectional sequence models.

the previous section.

3.3.3 Gated Recurrent Units (GRUs)

A Gated Recurrent Unit (GRU) is similar to an LSTM with the following differences:

- The forget and input gates are combined into a single update gate.
- The cell state and hidden state are merged into a single vector.

A GRU is updated using the following equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{3.15}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{3.16}$$

$$\tilde{h_t} = \tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \tag{3.17}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{3.18}$$

A diagram illustrating gating mechanisms for RNN, LSTM and GRU is shown in Figure 3.5.

3.3.4 Bi-directional RNNs and LSTMs

LSTMs have a single shortcoming, they make use of solely the previous context. Bidirectional LSTMs (Bi-LSTMs) make use of both the previous context as well as future context by iterating through the sequence in both directions to compute the hidden state vectors. We pass the forward and backward sequence through the LSTM assigning different weights and biases for each direction. This is used to compute two separate sets of activations for the sequence in forward and backward directions. A bidirectional sequence model architecture is illustrated in Figure 3.6.

The following mentions some fundamentals regarding a Bidirectional RNN which gives us understanding to develop on a Bidirectional LSTM.
Let $x = \{x_t\}_{t=0,1,2,..,T-1}$ define an input sequence of finite length T. An Bidirectional RNN maintains two hidden states, $\overrightarrow{h}$ and $\overleftarrow{h}$.

Algorithm 1 Bidirectional RNN Updates

Input: $x = \{x_t\}_{t=0,1,2,..,T-1}$, Parameters of Bi-RNN ($\overrightarrow{W}$, $\overrightarrow{U}$, $\overrightarrow{b}$, $\overleftarrow{W}$, $\overleftarrow{U}$, $\overleftarrow{b}$)
Output: $\overrightarrow{h}$ and $\overleftarrow{h}$

Suppose $\overrightarrow{h_{-1}}$ and $\overleftarrow{h_T}$ are 0 vectors.

1: **for** $t \leftarrow 0$ to $T-1$ **do**
2: $\quad \overrightarrow{h_t} \leftarrow \sigma(\overrightarrow{W}x_t + \overrightarrow{U}\overrightarrow{h_{t-1}} + \overrightarrow{b})$
3: **end for**
4: **for** $t \leftarrow T-1$ to 0 **do**
5: $\quad \overleftarrow{h_t} \leftarrow \sigma(\overleftarrow{W}x_t + \overleftarrow{U}\overleftarrow{h_{t+1}} + \overleftarrow{b})$
6: **end for**
7: **return** $[\overrightarrow{h} : \overleftarrow{h}]$

A Bidirectional RNN can be updated using the following equation:

$$\overrightarrow{h_t} = \sigma(\overrightarrow{W}x_t + \overrightarrow{U}\overrightarrow{h_{t-1}} + \overrightarrow{b}) \tag{3.19}$$

$$\overleftarrow{h_t} = \sigma(\overleftarrow{W}x_t + \overleftarrow{U}\overleftarrow{h_{t+1}} + \overleftarrow{b}) \tag{3.20}$$

The descriptive notations for inputs and parameters are similar to the RNN defined in the previous section. This is illustrated better in the forward pass updates for Bi-RNN in Algorithm 1.

3.3.5 Bi-directional LSTM with Self-Attention

The sequence of representation vectors is passed through a bi-directional LSTM (bi-LSTM) [18] with a hidden state of 100 memory units. A self-attention mechanism is adopted which gives attention to specific parts of the input sequence, giving a attention weight matrix of the input sequence. This mechanism is similar to the attention layer in [19]. The mechanism takes in the hidden states of the bi-LSTM at each time step. The attention weights are calculated for 30 sequence vectors, each giving attention to some specific part of the input sequence. The attention weights are applied to the output of the bi-LSTM, resulting in a matrix of hidden states giving attention to specific parts of the sequence. This is then flattened and passed through subsequent fully connected neural network layers to produce the output class logits.

