Exploring Learned Representations for Speech-based Applications

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

Bachelor of Technology

by

Aniket Agrawal (160108005)
Varshil Gandhi (160108016)

Under the guidance of
Prof. Rohit Sinha

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
June 2020

CERTIFICATE

This is to certify that the work contained in this thesis entitled "Exploring Learned Representations for Speech-based Applications" is a bonafide work of Aniket Agrawal (Roll No. 160108005) and Varshil Gandhi (Roll No. 160108016), carried out in the Department of Electronics & Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

Prof. Rohit Sinha
Project Supervisor

DECLARATION

The work contained in this thesis is our own work under the supervision of the guides. We have read and understood the "B. Tech./B. Des. Ordinances and Regulations" of IIT Guwahati and the "FAQ Document on Academic Malpractice and Plagiarism" of EEE Department of IIT Guwahati. To the Best of our knowledge, this thesis is an honest representation of our work.

Aniket Agrawal	Varshil Gandhi
Author 1	Author 2

Date: 17^{th} June, 2020

Acknowledgments

We want to thank our project supervisor, Prof. Rohit Sinha and our PhD mentor, Ganji Sreeram, for their assistance and dedicated involvement in every step throughout the process. Without their support and understanding over the last year, especially during the difficult times of COVID-19, we wouldn't have accomplished our goal.

Abstract

This report highlights the improvements in speech recognition and identification tasks by employing wav2vec, a learnable feature extractor based on an unsupervised learning method. This extractor can utilize easily available and abundant unlabelled data to give better representations of audio data. In this report, we have discussed the merits of using trainable feature extraction methods over traditional log filterbanks. The initial aspect was to improve the traditional automatic speech recognition (ASR) systems by replacing the log-mel filterbanks with the wav2vec [28] model. Moreover, we have tested and mentioned different loss functions commonly used for speech recognition, including connectionist temporal classification (CTC) loss [9]. We have compared the proposed model with the model used for earlier training and highlighted the character error rate (CER) improvements by introducing wav2vec in the existing ASR model.

As our final objective, we explored the language identification (LID) task in code-switched audio samples. We present a neural network for the code-switched spoken language identification task. Finally, we use an established listen, attend and spell (LAS) acoustic model [3] for the utterance level LID, which uses an attention network for decoding. Motivated by the achievements in the ASR task, we replaced the prior log-mel filterbanks with the wav2vec, a data-driven learnable feature extractor. The wav2vec model helps to deal with the unavailability of labelled speech data of multiple languages.

Unlike the traditional CTC-based models, which assume independence between adjacent output labels, both the wav2vec model and the LAS model assume a relation between serial speech data as well as the output labels. Further attention-based decoder captures the past as well as the next speech data for the LID task.

Contents

Chapter 1

Introduction

In various fields, like image classification, optical character recognition, or speech recognition, deep artificial neural networks have proved to be a boon, increasing accuracy levels similar to human performance. In speech recognition and identification tasks, despite massive datasets available for the English speech, researchers face difficulty extending these deep models to deal with multilingual tasks. The challenge regarding such extension is the limited data availability for some traditionally old and less documented languages like Telugu or Gujarati. Generally, transfer learning techniques are deployed to deal with such a shortage of data. But in such cases, transfer learning may cause significant scaling problems. Further, some languages differ so much that the source network's training data and the fine-tuning data may not remain identical enough. The transfer learning method will only worsen the performance of the model in such cases. Therefore, we explored unsupervised learning methods to learn better representations of speech data using unlabeled datasets. The unsupervised model, thus eliminates the need for labeled speech data to learn better speech representations. Even though the speech community may have less labeled data for Telugu language, unlabeled speech is easily available in abundance.

Unsupervised learning has shown great results in domains like natural language processing [6] [26] [2]. Moreover, unsupervised learning has shown improvements in tasks such as text classification and machine translation [7] [16]. In computer vision domains, vast improvements have been made using learned representation for ImageNet [5] on various tasks like image captioning [32] or pose estimation [24]. Furthermore, tasks with extremely small datasets such as speaker identification has shown accuracy improvements using pre-trained speech representations [27].

1.1 Review of automatic speech recognition

Firstly, we tackled the automatic speech recognition (ASR) problem with monolingual English audio data to validate our hypothesis. According to our hypothesis, these learned data representations by an unsupervised model would significantly outperform traditionally used log-mel filterbanks in any speech related tasks. Log-mel filterbanks are a rule-driven method and has no learnable parameters for optimization. On the other hand, unsupervised models like wav2vec [28] can learn to extract features from the speech by optimizing for other related tasks. We pursued this domain as ASR models can potentially pose as the solution for significant industrial and social problems. Ranging from its application in evolving search engines to healthcare industries and securities, speech recognition has eased and accelerated the conversion process. Currently, due to limitations in ASR, technologies are not able to make advanced improvements in human-computer interaction (HCI) domains. There have been many attempts to design a state-of-the-art ASR models for various languages.

Initially, ASR models were built for different speech utterances starting with single words to connected words and finally moving to continuous speech. Since the continuous speech was practiced, experiments were further performed on spontaneous speeches where the speech was not rehearsed and flowed with a peculiar accent or informal expressions. The advent of continuous speech recognition posed significant challenges for the researchers as the output label size was not fixed, and thus, making a single model only became possible after connectionist temporal classification (CTC) loss [9] emerged. Various approaches to ASR include the acoustic-phonetic approach [8] and the pattern recognition approach. The former method that had been studied for 40 years used methods like gaussian mixture models and support vector machines [31] and was based on the hypothesis that only finite distinctive phonemes exist in any spoken language. The latter approach centrally focused on forming speech pattern representations and matching the patterns of unknown speeches with known patterns. It incorporated techniques similar to hidden markov models for pattern matching. Finally, the deep learning models gained much popularity; [12] that combined both the methods and exploited the concepts of acoustic phonetics for pattern recognition.

We presented a deep neural network that learned to transcribe speech utterances into characters for the task of ASR. Unlike some prior models, which incorporated multiple models for acoustics, pronunciation, and language modeling, we developed an end-to-end pipeline, which

further captured the relation between characters predicted by the model, which was lacking in previous CTC [9] based models. We introduced a novel feature extraction model to replace the trivial log-mel filterbanks to achieve better performance in speech-to-text recognition tasks. Rather than log-mel filterbanks, we integrated the wav2vec model to calculate learned speech representations, with the original ASR model. We further decided to use a single wav2vec model to deal with code-switched data. Code-switched refers to the phenomenon of using two or more languages while speaking a sentence. The syntax of a code-switching sentence belongs to the *native language*, while the other language is referred to as *foreign language*. Some of the common examples include French-German, Hindi-English, Malay-English, Mandarin-English, and Spanish-English. Wav2vec model can learn relations between these sets of languages when the model is trained using such code-switched languages.

It is not advisory to use multiple models for identifying various languages in a spoken speech utterance because different languages may contain similar pronunciations of two completely different words with different meanings. For instance, 'door' word means 'thread' in Hindi, and the English word 'case' means 'hair' in Marathi. These examples force the model to understand the context in which the words were spoken. When spoken in a sentence, these words may mislead the classifier and give catastrophic results. Say, for instance, "mere liye door khol dena" is a sentence spoken in Hindi, but the word 'door' implies its English meaning rather than its Hindi interpretation, that is a 'thread'.

1.2 Review of language identification task

After our observed success in the monolingual ASR tasks, we planned to extend our hypothesis towards multilingual datasets, which was the ultimate foundation of our proposed theory. Some of the works found for multilingual tasks such as multilingual speech recognition were useful for further research. We traced the development of multilingual ASR through a few of the prevalent publications. The paper in [34] describes their work for English and Swedish language. They combined two monolingual language models by sharing parameters across both the languages, maintaining a better performance. The work presented in [19] focuses on the use of discriminative training procedures, which significantly improved their metrics for recognition of four different African languages. In addition to that, [11] suggested the use of joint optimization on shared data along with discriminative features. This approach improved their model at the cost of increased training time.

