End to End Speech Recognition using Connectionist Temporal Classification for Low Resource Phonetic Languages

A Project Report

submitted by

RACHCHH BHAVYA PRADEEPBHAI

in partial fulfilment of the requirements
for the award of the degree of

Dual degree (B.Tech + M.Tech)

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **End to End Speech Recognition using Connectionist Temporal Classification for Low Resource Phonetic Languages**, submitted by **Rachchh Bhavya Pradeepbhai**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual degree(B.Tech + M.Tech)**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. S. Umesh
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. S. Umesh for the continuous support during my Dual degree study, for his patience, motivation, and immense knowledge. His guidance helped me throughout my project and writing of this thesis. I could not have imagined having a better advisor and mentor for my Dual degree study.

I would also like to thank Vishwas and other Speech Lab students for their continuous help and support throughout my project.

I would like to thank IIT Madras for providing me outstanding education and best professors.

ABSTRACT

KEYWORDS: End to End Speech Recognition ; CTC; Low Resource; BiLSTM

Speech recognition has always been a complex system because of the multi-step traditional framework of Automatic Speech Recognition. However, because of the advent of Back propagation to train Deep Neural Networks in recent days and tremendous rise in the computation power of the modern systems, deep recurrent networks have been analyzed to be used in different stages of ASR and even as a complete end to end solution. End to end recognition has been shown to outperform tradition ASR with high amount of data (1000 hrs) but we show here that even with less data (50 hrs) for phonetic languages, Connectionist Temporal Classification performs at par with traditional ASR.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The problem of speech recognition is a prominent problem with the increasing use of digital devices and interactions with them. The speed of inputs through speech has been recorded to be approximately four times faster than typing out the text which gives a clear idea on the impact created by solving the problem. This section looks at the problem definition in detail.

1.1 Problem setting

Speech recognition can broadly be defined as the process of converting from speech to text. Consider that we have audio file which contains the spoken word "viva", the aim of the recognizer is to convert it into the machine encoded word "viva".

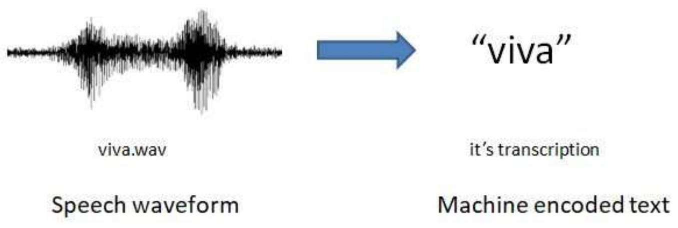


Figure 1.1: A high level view of a Speech Recognizer

Basically the problem is aimed at converting speech/audio files to corresponding transcripts with minimum error. This problem is aimed by solving it using the learning framework. The generic learning framework requires a few basic things to build a model:
1. Dataset of Inputs and their true (desired) Outputs
2. Definition of prediction error from the true inputs and definition of the loss function
3. An algorithm to minimize the loss function
Now the problem of speech recognition is not a simple problem since each input cannot

be mapped directly to the corresponding output. For example, there were 20K frames of input and it corresponded to four characters of output. Hence the number of inputs are not the same as the number of outputs. It is a case of sequence to sequence modelling. Also, mapping a fixed number of frames to a particular output does not work since the speed of speech for different individuals can be very different. Hence, the alignment has to be known beforehand (or has to be known by training another model). Thus, the process of getting an effective speech recognizer is much more difficult.

We have many terminologies and classification used in this domain of speech processing. We will see them one by one.

1.2 Terminologies and Classification

1.2.1 Speech quality

The speech input can be of two types depending on the number of speakers and the quality of speech:
1. Read speech:

This is the better quality of speech input where the speaker is speaking while reading the prompts, usually in a noise free environment and just one speaker speaking without a lot of errors/stammers.
2. Conversational speech:

This is the telephonic quality of speech or speech in a conversation where there may be multiple speakers in a single audio and/or there may be considerable amount of spoken noise involved.

Among these, Conversational speech recognition is obviously the more difficult problem since the input is of poorer quality and it may require an additional step of speaker diarization to mitigate the effects of multiple speakers. However, for the purpose of this thesis, we will only consider read speech as our inputs and hence we can assume homogeneity in terms of speaker for a particular utterance.

1.2.2 Amount of speech data

Based on the amount of data, the speech recognizers can be mainly classified into:
1. High/Abundant Resource Speech Recognition:

If the amount of data available is very high/abundant, it falls under abundant resource speech recognition. Usually, abundant amount of data is available for languages such as English, Spanish, German, etc.
2. Low Resource Speech Recognition:

If the amount of data available is very low, it falls under low resource speech recognition. Data from majority of the languages all over the world fall under this category as it is expensive to obtain highly annotated data. Most of the Indian languages fall under this category.

In this thesis, we will focus on Indian low resource languages. It is worth noting that low resource speech recognition is considerably more difficult than high resource languages, since the networks train considerably better as the amount of data increases.