3.3.6 Bi-directional LSTM with Soft-Aligned Attention

We apply a similar attention mechanism as that adopted in the encoder of [20]. We pass the sequence of representation vectors through a bi-LSTM with a hidden state of 100 memory units similar to the previous model. The difference lies in the attention mechanism adopted. We calculate a soft-alignment score between each of the hidden states of the bi-LSTM at each time step and the last hidden state. This gives the soft-aligned attention weights, which are applied to the output of the bi-LSTM at each time step to produce a single final hidden state vector. This is then passed through subsequent fully connected neural network layers to produce the output class logits.

3.3.7 Recurrent Convolutional Neural Networks

We apply a similar architecture as that adopted in [21]. We pass the sequence of representation vectors through a bi-LSTM with a hidden state of 100 memory units similar to the previous model. The hidden state at each time step is concatenated to the corresponding input representation. This is passed through a fully connected layer which maps the concatenated vector back to the hidden state size. The architecture takes care of the right context and left context as it is a bi-LSTM, which takes care of information and representation flow in the forward and reverse direction of the speech utterance. We perform max-pooling across all the sequences and pass the output through subsequent fully connected neural network layers to produce the output class logits.

3.3.8 Transformer

We use the encoder of the Transformer architecture similar to that adopted in [22]. We do not apply the positional encodings and masking mechanisms. The sequence of representation vectors are directly passed through two encoder layers, each of which comprise of self attention and position-wise feed-forward layers. The hidden dimensions of the position-wise feed-forward layers are 100 units. The set of hyper-parameters adopted similar to [22] are ($N = 2$, $d_{model} = 128$, $d_q = d_k = d_v = 32$, $p_{dropout} = 0.3$). The output of the encoder is flattened and passed through subsequent fully connected neural network layers to produce the output class logits.

3.4 Summary

Algorithm 2 VAE-Seq Language Identification Training

Input: Dataset D, VAE parameters (ϕ, θ), Sequence Model parameters (ξ)

Output: Optimized parameters ϕ, θ and ξ

1: Initialize parameters ϕ, θ and ξ
2: **repeat**
3: Sample mini-batch $M = \{x_i\}_{i=1,2,...|M|}$ of audio spectrograms from D by segmenting spectrogram of audio clips
4: Forward pass mini-batch M through VAE
5: Update parameters ϕ and θ using $\nabla_{\phi,\theta} L_{vae}(\phi, \theta, M)$
6: **until** convergence of ϕ and θ
7: **repeat**
8: Sample mini-batch $M = \{x_i, y_i\}_{i=1,2,...|M|}$ from D where x_i is a sequence of segmented spectrograms and y_i is label for i'th sample
9: Forward pass each sample in x_i through VAE encoder parameterized by θ to convert $\{x_i, y_i\}_{i=1,2,...|M|}$ to $\{v_i, y_i\}_{i=1,2,...|M|}$ where each v_i is a sequence of representation vectors for x_i
10: Forward pass each v_i through Sequence Model parameterized by ξ to give predicted labels $\{\hat{y}_i\}_{i=1,2,...|M|}$
11: Update parameters ξ using $\nabla_{\xi} L_{seq}(\xi, \{\hat{y}_i\}_{i=1,2,...|M|}, \{y_i\}_{i=1,2,...|M|})$
12: **until** convergence of ξ
13: **return** ϕ, θ and ξ

An illustration for training the VAE and Sequence Model is given in Algorithm 2.

Algorithm 3 VAE-Seq Language Identification

Input: Speech clip x, Optimized VAE parameters (ϕ, θ), Optimized Sequence Model parameters (ξ)

Output: Language label y

1: Forward pass the segmented spectrogram of x through the VAE encoder having optimized parameters θ to obtain a sequence of representation vectors v
2: Forward pass v through the Sequence Model having optimized parameters ξ to obtain the language label y
3: **return** y

An illustration for language identification using VAE and Sequence Model is given in Algorithm 3.