Conclusively, we aim to develop a model for language identification (LID) task. The initial stages of the project used wav2vec for extracting learned feature representations for targeting improvements in monolingual ASR tasks. The results clearly expressed the advantages of employing learned compact feature representations using the unsupervised model for speech recognition tasks. The LID task refers to recognizing the language in which a speech is spoken. In utterance level LID, we observe a relation or dependence of an output label on adjacent labels. For example, the adjacent characters in a word belong to the same language, and the probability of certain nearby words associated with the same language is higher. Hence, we take advantage of the listen, attend and spell (LAS) model [3] that uses the pyramidal recurrent-net and an attention-based decoder to identify the language. In addition, the wav2vec [28] approach has been used to derive more useful feature representations given the raw speech samples. Analogous to various automatic speech recognition tasks and text recognition tasks [6], feeding a learnable feature vector has proved to improve performance rather than feeding raw speech signals.

The LAS model stacked over the wav2vec model further tackles the dimensionality problem. The wav2vec model reduces the input length by a factor of 160, and the LAS encoder further reduces the input length by a factor of 2 with each recurrent layer. Without this length reduction, the model may converge too slowly, and with this unavailability of large datasets for multiple languages, the error rates may not reduce significantly. Further, training of complex

models requires an abundance of data, but training instances for tasks related to multiple languages are generally limited. Wav2vec provides a solution to the above-posed problem using unsupervised learning technique to extract high-level and compact features for further training.

LID comes naturally in human readers that are familiar with the language. Researchers over the 50 years of research have tried to mimic this human ability. However, unlike human capabilities, a LID task aims to learn thousands of languages [29]. Research on the LID task has been traditionally focussed on monolingual written documents [14]. However, there has been some work for speech systems. For example, [13] has worked on LID that focusses on the identification of spoken utterances using synthetic data. Also, [4] uses a Bayesian model to determine the language of words. It uses the relative frequencies of the character trigrams as probabilities for the model. Further, we focus on integrating some techniques used in speech recognition systems for improvement of LID task. We plan on learning useful representations of the speech signals that can be further used for faster and better convergence of LID models.

Prior to extracting learned representations of data, models were built upon a raft of feature extraction techniques including but not limited to mel-frequency cepstral coefficients (MFCC), linear predictive coding (LPC), and fast fourier transform (FFT). Learning pre-trained representations is widely used in various applications. As only a few models have been built to learn acoustic representations for complete sentences, including Bert [6] and wav2vec [28], we can draw references from acoustic to word recognition. This domain significantly aligns with the LID task as the word recognition shares a similar set of features. In the former area, learning word embedding has dramatically reduced the error rates, as evident in a few of the significant word embedding techniques like word2vec [18], Global Vectors GloVe [25]. We aim to see a similar improvement in the LID task.

We proceed in this paper by describing the model architecture in detail in chapter 2. This chapter explains both the proposed models and baseline models in three sections: preprocessor, acoustic model, and decoder. Chapter 3 discusses the experimental setup. It majorly describes the dataset details, the frameworks used, and the training details of the proposed model. Finally, chapter 4 mentions the metrics and results obtained, and chapter 5 concludes the entire report and mentions some possible future works.

Chapter 2

Model Architecture

The current design of numerous speech recognition models, involving many state-of-the-art models revolves around a similar template. Though the training procedure varies for every single model, the template does describe the model's inherent structure. The template consists of three significant modules:

1. Preprocessor
2. Acoustic Model
3. Decoder

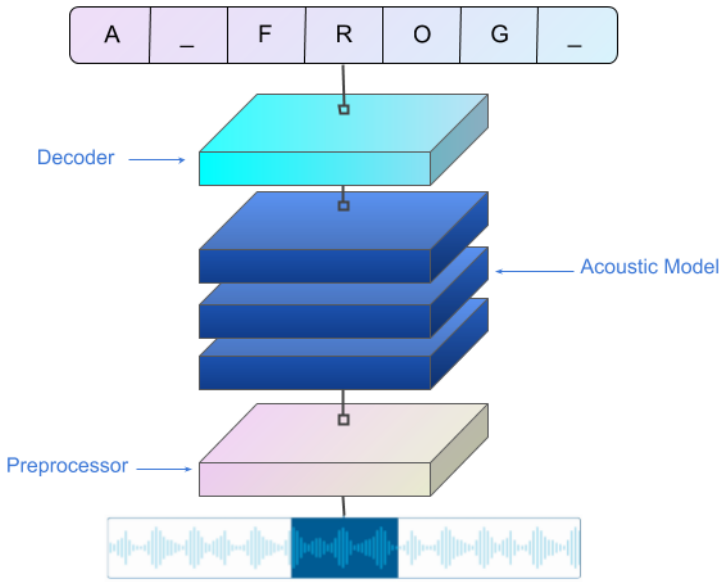

Figure 2.1: Template for speech-to-text models

The **preprocessor** module transforms the speech signal into compact and compressed representations that represent some essential qualities of the speech data. There are two significant

reasons why models use a preprocessor:

1. The speech signal is usually a very long sequence of decibel values. For example, a 10-second speech signal recorded using a device with 16 kHz recording frequency, will result in a sequence of 16,000∗10, that is 1,60,000 dimension vector. Such speech data, if represented as a vector and is provided to any learning model, will suffer from the curse of dimensionality and huge weights matrices.

2. Moreover, the speech data has temporal relations, and capturing these interdependence between data is more useful than simply using some decibel values in speech recognition tasks. Transformations of speech data and filtering is another reason for the preprocessor to exist.

In the following sections, we will mention the two preprocessor modules used in our experiments: the **log-mel filterbanks**, and the **wav2vec**.

The task of the **acoustic model** is to understand the preprocessed speech data and predict the probability distributions on phonemes or characters, depending on the prediction tasks. Depending upon the training procedure and loss functions, the outputs of the acoustic models vary. For example: In CTC-based models, the output of some speech sounding 'CAT' is 'C-C-C-C-A-A-A-T-T-T', while in seq-to-seq models with attention, the output is usually the probability distribution of every phoneme at a particular timestep. In our experiments, we implemented two acoustic model designs: **Listen, attend and spell** [3] acoustic model, and **DeepSpeech** [1] acoustic model.

Whatever the acoustic model design may be, each model require a **decoder** module to decode their output into the necessary format. In our experiments, we implemented a '**beam search decoder**' for the decoding task.

The following sections explain major components of each module used during experimentation and conclusively mentions all the models that we used for our LID or ASR tasks.

2.1 Preprocessor

2.1.1 Log-mel filterbanks

To train any speech recognition model, the initial step is to identify the relevant features in a sound signal that are crucial for the recognition and identification tasks. This involves discarding the background noise or emotions. The human ears are evolved to interpret audio frequencies on a non-linear scale. Therefore if the sound is presented on a linear frequency scale, it may compromise the recognition abilities of the system. Recent research [30] has provided empirical evidence suggesting that designing systems that work in a non-linear fashion improves speech processing capabilities. One of the popular approaches involves logarithmic filterbanks to obtain the desired non-linear frequency resolution.

Computational blocks

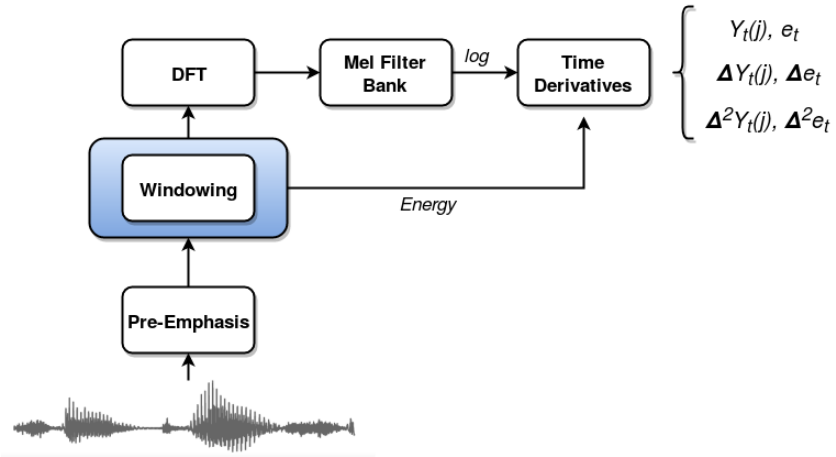

Figure 2.2: Block diagram for the computation of log-mel filterbank features including the short-time energy and their first and second-order time derivatives.