1.2.3 Output labels

Based on the type of output labels used, the speech recognizers can be classified into:
1. Phoneme (phone) based recognizers:

These recognizers first convert the speech into a string of phones and then find out the series of words the phones correspond to. Phoneme based recognizers need an additional input, Lexicon, which defines a map from Graphemes to Phonemes.
2.Grapheme (character) based recognizers:

These recognizers try to get the string of characters directly from the model and thus they do not require an additional Lexicon since they learn the Grapheme to Phoneme mapping by themselves.

1.2.4 Word Error Rate

Word Error Rate is the principal error measure used across the domain of speech recognition. Word Error Rate is defined as the fraction of number of insertions, deletions and substitution to the total number of words.

The performance metric used in this thesis is also Word Error Rate which will be mentioned in the Results section.

CHAPTER 2

Traditional Hybrid HMM framework

2.1 GMM-HMM

The first and the oldest model used for speech recognition is the GMM-HMM Model. This section discusses briefly about this traditional framework used for speech recognition.//

In this framework, the phones are assumed to be the hidden states along with the Markov assumption while the emission probability (the acoustic model) is modelled by a mixture of Gaussians. The phone transition probability describes the dynamics of the system. The GMM model captures the stationary component (acoustic features) while the HMM model captures the dynamics of the recognizer.
Given below is how the ASR pipeline looks like:

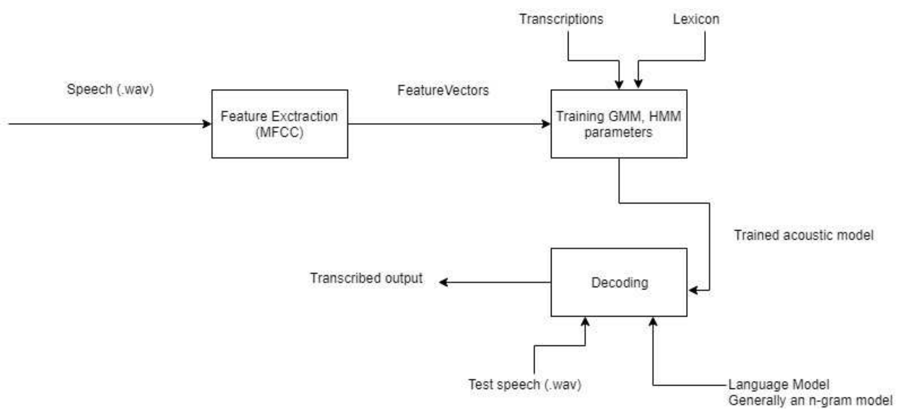


Figure 2.1: ASR Pipeline

As the raw audio is fed to the model, it gets converted (encoded) to a vector of very low dimension compared to the original one by calculating the Mel Frequency Cepstral Coefficients (MFCC features). To do this, a moving hamming window (25 ms) is passed through the audio, with a significant overlap (10 ms), a series of encoding steps are carried out (like DFT, calculating energies in a particular spectrum, and finally applying DCT to select the first 13 coefficients. Later, delta and delta-delta are added to have a localised derivative effect (dynamic consideration, a bit of context) in the input. The features obtained this way are called MFCC features.

The MFCC features are finally being fed to the training algorithm for setting the HMM and GMM parameters. Note that, we require a Lexicon (Pronunciation Model) and a Grammar (Language Model, usually n-gram model) along with speech data and transcripts. Lexicon is basically a mapping from words (a string of graphemes) to phones (a string of phonemes) and Grammar then incorporates the probability of a word given the previous words observed.

All the above information is encoded as a finite state transducer and can be effectively encoded in order to decode it easily later. The GMM parameters are calculated by Expectation Maximization while HMM parameters can be calculated by forward-backward algorithm. Viterbi algorithm, which is a dynamic programming approach can be used to decode the output to a test input.

Now, we see that the contextual information is not properly captured in this model as the only context we have is the forward transition probability. So, in order to consider for this, we use Context Dependent phones. To understand context dependency, consider triphone model which has different states for all possible left and right contexts. Note that moving from monopone to triphone, the number of states increases from N to N^3. Now, to model a contextual phone with a single state is an oversimplification and does not perform well. So we use a tristate modelling along with the triphone model, which assumes that each contextual phone consists of three states in transition. This further increases the number of states to $3N^3$.