CHAPTER 4

Experimental Results

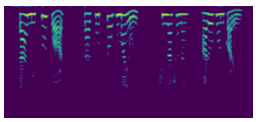
4.1 Dataset, Hyperparameters and Accuracies

We pre-train the VAE on segmented speech utterances from the CMU/IIITH Indic Speech Database [23] [24]. The database contains raw speech utterances in 8 languages, namely Bengali, Gujarati, Hindi, Kannada, Marathi, Punjabi, Tamil and Telugu. The raw audio is converted to a Mel spectrogram (with 40 Mel filter banks and FFT window of size 1024 units). The Mel spectrogram is then segmented along the time axis, with an overlapping window of 4 units, producing a sequence of spectrograms, each of dimensions (40×20). This produces smaller utterances for each speech utterance. The VAE is then trained to learn representations for these smaller utterances in an unsupervised manner.

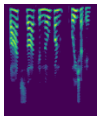
Similar pre-processing is applied on each speech utterance, prior to training the sequence models, to create a sequence of spectrograms, each of dimensions (40×20), which are then passed through the pre-trained VAE encoder to produce a sequence of representation vectors, each of size 128 units.

The sequence models are trained on the above pre-processed speech data. We use cross-entropy between the output logits and the labels as the loss metric, which is minimized using Adam optimizer [25], with learning rate of 10^{-4}, β_1 of 0.999, β_2 of 0.99 and weight decay of 10^{-5}.

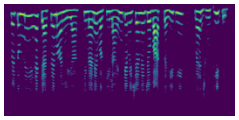
The confusion matrices obtained for each sequence model are shown in Figure 4.3.



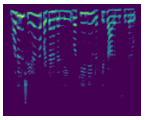
(a) Bengali



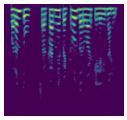
(b) Gujarati



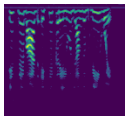
(c) Hindi



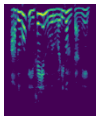
(d) Kannada



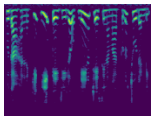
(e) Marathi



(f) Punjabi

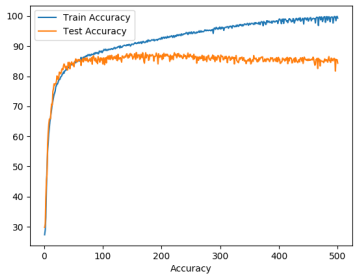


(g) Tamil

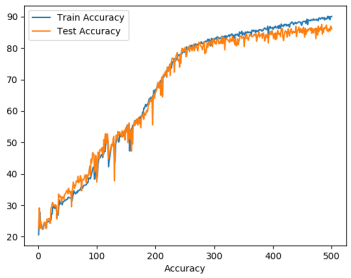


(h) Telugu

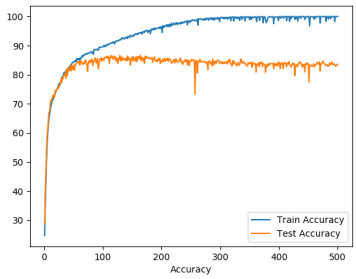
Figure 4.1: Sample Mel spectrograms of various speech utterances of specific languages from the CMU Indic dataset.



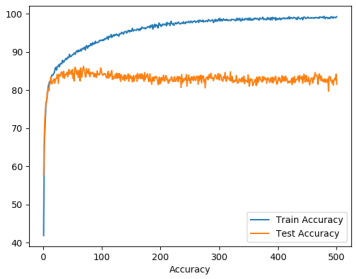
(a) Bi-LSTM with Self Attention



(b) Bi-LSTM with Soft Aligned Attention



(c) RCNN



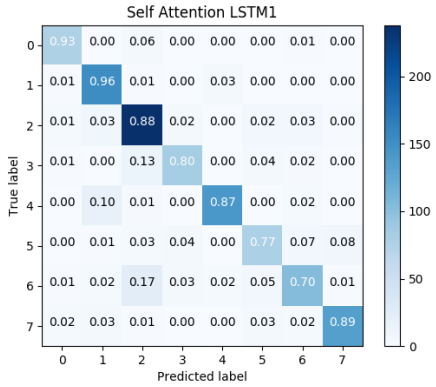
(d) Transformer

Figure 4.2: The training and testing accuracy plots. The horizontal axis denotes number of epochs and the vertical axis denotes the percent accuracy.