The audio signal is continuously changing, making it challenging to analyze the signals or apply the filterbanks. Therefore, we assume that the signal is stationary for a short time frame, and a window is applied to analyze that signal further. In our experiment, we took 25 ms overlapping windows with a stride of 10 ms. Generally, a rectangular or a hamming window is used for the purpose. For a window represented by $w[n]$, the windowed speech frame is derived

as $y[n] = s[n] * w[n]$. Example of a hamming window:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{N-1}, & \text{if } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

We calculate the short-time energy of the window as one of the features for model training. Further, we extract the spectral information from the window. Here we apply a 512 points discrete-time fourier transform (DFT) algorithm on the 25 ms signal window to recognize the different frequencies.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{2.2}$$

DFT gives the energy of each frequency. However, human ears are more sensitive towards low-frequency components as compared to higher frequencies. So the DFT is warped in mel scale, which makes the sound equidistant in pitch.

$$\text{mel}(f) = 1127\ln(1 + \frac{f}{700}) \tag{2.3}$$

This eventually results in 10 filters spaced linearly below 1 kHz and remaining filters spread logarithmically above 1 kHz. Subsequently, we need to take the logarithm of the energies produced by mel warping. Extending the analogy to human ears, we do not perceive loudness on a linear scale. To double the perceived level of loudness, we need to spend eight times more energy. This indicates that significant variations in signals at high frequency may not be differentiable to human ears as compared to low-frequency signals.

For the final output, we received a 123-dimensional vector for each frame. The 512 point DFT computed for each frame is first converted to a mel scale. Further, we divide the frequencies into 40 filterbanks and report the energies of each filterbank, resulting in a 41-dimensional vector, including the dc energy returned for each of the sliding windows. Then we calculate the first-order and second-order time derivatives of these 41 features, each returning another 41-dimensional vector. Thus the resulting vector size of each frame is (123,1). The final output size used for training is (batch size, temporal length, 123). The derivatives of these features are given as:

$$\triangle_t = \frac{\sum_{n=1}^{N} n * (c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.4}$$

where $\triangle_t$ represents the time-derivative features and c_t represents the filterbank energies.

2.1.2 Wav2vec model

The wav2vec model is aimed at learning compact and useful features from raw speech data that can be used either solely as an encoder or as a feed-forward network for other models. Based on the fact that most languages share a standard set of tones, wav2vec can learn shared representations for different languages. As shown in [28], using wav2vec model features helps in surpassing performance in certain speech recognition tasks compared to the log-mel filterbanks.

Architecture

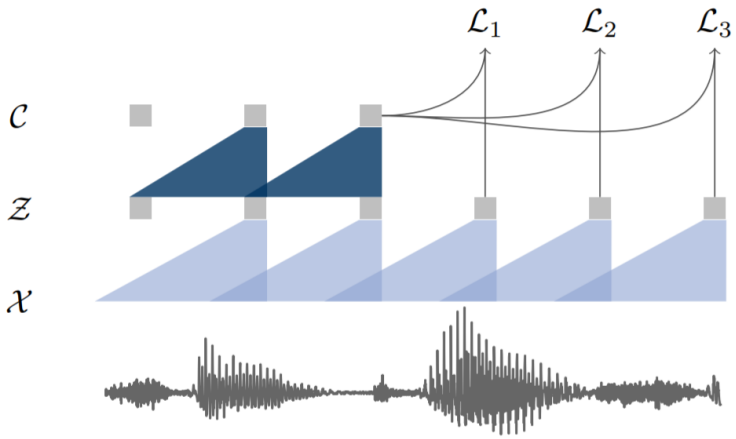

Figure 2.3: Representation of pre-training from audio data X, which is encoded using the encoder and context networks that are piled on top of each other. The arrows represents the model's optimization on next-step prediction task. [28]

The wav2vec model has two components. First is the encoder network $f : X \rightarrow Z$, and the second one is the context network $g : Z \rightarrow C$.

The encoder will take raw audio samples as the input and feed it into a five layer convolutional layer with respective kernel sizes as (10, 8, 4, 4, 4) and strides as (5, 4, 2, 2, 2). The encoder network takes in 30 ms of 16 kHz data, which is a 480-dimension vector in each window with a stride of 10 ms. The stride of the encoder results in a low frequency (100 Hz) data.

The receptive field of 30 ms is further amplified in the context network. The context network will take $(z_i, .., z_{i-v})$ encoder outputs extending its receptive size to $30 * v$ ms and produce a single contextualized tensor $c_i = g(z_i...z_{i-v})$. In the currently implemented model, the total receptive field of the context network is 210 ms. The context network further extends

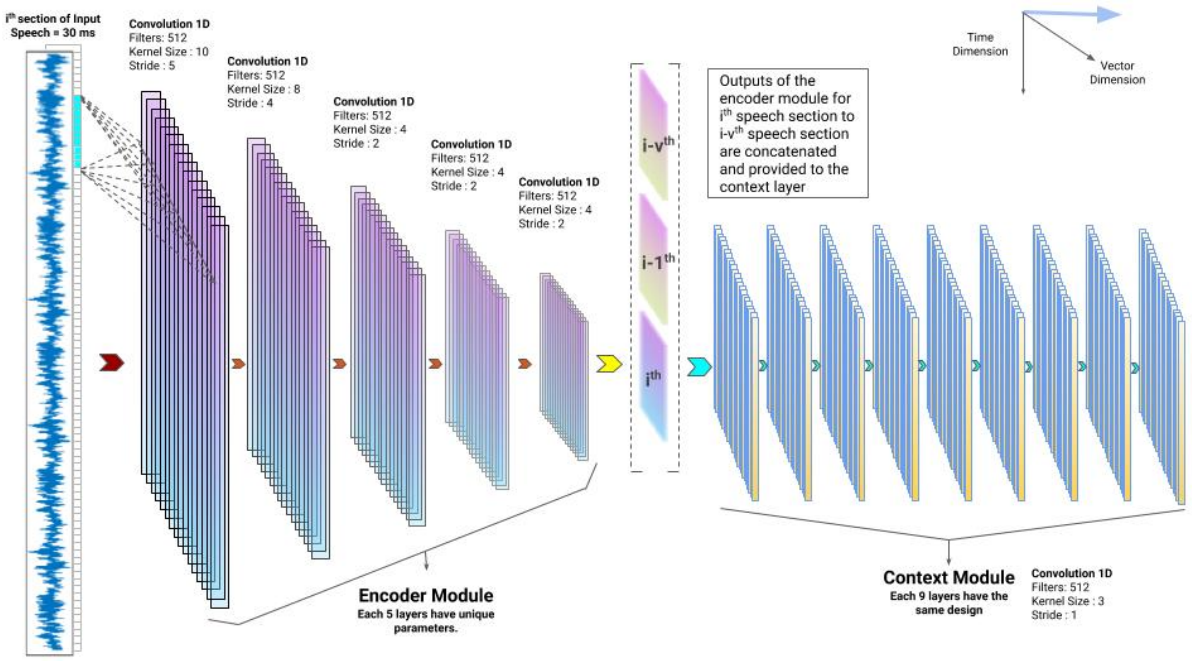

Figure 2.4: Illustrating the overall architecture if the wav2vec model using a small part of the entire speech signal. The encoder module processes 30 ms speech audio into compact representations with reduced dimension. The context module takes in the encoder output for the i^{th} speech section till $(i-v)^{th}$ speech section.

over encoder with nine convolutional layers, each with kernel size three and stride one keeping the temporal dimension of the output constant.

The layers of both the encoder and context networks consist of a causal convolution with 512 channels, a group normalization layers, and a rectified linear unit (ReLU) nonlinearity. For better results, we have applied normalization across both the feature and temporal dimensions.

Optimization

Wav2vec is designed to engage in a self-supervised training procedure. Self-supervision is accomplished by training the model to predict the next timestep it is supposed to encounter given previous timesteps have already occurred. To achieve this objective, the model produces some distractor samples from a uniform distribution and tries to distinguish the original sample k steps ahead in the future from these distractor samples. Wav2vec uses a contrastive loss function [20] for its optimization given by:

$$\text{Loss} = -\sum_{k=1}^{K}\sum_{i=1}^{T-k}\left(\log\sigma(z_{i+k}^{T}h_k(c_i)) + \lambda_{\hat{z}\in p_n}\mathrm{E}[\log\sigma(-\hat{z}^{T}h_k(c_i))]\right) \tag{2.5}$$

where σ represents sigmoid function, z_{i+k} is the true sample to be predicted with a probability of $\sigma(z_{i+k}^{T} h_k(c_i))$, and h_k represents a affine transformation, $h_k(c_i) = W_k c_i + b_k$ for each step k.