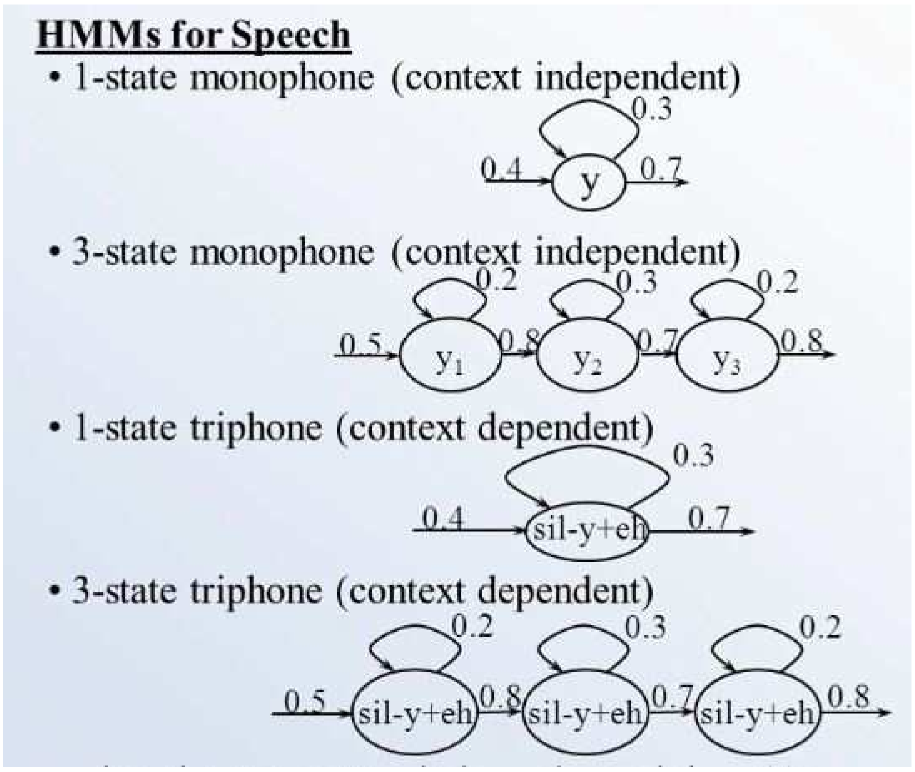

HMMs for Speech
• 1-state monophone (context independent)
0.3
0.4
y
0.7
• 3-state monophone (context independent)
0.2
0.3
0.2
0.5
y1
0.8
y2
0.7
y3
0.8
• 1-state triphone (context dependent)
0.3
0.4
sil-y+eh
0.7
• 3-state triphone (context dependent)
0.2
0.3
0.2
0.5
sil-y+eh
0.8
sil-y+eh
0.7
sil-y+eh
0.8

Figure 2.2: Triphone tristate modelling

Although the representation power of the dynamics of the system increases drastically by using triphone tristate modelling, it also results in a tremendous increase in computation required for training and decoding. Not just the computation, the data required to have a good estimate of the parameters also shoots up so as to not overfit the parameters. Now, the above two problems are tackled by introducing phonetic decision trees, which tie together states which are acoustically indistinguishable, which means they share the same parameters.

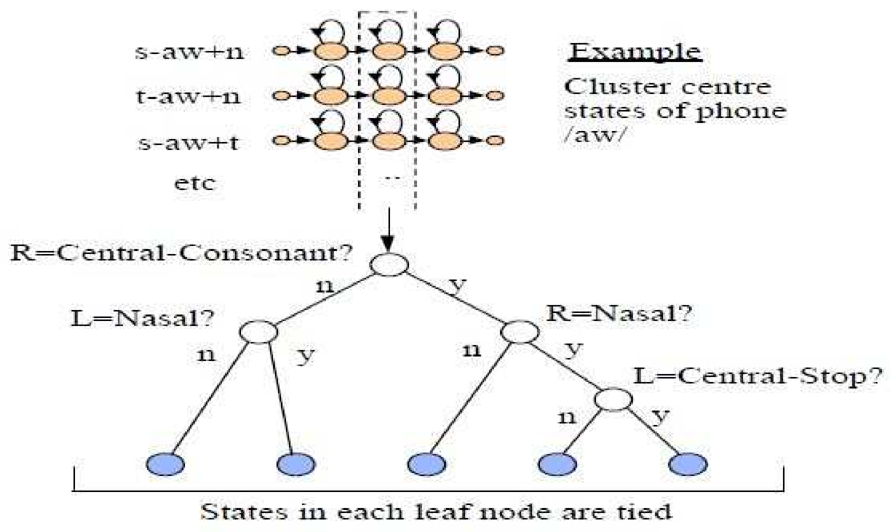


Figure 2.3: Phonetic decision tree

2.2 Problems with GMM-HMM and DNN-HMM

For modelling the acoustics, mixture of Gaussians was used in place of Gaussians because the frequency data (voiced phones) is inherently multi-modal. Hence, we required a better representation power than just fitting a Gaussian. But the performance of GMM-HMM model was considerably sub par compared to the requirements and also the rise in the Neural networks due to the advent of back propagation changed the scenario for a lot of fields.

Deep Neural Networks (DNN) were now considered to replace the Gaussian Mixture Model (GMM) for modelling the acoustics. This provided the acoustic model a far better representation power compared to GMM as DNN can model any kind of non-linearity given enough number of layers (model complexity) and enough data (so as to reduce overfitting). This gave rise to increasing use of DNN-HMM hybrid rather than GMM-HMM hybrid. The performance obtained with DNN-HMM was considerably better than the performance obtain with GMM-HMM.