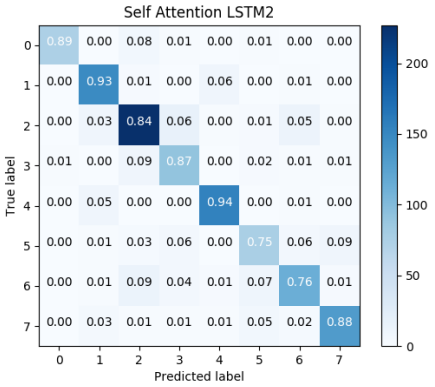
4.2 Qualitative Analysis

The T-SNE [26] embedding space of the representation in the last hidden layer assigned by each sequence model are shown in Figure 4.4. The visualizations give important deductions regarding the origins of each language considered. Indian languages are broadly categorized as the North Indian languages and South Indian languages. Each such category is considered to have similar dialects and properties.

In each T-SNE embedding space plot, we observe that Bengali and Hindi clusters appear close to each other, as Bengali and Hindi are indeed similar languages. Similar is the case with Marathi and Gujarati clusters, geographically being neighbouring states (regions) in India. The clusters of the South Indian languages of Tamil, Telugu and Kannada, geographically being neighbouring states, must appear near each other which prevails in most of the embedding space plots. The Punjabi cluster appears near the clusters of the South Indian languages of Tamil, Telugu and Kannada, an error which prevails in all the embedding space plots. This error may be due to similar spectral, tonal or acoustic characteristics. There is clear distinction between the South Indian languages (believed to have Dravidian roots) and the North Indian languages (believed to have Indo-Aryan roots) in each embedding space, an important experimental finding.

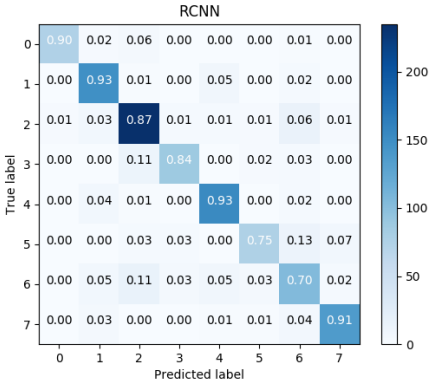


(a) Bi-LSTM with Self Attention (1 layer)

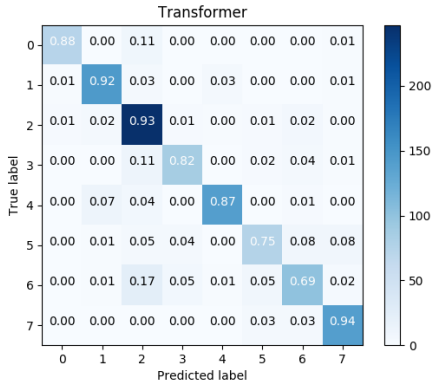


(b) Bi-LSTM with Self Attention (2 layers)