The level of training difficulty is gradually increased to obtain better results. The model was tasked to predict sudden changes immediately after an unaltered portion of the clip. Further, the model was trained to anticipate frames comparatively farther in the future.

The wav2vec is aimed to understand the meaning of the speech waveforms by predicting future samples and create useful compact representations from raw audio. Thus, these representations can significantly improve the performance metrics of various speech processing tasks.

2.2 Acoustic Models

2.2.1 Listen, attend and spell model

The features extracted by the preprocessor module are used as an input for the listen , attend and spell (LAS) acoustic model. There are mainly two major components in the LAS model [3], *listener* and *speller*. The function of the first part, the *listener*, is to form high-level features from the processed input speech. We extract the processed input speech from the raw signal using some preprocessor module. Then, it is passed into the LAS encoder to create compact features that capture sequence. The second part is known as the *speller* and has an attention network stacked over the recurrent neural network (RNN) decoder and the listener outputs act as the input for the RNN layers. This network finally outputs the character probability for each frame. Using LAS we need to model:

$$\mathrm{P}(y|x) = \prod_i \mathrm{P}(y_i|x, y_{<i}) \tag{2.6}$$

where x represents input to the LAS model, and y_i represents the output of the i^{th} attention. As the name suggests, listen represents the encoder; and attend and spell represents the attention based decoder.

$$h = \mathrm{Listen}(x) \tag{2.7}$$

$$\mathrm{P}(y|x) = \mathrm{AttendAndSpell}(h, y) \tag{2.8}$$

Here h represents the output of the listen network.

Architecture

To capture the sequence in the input, the *listener* uses a bipyramidal long short term memory (BLSTM) [10]. Moreover, to reduce the computation complexity for attention mechanism and to produce compact high-level features, encoder engages a pyramidal BLSTM (pBLSTM) that reduces the temporal dimension by a factor of two in each successive layer. Thus in the current setting, the i_{th} node in the j_{th} layer is given by

$$h_i^j = \mathrm{pBLSTM}(h_{i-1}^j, [h_{2i}^j, h_i^{j-1}]) \tag{2.9}$$

The output distribution is a function of the current decoder state and context. The current decoder state is a function of the decoder state, output, and context of the previous timestamp.

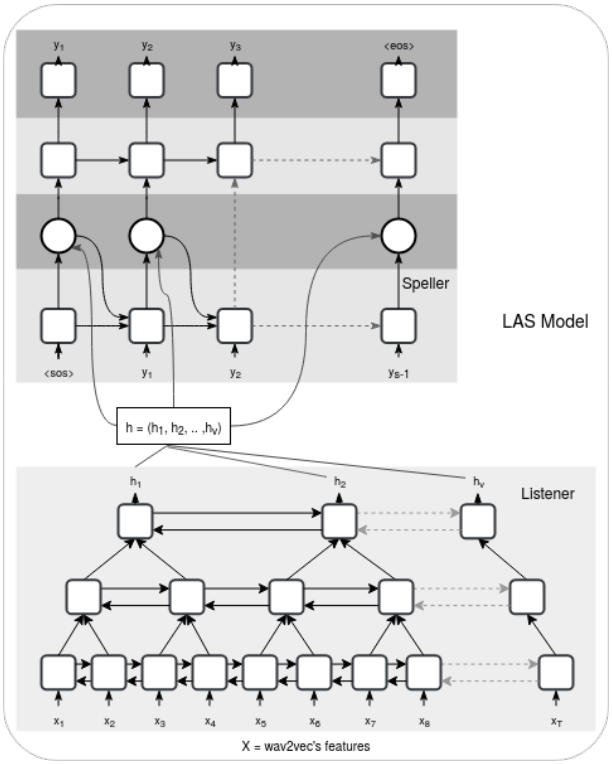

Figure 2.5: LAS acoustic model architecture: The listener is a pyramidal BLSTM encoding our input sequence x into high-level features h, the speller is an attention-based decoder generating the y characters from h. [3]

An attention mechanism produces the context vector. Specifically,

$$c_i = \text{AttentionContext}(s_i, h) \tag{2.10}$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \tag{2.11}$$

$$\text{P}(y_i|x, y< i) = \text{CharacterDistribution}(s_i, c_i) \tag{2.12}$$

where c_i represents context states and s_i represents decoder states.

Optimizer and Decoder

The LAS model can be efficiently trained end to end without the requirement of any pre-training of separate parts. The decoder predicts the output at t^{th} timestep based on the previously occurred timesteps. During training, we can feed the ground truth for previous characters and maximize the log probability for model optimization.

$$\max_{\theta} \sum_{i} \log \text{P}(y_i|x, y^*_{<i}; \theta) \tag{2.13}$$

where $y^*_{<i}$ represents the ground-truth.

However, while decoding the ground-truths would not be available. To combat this situation, we randomly feed predicted characters and ground-truth with some sampling rate. Decoding is achieved using a novel beam search algorithm mentioned in Section 2.3.1, where for each timestep, β most likely beams are chosen to store β hypothesis of possible outputs.

2.2.2 DeepSpeech model

The *Interspeech language identification challange 2020* proposed this baseline. For this baseline model, preprocessor and decoder module used are mentioned in this section itself to ease the flow of understanding. The speech input is first passed through a short-time fourier transform (STFT) feature extractor, which returns a matrix. The matrix is further split in a magnitude and phase component using the 'magphase' function of the *Librosa* library. The output matrix is converted as:

$$\text{output matrix} = \log(1 + \text{output matrix}) \tag{2.14}$$

and fed to the model architecture [1]. Then to optimize the model, we use a CTC [9] loss function.

STFT feature extractor (Preprocessor)

A standard fourier transform gives the frequency information averaged over the entire time signal. But the better speech features would require frequency information localized over time frame intervals which is given by STFT.

$$X_{STFT}[m, n] = \sum_{k=0}^{L-1} x[k]g[k - m]e^{-j2n\pi k/L} \tag{2.15}$$

Acoustic model architecture

We used a multilayer neural model with 2 convolutional (Conv2D) layers, 5 long short term memory (LSTM) layers, and a fully connected layer. We feed the padded feature vector to the first Conv2D layer with 32 filters, kernel size $(41, 11)$, and stride $(2, 2)$. Then it is followed by a batch norm and a hard tanh activation layer. Then the output is again padded and provided to a second Conv2D layer with 32 filters, kernel size $(21, 11)$, and stride $(2, 1)$. It is again followed by a batch norm and a hard tanh activation layer. The output is then fed to 5 layers of bi-directional LSTM network, where each layer has 1024 hidden neurons for each time step.

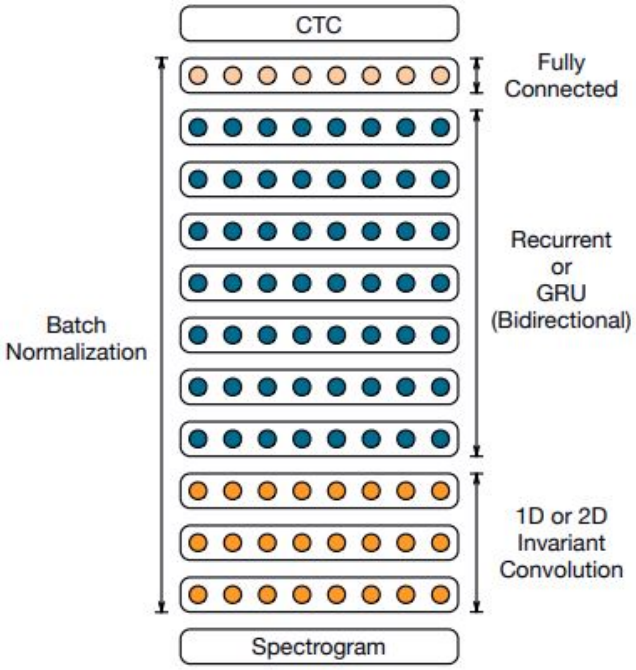

Figure 2.6: DeepSpeech acoustic model design, where first three layers are the Conv2D layers, followed by seven bidirectional LSTM layers and a last fully connected layer. Finally, a CTC based decoder is used. [1]

The final prediction is then formed for each time step by passing through a fully connected layer of size (1024, number of output characters) with a softmax activation layer. The probability of the k^{th} character at t^{th} timestep for a input-output pair (x,y) is given by:

$$p(l_t = k/x) = \frac{\exp(w_k^L \cdot h_t^{L-1})}{\sum_j \exp(w_j^L \cdot h_t^{L-1})} \tag{2.16}$$

where t denotes the time step, L denotes the layer number, w are the weights of fully connected layer, h is the output of LSTM layer and l denotes labels.