Initially, mainly feedforward neural networks were considered to model the acoustics. But it is easy to realize that feedforward networks, although they improve the representation power significantly, still lack contextual information or memory. It is not possible for information to persist in a Feedforward neural network and hence the alignment of speech and dynamics were always left to HMM to handle.

Next came Recurrent Neural Networks (RNNs) and the improved version of RNNs, Long Short Term Memory (LSTMs). The state of the art hybrid networks today are TDNN-LSTM (Time Delay Neural Networks chained with Long Shorm Term Memory units) coupled with HMMs.

However, the major shortcomings of this hybrid approach are
1. It is a multi-process task, hence significant amount of expertise in various fields are required to succesfully train the speech recognition model.
2. The alignments obtained using the previous step has to be fed back to the model and these alignments might have significant errors in them.
3. Since there are a lot of states (context dependency) the amount of time it takes to train and decode is considerably high. Decode time has a direct impact on the usability of the system for real-time decoding tasks.
4. Since the training involves a series of steps, an error in any step will cause propagation of the error to the remaining steps and might break the system down.

CHAPTER 3

End to End speech recognition

We have seen how GMM can be replaced by better neural network models and performance can be enhanced. The major aim of End to End speech recognition now is to address the shortcomings of the hybrid model. First of all, we aim to simplify the overall process to merge both acoustic modelling and context dependency into one.

There are two major ways to go about End to End speech recognition as mentioned below:
1. Connectionist Temporal Classification
2. Sequence to sequence transduction

We will focus and build using the first method, CTC in this thesis. Before the description of the CTC model gets underway, we will analyze how will we get to the CTC model.

3.1 Importance of BiLSTM

Our aim is to infuse context dependency alongwith state information in a single powerful unit. We will basically need a cell which can persist information, carry the information over time. Feedforward networks fail here, they treat each input as independent and the output at any instant depends only on the current input.

Now, to get around this, we use the recurrent neural networks. They use the previous state information and current inputs to determine the next step and in turn determine the output. The RNN cell is as shown below:

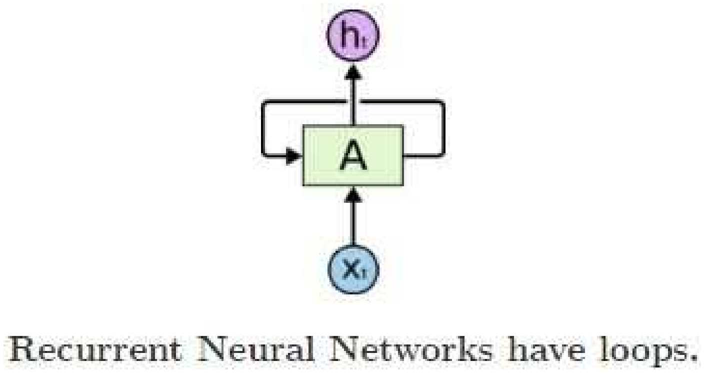


Figure 3.1: Recurrent Neural Network Cell

In speech, we require a very long term dependency as the frame rate can be very high and the speaker might have a slow pace in speaking. So, we analyze how RNN responds to long term dependency can whether it can support the dependency we require. Now, RNN suffers from this problem of exploding and vanishing gradients, which means, for RNN, sensitivity decays exponentially, which makes RNN incapable for longer context. The decay of sensitivity is shown in the figure below:

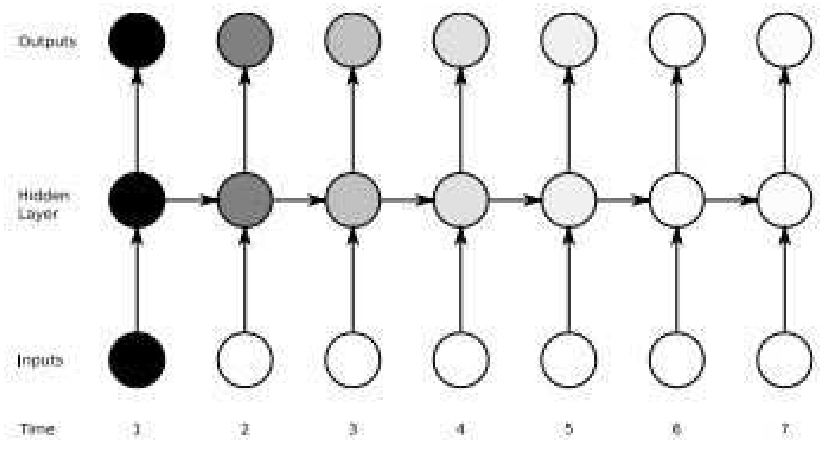


Figure 3.2: Sensitivity decay in RNN

Now, to solve the long term context dependency problem, a special kind of RNNs, Long Short Term Memory networks (also called LSTMs). They have been explicitly designed to avoid this problem. The design of an LSTM Cell is shown below:

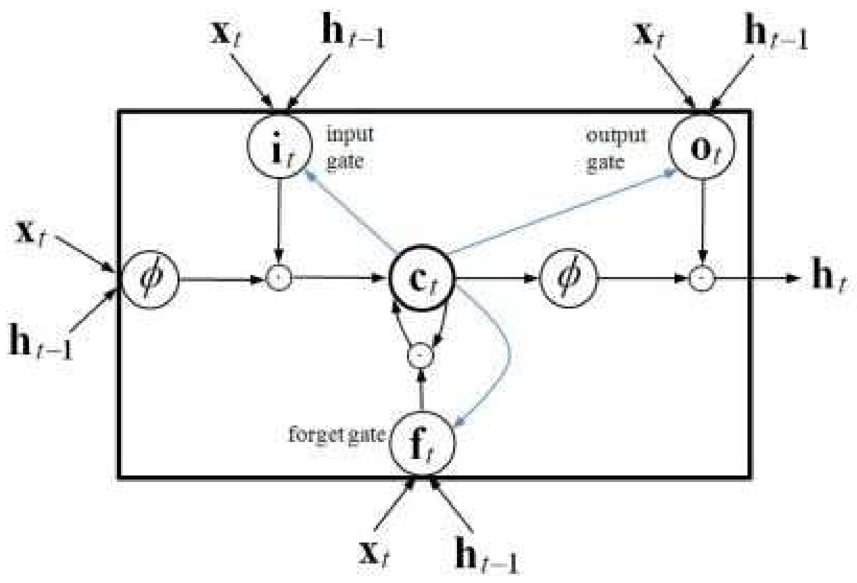


Figure 3.3: LSTM Cell

The equations governing the behaviour and output of the LSTM Cell is as shown below:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) && \text{input gate}\\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) && \text{forget gate}\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) && \text{memory cell}\\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) && \text{output gate}\\
\mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t)
\end{aligned}$$

Figure 3.4: LSTM Equations

The flexibility of having input, output and forget gate allows the cell to persist information in the cell memory for a long time. Now, there is one problem with this cell and the dependency it covers. It can consider past inputs, that is left context very efficiently, whereas the network has no information on the future inputs, that is the right context. In order to match this requirement, we use BiDirectional LSTMs (known as BiLSTMs). BiLSTMs are basically two independent layers, one taking inputs in the forward direction and the other layer in the reverse direction. The cell state from both the layers are then concatenated to have effects from both the forward context and the reverse context. The concatenated output is then fed to an output layer to get class probabilities. The BiLSTM cell looks like the following:

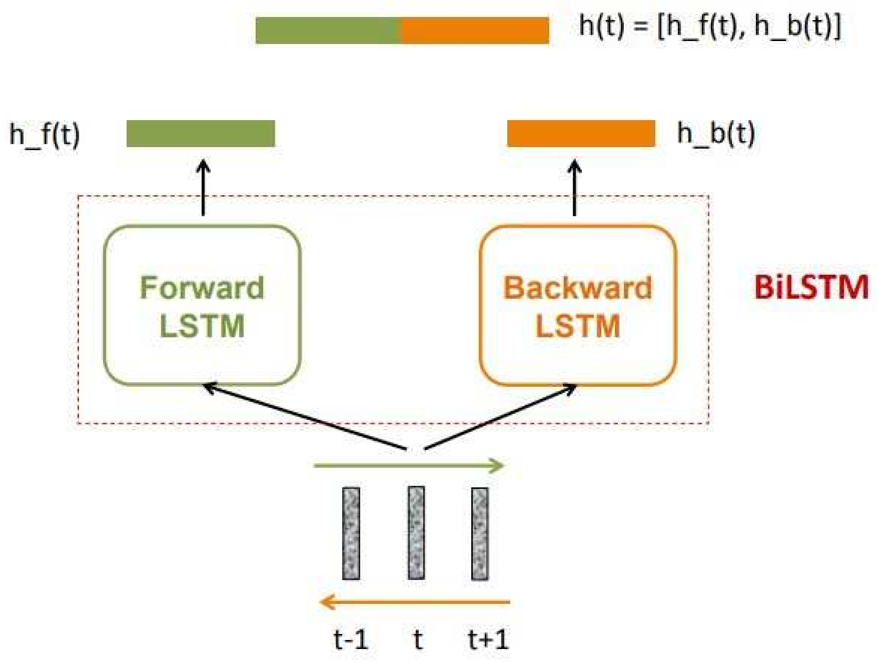


Figure 3.5: BiDirectional LSTM

Now, once we have the bidirectional output, the representation power of the BiLSTM Cell can be further improved by stacking multiple BiLSTM layers to make it a deep BiLSTM network in order to get a better acoustic and context modelling. The deep BiLSTM network looks like the one shown below:

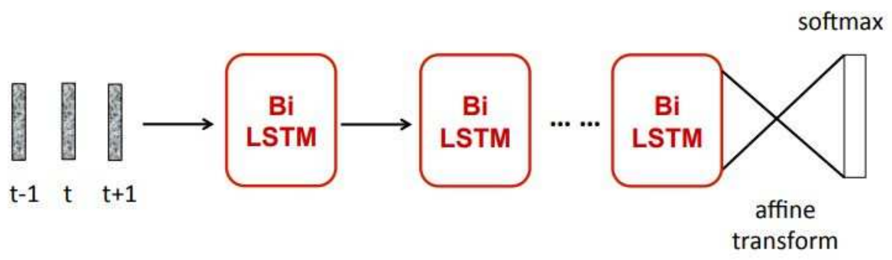


Figure 3.6: Deep BiLSTM

Note that the output layer in a generic deep BiLSTM layer used for classification is the softmax layer. The softmax layer converts a set of numbers on the real line to a point on the probability simplex.

3.2 Connectionist Temporal Classification

We need to map a sequence $(X_1, X_2, ..., X_T)$ (which are the Mel-Filterbank features) to a sequence $(Y_1, Y_2, ..., Y_U)$ (which is the desired transcription) where $U <= T$

Now, we don't have aligned input, hence, the complexity still remains the same as we can't emit a label every frame. Consider Grapheme (character) based speech recognition now. The easiest way to solve this problem is to allow repetition of labels and merge all the repeated labels after each frame has emitted a label. For example, if input had nine frames, i.e., $T = 9$ and $Y = viva$, thus, U=4, the possible output paths will be $"vviivvaaa"$, $"vvviivvva"$ and so on, as they will merge to give the desired output to be $"viva"$.

Now, this method will again pose an important problem, the output will never have a repeated letter. For example, even if the desired output was $"speech"$, the output can only be $"spech"$.

The other problem will be the network should always return a label irrespective of whether it is confident of emitting a label or not. This might introduce a lot of insertions and some substitutions.

To tackle this, we introduce and allow emitting a blank label (ϵ). We specify the rule such that all the repeated characters should be merged and then the blank labels should simply be eliminated. For example, if $n = 12$, $"ssppee\epsilon echh"$ and $"s\epsilon p\epsilon e e e e \epsilon e c c h"$ both collapses to $"speech"$.

The blank label will also represent silence.

Now there are two approaches in CTC as mentioned in the previous chapter: 1. Using phonemes as CTC Labels 2. Using graphemes as CTC Labels

CHAPTER 4

Experiments and Results - English

4.1 English language:- Wall Street Journal Corpus

The Wall Street Journal Corpus contains 80 hours of read speech using two microphones recording rich quality speech. The speech is taken from machine readable corpus from Wall Street Journal news text. The results, in terms of WER obtained using hybrid models and CTC for both character and phone based are given below:

1. Using hybrid DNN : 9.91
2. Using hybrid TDNN-LSTM: 7.32
3. Using CTC (phonemes) : 11.39
4. Using CTC (characters): 14.23

4.2 Analysis and inferences

We will first analyze why CTC (characters) model performs worse than CTC (phonemes) model. For the phonemes model, the translation from words to phones is provided externally and this translation has been provided and verified by linguistics experts and thus will be state of the art. On the other hand, the characters model needs to learn the graphemes to phonemes model on its own and is thus trying to solve a more difficult problem with the same data. Hence, the WER associated with characters model is higher.

We next analyze why CTC performs worse with respect to hybrid models. In CTC, the BiLSTM tries to learn the context dependency on its own. This context dependency requires a lot more data to be learnt by the network itself. Moreover, CTC was trained on context independent phones while hybrid models were trained on context dependent phones (senones) so they are clearly not on level playing fields. Even with switchboard data (300 hours), hybrid model performs better. However, it has been shown by Google and Baidu that CTC outperforms state of the art hybrid models around 1000 hours of data. So, clearly CTC requires higher amount of data than hybrid models.

4.3 Grapheme to Phoneme Mapping

We will dig the disparity between phone model and character model further in this section.

Ideally, if the Grapheme to Phoneme mapping is one-to-one, character model should work at par with the phone model. But, the presence of homophones in English makes the mapping many to one, whereas multiple pronunciation of the same word makes it many to many.

For example, word "read"(past tense) and "red" have the same pronunciation, or in other words their phonetic transcription is identical which makes them homophones. Now, the word "read"(present tense) and "read"(past tense) have the same character representation but differing phonetic transcription. This constitutes multiple pronunciation of a single word.

As a combined effect of both of these, the G2P is many to many mapping which is very hard to learn. Hence, for English there is a significant difference between phone and char models.