(c) Bi-LSTM with Soft Aligned Attention

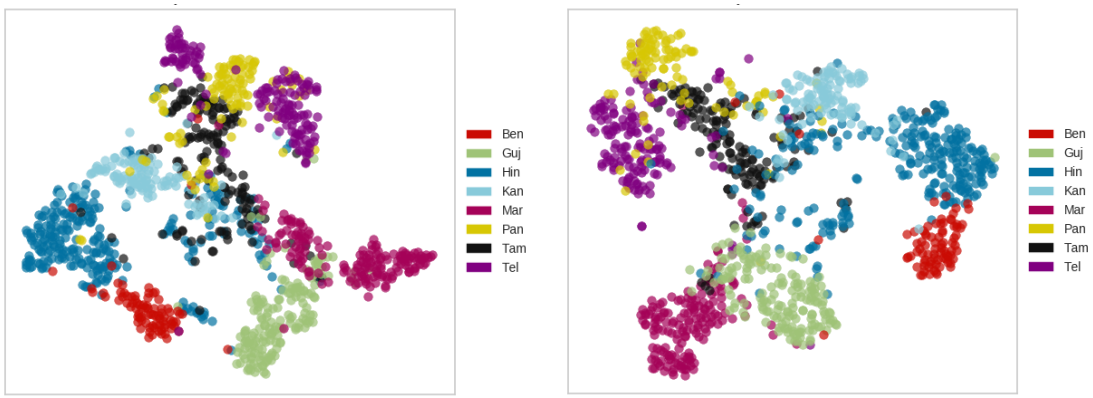


(d) RCNN



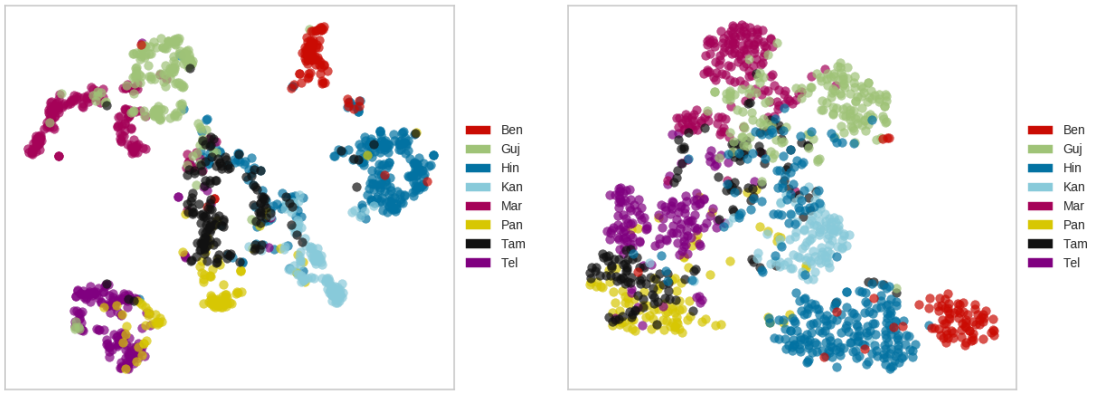
(e) Transformer

Figure 4.3: Confusion matrices for each sequence model on test data. The corresponding labels are 0 for Bengali, 1 for Gujarati, 2 for Hindi, 3 for Kannada, 4 for Marathi, 5 for Punjabi, 6 for Tamil and 7 for Telugu.



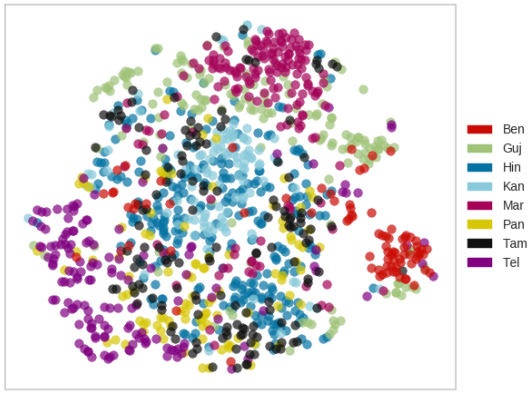
(a) Bi-LSTM with Self Attention (1 layer)

(b) Bi-LSTM with Self Attention (2 layers)



(c) Bi-LSTM with Soft Aligned Attention

(d) RCNN



(e) Transformer

Figure 4.4: T-SNE embedding space of representations in the last hidden layer assigned by each sequence model on test data.

CHAPTER 5

Conclusion and Future Work

In this thesis, we have introduced a new framework for Indian language identification using VAE representation learning and state-of-the-art sequence models to capture the temporal characteristics of speech. The framework performs well on identification of 8 well known languages. The framework also helps improve language identification in the future as sequence models in natural language processing become better capturing long range dependencies and other temporal aspects of sequences. It can be applied in several other speech processing scenarios as well where the task requires representation learning from speech utterances and subsequent classification using a sequence model. We see VAEs are powerful probabilistic models which can learn useful representation from speech utterances and these representations can be utilized in several downstream tasks.

Latent vectors learnt by a VAE from speech are extremely useful representations for several other tasks for speech processing and can aid multi-modal data classification. Several broad tasks, such as **emotion recognition**, **person identification**, **music analysis**, etc., require learning joint representations for several modalities.