Loss Function

Since the input size is variable, the output size is also variable. So we use a CTC loss function [9] for optimization.

Encoding of output labels: We need to encode our output labels to be of variable lengths. Eg: the word 'our' can be actually predicted as 'oooouurrrrrr'. So we need to encode 'our' to various alignments. Certain rules for encoding are:

1. Use a special character '-' to distinguish between the same characters that occur consecutively.

2. '-' can be repeated as many times, which will be taken care of while decoding.

3. We can repeat each character as many times.

   Eg: 'is' can we written as '-iiisss'

Loss Calculation: The model outputs a matrix of size (number of characters, timesteps). At each timestep, we have probabilities of each character. Suppose we have probabilities for two timesteps, and the output label is 'a'. We can form alignments as '-a', 'a-', or 'aa'. Then we calculate the scores of each alignment by multiplying the probabilities at each timestep. For e.g. score for '-a' can be calculated as $P(l_1 = \text{'-'}|x) * P(l_2 = \text{'a'}|x)$, where x is the input. The net score is calculated as the sum of the scores of all alignments and the loss function is given as negative logarithm of the net score. This loss function is used for optimization.

Decoding: Finally decoding is done while predicting outputs. Decoding is simply performed in 2 steps.

- Remove all the duplicates from the prediction.
- Remove the special character '-' from the prediction.

  E.g. '-aaa-aa-bccc' will be decoded as 'aabc'.

2.3 Decoder

2.3.1 Beam search decoder

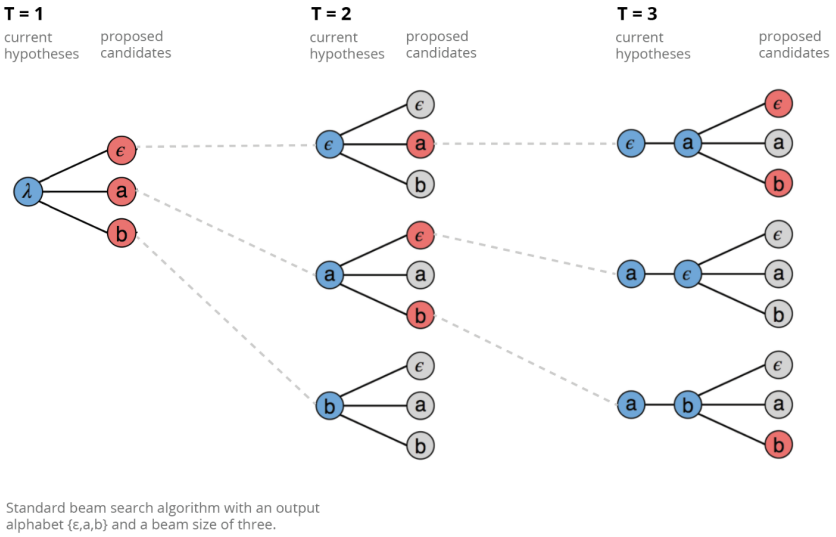

Figure 2.7: Standard beam search. Image from https://distill.pub/2017/ctc/

We have modeled a sequence network with an attention network to output the probabilities of characters for each attention. In a greedy approach, we would go in a particular order on the output labels and select the character with the highest probability. The sequence networks at t^{th} timestep throw the following output:

$$\hat{y} = \arg\max_{y} \mathrm{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \tag{2.17}$$

But if the probability of a beginning character in the ground truth is low according to the model trained, then the actual label's prediction may be compromised. Beam search algorithm partially solves this by considering a beamwidth of β.

In a beam search algorithm, at each time step, probabilities of all the possible characters are predicted depending on previously occurred characters and the attentions learned, and out of the vocabulary, top β possibilities are stored. Searching is initiated with the start of statement <sos> token and is propagated until an end of statement <eos> token is predicted. Consider an example where beamwidth β is 3, and the size of the vocabulary is 50, including characters, punctuations, and numbers. For the first character, there will be a total of fifty possibilities, out of which the top three will be stored. Next, for each of these three potentialities, fifty vocabulary

characters will be predicted, and out of the hundred and fifty possibilities, only three will be stored. This will continue until a <eos> token is predicted, and the statement will be stored in a bank of possible outputs of the ASR model.

To select the best suitable sentence for the output, we need to multiply the probabilities of all the characters. Since probabilities are less than one, the net probability for a long sentence may vanish due to numerical underflow affecting the gradients and may result in slow learning. To aid this problem, we take logs of the probabilities and add them instead. Based on the fact that log is a monotonically increasing function, maximizing log function will have the same result as maximizing the $\mathrm{P}(y|x)$.

$$\text{text\_output} = \arg\max_{y} \prod_{t=1}^{T_y} \mathrm{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \tag{2.18}$$

$$\text{text\_output} = \arg\max_{y} \sum_{t=1}^{T_y} \log \mathrm{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \tag{2.19}$$

Since multiplying fewer probabilities results in a higher number, the model will soon achieve a bias for shorter statements rendering the objective function ineffective. Thus, we introduce a length normalization factor according to the equation:

$$s(y|x) = \frac{\log \mathrm{P}(y|x)}{|y|_c} + \lambda \log \mathrm{P}_{\mathrm{LM}}(y) \tag{2.20}$$

where λ is the language model (LM) weight, $|y|_c$ is the number of characters and P_{LM} represents LM probability. In our implementation, language model is not used, implying λ is 0.

2.4 Summary of the models explored

As evident in the speech recognition community, there is ample speech data from news, conversations, or other sources. Still, most of them are unlabelled, especially for a few native or uncommon languages. Since it is impractical to annotate the data manually, wav2vec has been a boon to the speech recognition and identification tasks, attributing to the fact that it is entirely unsupervised. Thus, our proposed model utilizes the power of wav2vec models by replacing the filterbank features. Each of our end-to-end models can be described using the template. Concerning the template, the baseline model is represented in Table 2.1.

Module name	Baseline model	Proposed model	DeepSpeech**
Preprocessor	Log-mel filterbanks	Wav2vec	STFT
Acoustic model	LAS	LAS	DeepSpeech
Decoder	Beam search	Beam search	CTC decoder

Table 2.1: Representation of various baseline model structure and the proposed model with respect to the template figure 2.1. **DeepSpeech model is a baseline model used for comparing our results with models of different underlying structures.

In the proposed model, we replaced the preprocessor module; naive log-mel filterbanks with a learnable feature extractor: wav2vec. We used these compact signal representations to train the LAS acoustic model. For the decoder module, we use the beam search decoder. Every other aspect of the model remains the same apart from the preprocessor module and the training procedure. As the proposed model uses a learned preprocessor module, wav2vec is pretrained on unlabeled data before training acoustic models.

Firstly, we started with ASR task on mono-lingual English audio data, where we observed that wav2vec preprocessor improved accuracy of the model by a substantial amount (Refer Table 4.1). After the ASR results, we implemented the same model setup for LID tasks as well. Furthermore, we also used another baseline model DeepSpeech trained on CTC loss for comparison of results.