CHAPTER 5

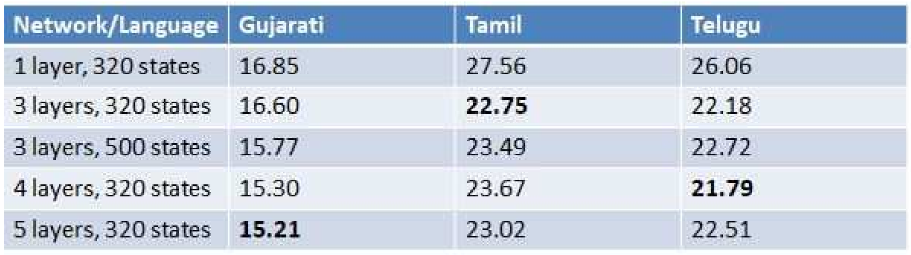
Experiments and Results - Phonetic languages

We will now look at Indian languages: Gujarati, Tamil and Telugu and try to build models for low resource data. The amount of annotated speech data available for Indian languages is very limited, hence this comes under the low resource setting.

The languages in which one can infer the pronunciation exactly by looking at the graphemes and infer the spelling by hearing the word are called phonetic languages. English is not a phonetic language, but Indian languages like Gujarati and Telugu are highly phonetic. In phonetic languages, basically graphemes to phonemes have a one to one map.

The dataset that is being used for this project is 40 hours of read speech data each for Gujarati, Tamil and Telugu provided by Microsoft as part of the Interspeech Conference challenge.

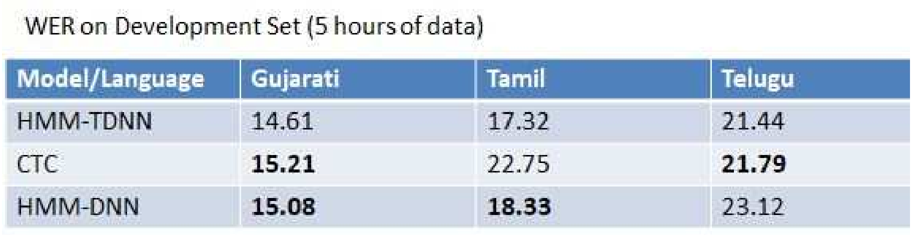
The results obtained by building end to end models using CTC with differing number of layers and number of states in each direction for all three languages are shown below:



Network/Language	Gujarati	Tamil	Telugu
1 layer, 320 states	16.85	27.56	26.06
3 layers, 320 states	16.60	**22.75**	22.18
3 layers, 500 states	15.77	23.49	22.72
4 layers, 320 states	15.30	23.67	**21.79**
5 layers, 320 states	**15.21**	23.02	22.51

Figure 5.1: CTC models for different hyperparameters

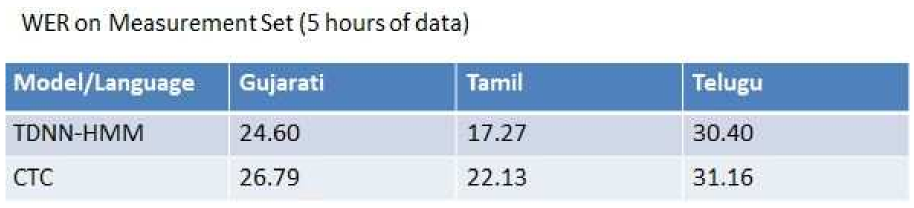
Summarising the best models obtained using CTC with the best models using state of the art hybrid models and hybrid models using DNN for the development set:


WER on Development Set (5 hours of data)

Model/Language	Gujarati	Tamil	Telugu
HMM-TDNN	14.61	17.32	21.44
CTC	**15.21**	22.75	**21.79**
HMM-DNN	**15.08**	**18.33**	23.12

Figure 5.2: Comparing WERs on Development Set

Similarly, comparing the best model obtained using CTC and state of the art hybrid models for measurement set:


WER on Measurement Set (5 hours of data)

Model/Language	Gujarati	Tamil	Telugu
TDNN-HMM	24.60	17.27	30.40
CTC	26.79	22.13	31.16

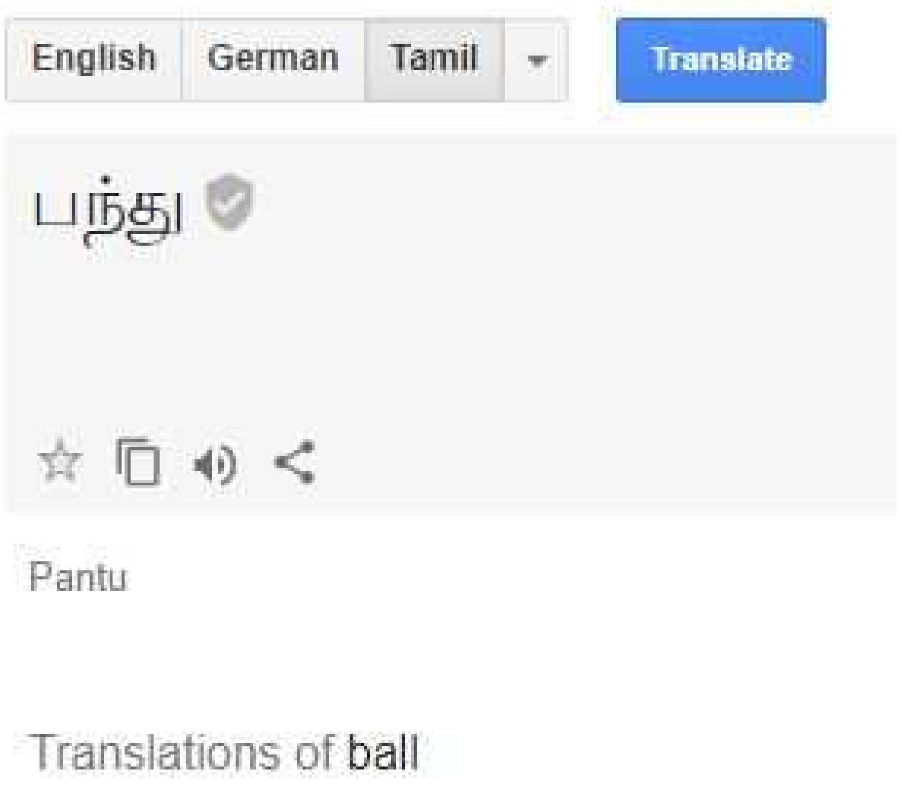
Figure 5.3: Comparing WERs on Measurement Set