Future direction requires the exploration of learning latent representations for audio and speech, and fusion of representations for several modalities such as text and images for performing downstream tasks by intelligent systems.

REFERENCES

[1] Yun Lei, Luciana Ferrer, Aaron Lawson, Mitchell McLaren, and Nicolas Scheffer. “Application of Convolutional Neural Networks to Language Identification in Noisy Conditions”. In: *Odyssey*. 2014.

[2] V. MounikaK., Sivanand Achanta, R. LakshmiH., Suryakanth V. Gangashetty, and Anil Kumar Vuppala. “An Investigation of Deep Neural Network Architectures for Language Recognition in Indian Languages”. In: *INTERSPEECH*. 2016.

[3] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *CoRR* abs/1312.6114 (2013).

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[5] Rong Tong, Bin Ma, Donglai Zhu, Haizhou Li, and Eng Chng. “Integrating Acoustic, Prosodic and Phonotactic Features for Spoken Language Identification”. In: vol. 1. June 2006, pp. I –I. DOI: 10.1109/ICASSP.2006.1659993.

[6] Liang Wang, E. Ambikairajah, and E. H. C. Choi. “Multi-lingual Phoneme Recognition and Language Identification Using Phonotactic Information”. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 4. 2006, pp. 245–248.

[7] Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. “Language Identification Using Deep Convolutional Recurrent Neural Networks”. In: *ArXiv* abs/1708.04811 (2017).

[8] Javier Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and Pedro Moreno. “Automatic language identification using Long Short-Term Memory recurrent neural networks”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (Jan. 2014), pp. 2155–2159.

[9] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[10] Sarthak, Shikhar Shukla, and Govind Mittal. “Spoken Language Identification using ConvNets”. In: *AmI*. 2019.

[11] Grégoire Montavon. “Deep learning for spoken language identification”. In: (Jan. 2009).

[12] Metso Leena, K. Srinivasa Rao, and Bayya Yegnanarayana. “Neural network classifiers for language identification using phonotactic and prosodic features”. In: *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing, 2005.* (2005), pp. 404–408.

[13] Ramakrishna Thirumuru, Ravikumar Vuddagiri, Krishna Gurugubelli, and Anil Kumar Vuppala. “Significance of Accuracy in Vowel Region Detection for Robust Language Identification”. In: *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)* (2018), pp. 826–830.

[14] Aarti Bakshi and Sunil Kumar Kopparapu. “Spoken Indian language classification using artificial neural network — An experimental study”. In: Feb. 2017, pp. 424–430. DOI: 10.1109/SPIN.2017.8049987.

[15] Tirusha Mandava and Anil Kumar Vuppala. “Attention based Residual-Time Delay Neural Network for Indian Language Identification”. In: *2019 Twelfth International Conference on Contemporary Computing (IC3)* (2019), pp. 1–5.

[16] Wei-Ning Hsu, Yu Zhang, and James Glass. “Learning Latent Representations for Speech Generation and Transformation”. In: *Interspeech*. 2017, pp. 1273–1277.

[17] Alex Sherstinsky. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network”. In: *ArXiv* abs/1808.03314 (2018).

[18] M. Schuster and K.K. Paliwal. “Bidirectional Recurrent Neural Networks”. In: *Trans. Sig. Proc.* 45.11 (Nov. 1997), pp. 2673–2681. ISSN: 1053-587X. DOI: 10.1109/78.650093. URL: http://dx.doi.org/10.1109/78.650093.

[19] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 551–561. DOI: 10.18653/v1/D16-1053. URL: https://www.aclweb.org/anthology/D16-1053.

[20] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473 (2014).

[21] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. “Recurrent Convolutional Neural Networks for Text Classification”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 2267–2273. ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=2886521.2886636.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[23] “CMU INDIC Speech Synthesis Databases”. In: (). http://festvox.org/cmu_indic/.

[24] Kishore Prahallad, E Kumar, Venkatesh Keri, Rajendran Suyambu, and Alan Black. “The IIIT-H Indic Speech Databases”. In: (Nov. 2012).

[25] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014).

[26] Laurens van der Maaten and Geoffrey Hinton. *Visualizing data using t-SNE*. 2008.