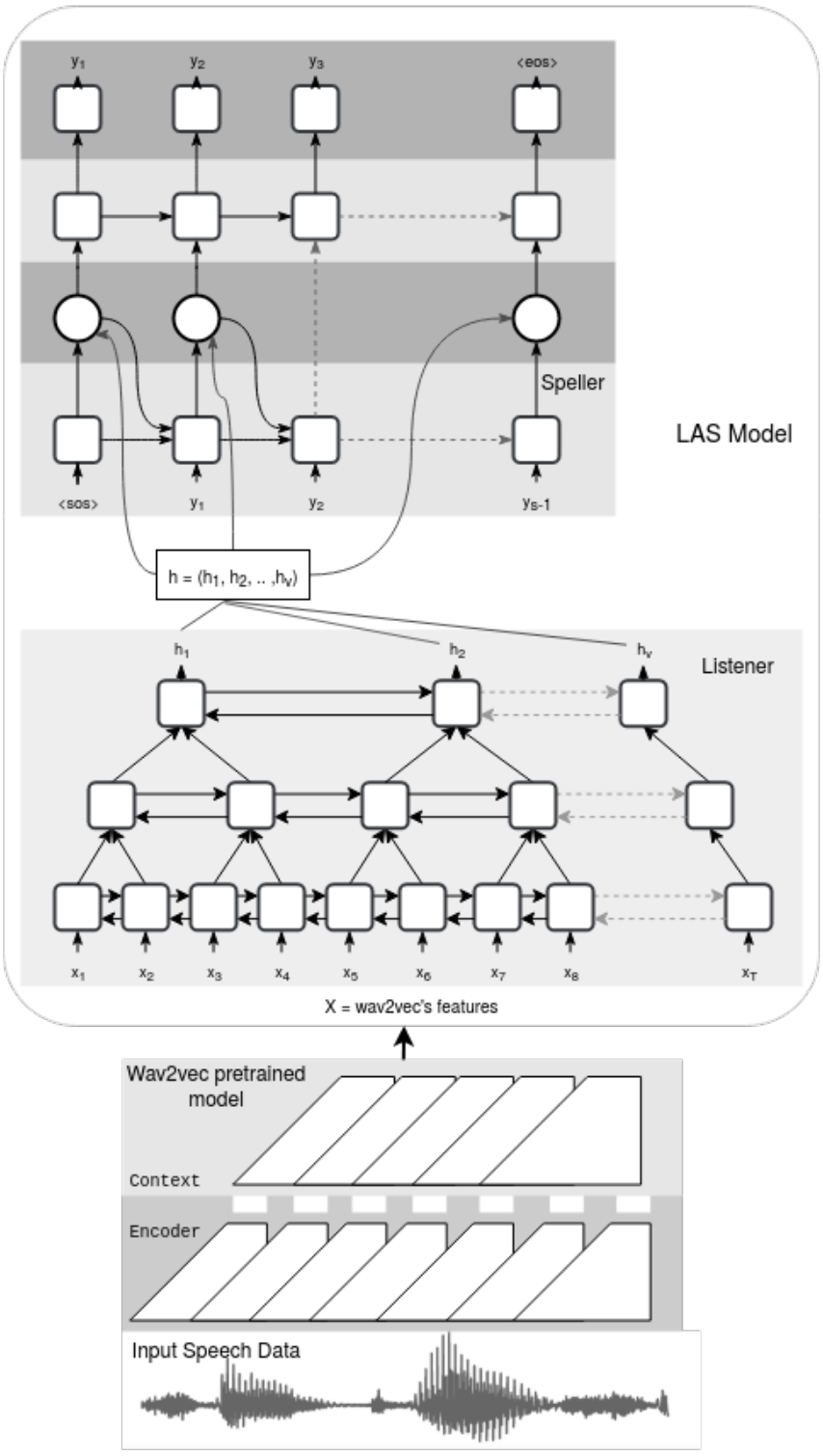

Figure 2.8: Proposed Model Architecture: The preprocessor section refers to the wav2vec model, the LAS model represents the acoustic model structure, and a beam search algorithm is used as decoder. The difference between the baseline model and proposed model design is the wav2vec block.

Chapter 3

Experimental Setup

3.1 Datasets

There are two primary tasks that we targeted upon during the course of the project. The initial goal was focused upon the ASR task for monolingual English language, while the latter was targeted upon multilingual speech identification task. Given the time restriction, we were able to generate results only on the task of utterance level LID for Telugu-English dataset.

3.1.1 Automatic speech recognition

Libri speech

Libri speech dataset [22] is one of the largest speech datasets currently used as a standard dataset in most of the speech recognition tasks. This corpus includes approximately 1,000 hours of English speech, derived from audio books read by multiple speakers. Such datasets are useful in designing language models in speech recognition tasks. In our experiments, we used 5 hours of *Libri speech* dataset with labels for training the LAS model. While the wav2vec model is pretrained on 960 hours of *Libri speech* data without any labels.

Wall street journal (WSJ) speech

WSJ dataset [23] contains English speech collected from a large variety of speakers with transcribed labels. The dataset is older and smaller than the current datasets such as Libri, but some suites of WSJ dataset are considered as standard for error rate calculations during testing. The error metrics for our models are calculated on the *'nov92 suite* of WSJ dataset.

3.1.2 Language identification

For wav2vec pre-training, we used a combination of two significant datasets released during two Interspeech conference events:

1. For classifying audio into code-switched or mono-lingual at utterance level, the 2020 Interspeech workshop on code-switching in multi-lingual communities released a standard train, dev split dataset for two major tasks,

   (a) Language identification at the utterance level.

   (b) Language identification at 200 ms Frame level.

   The dataset consists of code-switched datasets on Telugu-English, Tamil-English and Gujarati-English languages. A standard example is mentioned below:

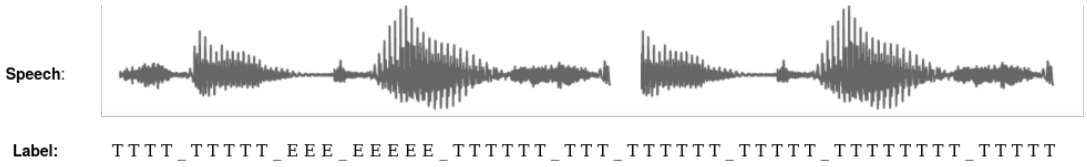

Figure 3.1: Utterance Level Data Sample (T = Telugu, E = English, _ = Word separation)

Figure 3.2: 200 ms Frame Level Sample (T = Telugu, E = English, S = Silence)

2. For accelerating the research in the domain of multi-lingual speech recognition using minimal resources, Microsoft Research released a large corpus of labeled speech data on three languages namely Tamil, Telugu and Gujarati during the *2018 Interspeech special session*. The data is for speech recognition task and contains labels consisting of phonemes.

As wav2vec pre-training requires large amounts of unlabelled speech data, we combined the train speech dataset from **1** and complete train, dev, test split of **2**, totalling to **289.51 hours** of speech data for multi-lingual pre-training. We believe that as multiple languages share some

common features, multi-lingual feature representations may provide critical advantages over the problem of training a new encoder per language.

To train and evaluate the acoustic model; we use the Telugu-English code-switched dataset for LID task **(1-a)** at utterance level, which provide labels of which language the utterance belongs. The baseline acoustic model is precisely similar in structure with our model, the only difference being the input features are computed using 40 log-mel filterbanks with a 25 ms sliding window. (Refer to Section 2.1.1 and 2.4)

3.2 Toolkits for building models

Nabu

Nabu is an ASR framework built on Tensorflow with existing implementations of some end-to-end speech to text recognition models. The 4 major divisions of the framework based on the sub-tasks it achieves are data preparation, training, testing, and decoding.

For each of the mentioned tasks, there exists a recipe that defines functions and modules being used. For instance, data preparation recipes include the name of the data processors to call during feature extraction. The essential elements to the Nabu framework are its user-friendly CLI tools, adaptive data pipelines, and code modularity through 'recipes.' Also, Kaldi ASR framework is compatible with Nabu, thus giving access to modules not available in Nabu.

Nabu is one of the only structured codebases with the implementation of the LAS model, which we would be using as a baseline. For more details, refer to the framework documentation at https://vrenkens.github.io/nabu/.

Fairseq

Fairseq [21] is a recent framework by Facebook, allowing researchers to design custom models for sequence modeling tasks such as translation, summarization, language modeling, and other text generation tasks. The framework is a good option for current research work in ASR as they have shown reference implementations of over 19+ research papers, some of which are still the state-of-the-art in some specific domains. For example, the toolkit provides their original implementation of the wav2vec model and commands to modify the training & model structure accordingly. For more details on the Fairseq framework, refer to the following link. https://github.com/pytorch/fairseq

Espresso

Espresso [33] is an ASR toolkit based on the PyTorch and the neural machine translation toolkit Fairseq. The fairly new toolkit, dated 15 October 2019, uses only Kaldi as its dependencies, unlike other prevalent toolkits like Espnet that depend on Chainer as well. Moreover, Espresso plans to go independent of any such dependencies. Attributing the complete modularity of Espresso framework, various language models are incorporated easily in the framework. Espresso already contains a CNN-LSTM Encoder and an LSTM Decoder with Attention. Other advantages of Espresso includes parallelization and distributed training and decoding for quick turnaround of experiments. We plan on adding the LAS class in the speech models class of the Espresso in the future.