5.1 Analysis and Inference

As can be seen from the results, CTC performs at par with hybrid DNN models (except Tamil) and comparable to state of the art hybrid models for both Development and Measurement set. Note that this model is trained only on 40 hours of data. Because of the phonetic nature of the languages, the context dependency is much easier to learn than English which is apparent from the results shown above. Note that, for phonetic languages there is no difference between character model and phone model. Thus, for phonetic languages, CTC works at par with state of the art hybrid models even with very less data. It is worth mentioning that CTC was trained only on context independent phones (around 40) while hybrid model was trained on 5000 states. Yet the results are almost identical for both.

5.2 Discrepancy with Tamil

While the results are at par for both Gujarati and Telugu, for Tamil, they are quite off. The reason for this is for Tamil, the G2P mapping is not one one. For example, consider the translation of Ball into Tamil.



பந்து

Pantu

Translations of ball

Figure 5.4: Translation of Ball

The real translation of ball is "Bandu", but it can be mistakenly written as "Pantu", because the consonants used for "pa" and "ba" are same, also the consonants used for "ta" and "da" are the same. This introduces the classic case of multiple pronunciation, which we talked about in English. Which implies the G2P for Tamil is not one-to-one and thus learning context dependency is a much involved problem and hence it requires more data compared to HMM.

5.3 Speed Comparison

We mentioned that slow speed is one of the drawbacks of hybrid HMM models and if we want to use the speech recognizer for Real time decoding, we will need to build faster models. In this section, we compare the speed of decoding to see whether that objective has been fulfilled.

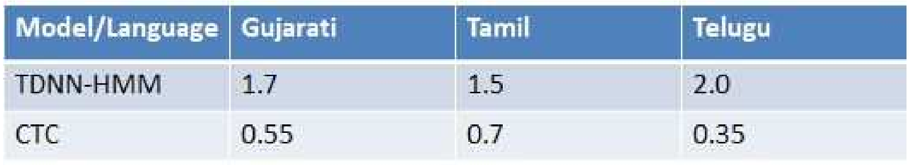
The decoding speed is compared by calculating the Real Time Factor(RTF) whose definition is mentioned below:

$$\text{RTF} = \frac{ElapsedTime * NumOfThreads * 100}{FrameSubsamplingFactor * FrameCount}$$

Figure 5.5: Real Time Factor

This means that for RTF=1, the decoding rate will be the same as the input frame rate. Also, RTF<1 is preferable since systems with RTF<1 can be used in real time.

The RTF obtained using CTC models and hybrid models are shown below:



Model/Language	Gujarati	Tamil	Telugu
TDNN-HMM	1.7	1.5	2.0
CTC	0.55	0.7	0.35

Figure 5.6: Real Time Factor Comparison

Thus, CTC models are considerably faster than the TDNN-HMM networks and they can be sed in real-time applications.

CHAPTER 6

Conclusions and Scope for Future Work

End to End speech recognition models using CTC have a comparative advantage for phonetic languages as they require lesser data to be at par with the hybrid models in terms of WER and they are also considerably faster than hybrid models. Moreover, training CTC models is much easier and they can be used in real-time decoding.

Multilingual and crosslingual speech recognition can be explored for CTC and the framework does support using data from different languages. Online decoding framework can also be designed as the BiLSTM-CTC model is inherently non-causal since it uses future inputs, solving this problem can be looked upon.

CHAPTER 7

References

1. Supervised Sequence Labelling with Recurrent Neural Networks, by Alex Graves 2. CTC Tutorial: https://distill.pub/2017/ctc 3. Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding," in Proc. Automatic Speech Recognition and Understanding Workshop (ASRU), Scottsdale, AZ; U.S.A., December 2015. IEEE. 4. LSTM Tutorial: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ 5. The Kaldi Speech Recognition Toolkit, by Daniel Povey et. al.