3.3 Pre-training procedure for the wav2vec model

We implemented the architecture of wav2vec proposed in [28] and described in 2.1.2 using the setup from fairseq library [21] in Pytorch. The model is trained for 15 million update steps, and uses Adam [15] as its optimizer. The trainer uses a cosine learning rate scheduler [17] annealed over 15 million update steps starting with a learning rate of 10^{-7} and then warming up gradually to $5 * 10^{-3}$ and then decay it following a cosine curve to 10^{-9}. The contrastive loss uses ten distractor samples and trains on predicting the actual sample from the collection of negative samples and correct speech sample.

We trained our multi-lingual wav2vec model on a single Nvidia GeForce 1080Ti GPU with 6 Gb memory, requiring us to decrease the number of audio sequences per batch compared to [28]. We used 0.5 million as the maximum number of frames per batch and 0.1 million as the maximum frames each sample can hold in a batch. Apart from the datasets, the pre-training procedure is the same for both ASR and LID tasks.

The work reported in [28] has shown that utilizing more speech data for training wav2vec creates better feature representations. Given that we used a limited amount of data and no specific speech dataset for English speech as a part of LID competition rules, there is a very high possibility of result improvements if other sources of speech dataset are taken into consideration.

3.4 Training details for acoustic models

Before training, the input data is transformed into compact representations using wav2vec for our model. Similarly, the same is done for the baseline model using log-mel filterbanks. These feature representations act as the input data for the training of the LAS model to perform character level prediction of which language the utterance belongs. Here, we specify the exact parameters used for the training of acoustic models. We use the Nabu toolkit implementation of LAS for training and evaluation of the baseline as well as our model. The listener module of LAS consists of 2 pyramidal BLSTM layers with a non-pyramidal BLSTM layer stacked upon the pyramidal layers, each layer having 256 nodes. We added a Gaussian input noise with a standard deviation of 0.6 during training and a dropout rate of 0.5. The speller module consists of two layer LSTM with 128 nodes each, with a dropout rate of 0.5. The trainer uses an exponential decay learning rate scheduler with a decay constant of 0.1 and batch size of 32.

3.4.1 Automatic speech recognition model

Both our proposed LAS model & baseline LAS model are trained for 100 epochs to minimize average cross-entropy loss on monolingual ASR task at character level. The model predicts 29-dimensional character probabilities for each character, 26 of which represent the phonemes of English language, and other three represent blank, silent, and 'end of statement' $< eos >$ labels. However, during inference and test runs, we used a simple beam search decoder implementation described in 2.3.1 with a beam width $\beta = 16$ and maximum character length output to be 300.

3.4.2 Language identification model

The proposed & baseline model are trained for 100 epochs to minimize average cross-entropy loss on LID task at the utterance level. The model predicts 5-dimensional character probabilities for each character, 2 of which represent the language to which the utterance belongs, and other three represent blank, silent, and 'end of statement' $< eos >$ labels. For example, in our experiments with Telugu-English datasets, a typical output/label of length n may look like $y = \{< sos > y_1 y_2 .. < eos >\}$ where $y_i \in \{$T, E, _ , sil$\}$ and $y_n = \{< eos >\}$. However, during inference and test runs, we used a simple beam search decoder implementation described in 2.3.1 with a beam width $\beta = 4$ and maximum character length output to be 300.

Chapter 4

Metrics and final results

4.1 Automatic speech recognition task

We evaluate our model with pre-trained wav2vec representations and the baseline with log-mel filterbanks on the ASR task for Libri speech dataset. We measure the performance on the '*nov-92*' dataset using Character error rate (CER) as the metric. The CER is defined as:

$$CER = (i + s + d)/n \tag{4.1}$$

where n is the total number of characters, the minimal number of character insertions i, substitutions s and deletions d required to transform the reference text into the output. The table below mentions the CER values by these models.

Results on	
Model Name	Character Error rate
Baseline model (10 epochs)	**0.750**
Proposed model (10 epochs)	0.752
Baseline model (100 epochs)	0.574
Proposed model (100 epochs)	**0.524**

Table 4.1: Results on ASR task for WSJ 'nov-92' English speech dataset. All models are trained on 5 hours of Libri dataset.

The observations present an improvement in the character error rate of the LAS model as the proposed model performed far better than the baseline model with filterbank features. When we compared the results at 10 epochs, the baseline showed better performance than our proposed model by a small margin. The increased size of feature vectors may be one of the possible reasons behind the lagging performance of the proposed model, thus indicating that LAS

models with wav2vec training may need more training epochs when it comes to more complex tasks like LID. As mentioned earlier, the filter-bank feature vectors are 123-dimensional, while the wav2vec features are 512-dimensional, thus increasing parameters and slowing the learning pace for initial epochs.

4.2 Language identification task

We evaluate our model with pre-trained wav2vec representations and the baseline with log-mel filterbanks on the LID task at utterance level, on a Telugu-English code-switched dataset. Though our wav2vec module was trained on all 3 languages; Gujarati, Telugu and Tamil, we were not able to perform evaluation for other languages due to time restrictions. We measure LID performance on the dev split of Telugu-English dataset using various metrics such as character error rate, accuracy and equal error rate. We also compare our performance with another baseline model, DeepSpeech baseline as mentioned in 2.2.2, an end-to-end multi-layer model [1] consisting of 5 layers of LSTM, each consisting of 1024 neurons and trained using CTC loss.

4.2.1 Various comparision metrics

The models are trained with the goal of minimizing average cross-entropy error, that in turn helps decrease character error rate. During the evaluation, the beam search decoder outputs a character sequence as output, which we compare to the ground truth sequence using Levenshtein distance, which is usually used to calculate Character error rate. The CER is defined as mentioned in Section 4.1.

Another metric that we used for comparing models is accuracy. Accuracy is defined as predicting whether a speech is monolingual or code-switched, aka binary classification. However, as our models are trained for optimizing CER task, we output 1 for those samples whose output character sequence has both 'E' and 'T', while 0 for a monolingual character sequence. We approach the sequence labels in the same way and calculate the accuracy as

$$accuracy = \frac{No\ of\ correct\ predictions}{Total\ number\ of\ samples} \tag{4.2}$$

Similarly, equal error rate (EER) is defined as:

$$EER = \frac{\frac{TFR}{T} + \frac{TFA}{T}}{2} \tag{4.3}$$

	CER	Accuracy	EER
DeepSpeech Baseline	–	76.8%	11.6%
LAS + filterbanks Baseline	0.31	81.8%	9.1%
LAS + Wav2vec **Proposed**	0.46	69.2%	15.4%

Table 4.2: Results on utterance level LID classification task on Telugu-English code-switched dataset. All acoustic models are trained for 100 epochs except the DeepSpeech baseline model, which is trained for 40 epochs, provided by Interspeech LID competition. The numbers are being reported on the dev set.

where TFR = Total no of false rejects, TFA = Total no of false accepts and T = Total no of samples.

4.2.2 Training and validation graphs

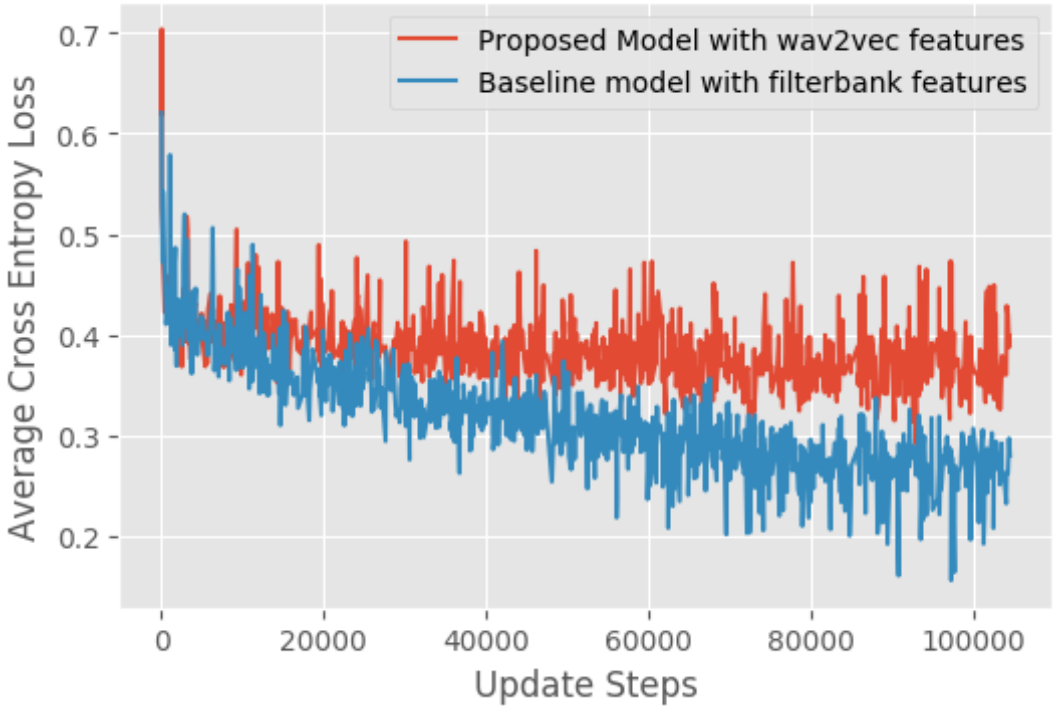

Figure 4.1: Training Loss graphs which represents the average cross entropy loss of the proposed model and baseline model vs update steps. Each epoch update is equal to 1045 batch update steps. Graphs are extracted from tensorboard.

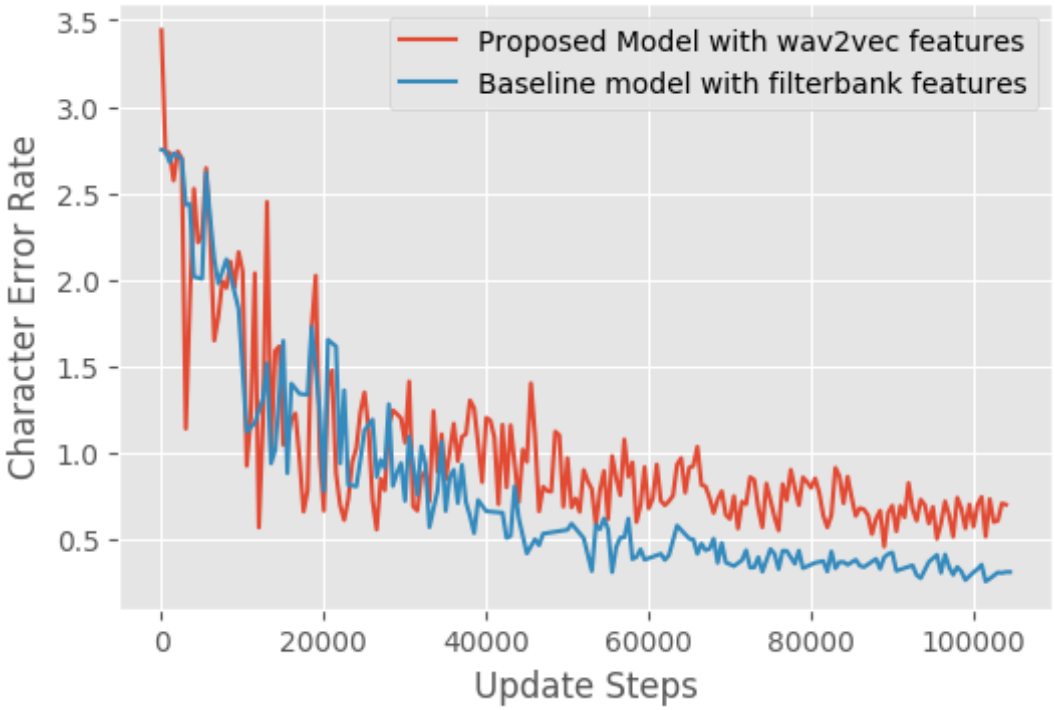

Figure 4.2: Validation Loss graph which represents the character error rate of the proposed model and baseline model vs update steps. Each epoch update is equal to 1045 batch update steps. Graphs extracted from tensorboard.

Chapter 5

Conclusion and future work

Our project exploiting a novel method of a learnable feature extractor based on unsupervised learning, and evidently demonstrate that it outperforms naive rule-based feature extractors like log-mel filterbanks. Initally, we thoroughly researched the field of ASR and various methods for decoders and encoders. We further completed an intensive literature survey and incorporated and studied various models, including both CTC losses and phoneme probabilities. As demonstrated by the results on ASR using the LAS model stacked over the wav2vec, the accuracy and error rates were much better than the baseline, which used a log-mel filterbank with the LAS acoustic model.

Motivated by the achievements of wav2vec in the ASR task, we exploited the wav2vec model for the multilingual LID tasks as well. We tested the hypothesis that various languages share some common representations through learned features, and training the wav2vec with multiple languages as well some code-switched data may actually improve performance. The major hindrance we faced was the requirement of extensive compute resources and prolonged training time of wav2vec, and so we used a smaller version of wav2vec for the LID task. The available dataset was largely biased towards a single language, and we received merely a few examples of other languages. We trained the wav2vec model with data much less than the previously trained wav2vec model for the ASR task and achieved comparable results with the baseline. Due to scarce data, the feature extractor could not learn better features. During the wav2vec training, the loss function followed a steep and a monotonic descend, which clearly implies that wav2vec training is not completed. If time permitted, then we could extract better features for improved accuracy and significantly low error rates. Lastly, the wav2vec was trained on three languages: Gujarati, Telugu, and Tamil, but was not trained on English seperately. Whereas, the baseline was trained and tested on code-switched English and Telugu only. This resulted in some misleading features in the wav2vec model and thus degraded the performance.

Future Works

- If enough resources are available, wav2vec large would be trained on a much larger and equalized dataset to achieve a significant improvement in performance.
- Moreover, the parameters of wav2vec are frozen while LAS training due to limited computational resources. In the future, the wav2vec fine-tuning may give improved results.
- Further, the only module of the proposed model that is not learnable is the beam search decoder. We would implement a differential beam search decoder for making the proposed model completely end-to-end.
- Lastly, we plan to integrate the existing models in a single framework to increase the controllability and optimization of our model. Currently, our wav2vec is built on PyTorch in Fairseq toolkit, while the rest of the LAS model is built on Tensorflow in the Nabu toolkit. The severe challenges posed to unify the toolkits are the different framework and dependencies for both the models. Once united on a potential framework, Espresso or Fairseq, these frameworks can readily support the extension for multilingual models on our current designs.

Bibliography

[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

[2] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019.

[3] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, 2016.

[4] Kenneth Church. Stress assignment in letter to sound rules for speech synthesis. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2423–2426, 1986.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. *arXiv preprint arXiv:1903.09722*, 2019.

[8] William F Ganong. Phonetic categorization in auditory word perception. *Journal of experimental psychology: Human perception and performance*, 6(1):110, 1980.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[10] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with bidirectional lstm. In *Automatic Speech Recognition and Understanding Workshop*, 2013.

[11] Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. Multilingual acoustic models using distributed deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8619–8623, 2013.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[13] Arthur S House and Edward P Neuburg. Toward automatic identification of the language of an utterance. i. preliminary methodological considerations. *The Journal of the Acoustical Society of America*, 62(3):708–713, 1977.

[14] Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. Reconsidering language identification for written language resources. *European Language Resources Association*, 2006.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[17] Ilya Loshchilov and Frank Hutter. Sgdr: stochastic gradient descent with restarts. corr abs/1608.03983 (2016). *arXiv preprint arXiv:1608.03983*, 2016.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[19] Thomas Niesler and Daniel Willett. Language identification and multilingual speech recognition using discriminatively trained acoustic models. In *Multilingual speech and language processing*, 2006.

[20] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[21] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.

[22] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[23] Douglas B Paul and Janet M Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

[24] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019.

[25] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI (preprint)*, 2018.

[27] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028, 2018.

[28] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[29] Gary F. Simons and Charles D. Fennig. *Ethnologue Global Dataset*. SIL International, 2017.

[30] Panu Somervuo, Barry Chen, and Qifeng Zhu. Feature transformations and combinations for improving asr performance. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[31] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[32] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2016.

[33] Yiming Wang, Tongfei Chen, Hainan Xu, Shuoyang Ding, Hang Lv, Yiwen Shao, Nanyun Peng, Lei Xie, Shinji Watanabe, and Sanjeev Khudanpur. Espresso: A fast end-to-end neural speech recognition toolkit. *arXiv preprint arXiv:1909.08723*, 2019.

[34] Fuliang Weng, Harry Bratt, Leonardo Neumeyer, and Andreas Stolcke. A study of multilingual speech recognition. In *Fifth European Conference on Speech Communication and Technology*, 1997.