Acoustic Models for Speech Recognition in Reading Miscue Detection

MTP Stage 1 Report

Submitted in partial fulfillment of the requirements for

Master of Technology

by

Shreeharsha B S

(18307R002 EE1)

Under the guidance of

Prof. Preeti Rao

Department of Electrical Engineering

Indian Institute of Technology Bombay

December 2020

Declaration

I declare that this written submission represents my ideas in my own words. I have adequately cited and referenced the original sources where necessary. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shreeharsha B S
(18307R002)
Department of Electrical Engineering
IIT Bombay

Date: December 10, 2020

Acknowledgement

I express gratitude towards my guide, Prof. Preeti Rao, for her patience and guidance throughout the project. I also thank Nagesh Nayak and Kamini Sabu for their help and comments in this work. I would also like to thank Asha, Avinash and all other members of the DAP lab for their support. I also thank the members of the kaldi help group for answering the questions I had.

Shreeharsha B S

Abstract

Literacy is an essential skill for the betterment and prosperity of an individual. Improved literacy rates provide political, cultural, social and economic benefits. It acts as bridge by connecting people to the world and all the information available in it. In this work, the development of an automatic assessment system for evaluating the reading abilities of children (by analyzing audio recordings of literacy tests) in Hindi are examined.

The inherent difficulties involved in collecting children's data for research (which leads to a shortage of the data available) and the highly varying speaker characteristics and background noises of various types present obstacles for automatic assessment systems. These challenges are tackled in this work by building automatic speech recognition (ASR) systems using relatively more abundant adult speech datasets and adapting these systems to the specific use case. The ASR system used is the state of the art TDNN-HMM model in kaldi. Adapting the models to the target speech is done with the help of a limited amount of labelled target data using weight transfer techniques. Further data augmentation methods are explored to, both, improve the match between train and target data and to increase the adaptation data size. With these two techniques, improvements in the detection of reading miscues are obtained. This work also enlightens the road for further experimentation and future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Literacy is an important measure of prosperity and also critical to the well being of an individual and his/her community. It has been found that Children at the Bottom of the Economic Pyramid (BOEP) have limited access to good education standards, particularly in the field of language learning and reading. Annual Status of Education Report (ASER) is literacy test and survey conducted by a branch of the NGO (Non-governmental organization) Pratham. It is carried out by trained volunteers/surveyors. In 2018, this survey covered nearly 5,50,000 children in different districts across India, belonging to the 3-16 year age group [5]. The results from the survey have some worrying implications are: 27.2% of class VIII students sampled were unable to read a text that is meant for a class II student. These percentages are worse than what they were 10 years ago, "In 2008, 84.8% of Class VIII students could read a text meant for Class II; by 2014, only 74.6% could do so" [5] [6]. A more complete review of literacy surveys, discussion and implications of technology use can be found in [7].

The above results concerning reading skills were obtained by manual assessments of children selected across many districts and villages. With the use of automatic assessment systems (which use speech recognition and signal processing tools) reading skills can be measured while reducing the drudgery involved in individual assessments and the much more difficult problem of providing individual feedback on the reading assessment can be resolved. Automatic speech recognition is used to convert speech to text and can be used to quickly perform an assessment of the child and his/her ability to read a reference text. These quick assessments can also act as an objective measure of learning intervention programs and their effectiveness (or lack thereof) by the changes in the reading ability of children before and after.

Building an ASR system for children's speech is a challenging task because of the high degree of variability in their speech patterns which is explained due to the 'unrefined' motor control, during speech production, of children which gets more refined with age [8]. These difficulties are not as persistent when building systems for adult speech. Modern ASR systems use statistical methods to predict the text output of a speech signal. These methods are heavily reliant on the type of speech data available and its amount. This also implies that with a decent representation of all types of children's speech much of the acoustic variability can be handled and accounted for. The crux of the ASR system that models this variability is called the acoustic model. Acoustic models, in general, find mappings and relationships between speech features and linguistic units (phones or characters). With advancements in the complexity and modeling power of state of the art acoustic models, many recent research tasks have focused on building ASR systems for children's speech.

Obtaining a representative set of children's speech data is a challenging task. Regional accents, gender and age play a big role in the variability of data. A great addition to the scarce children's speech dataset is the ASER dataset [9]. Although it does not contain the transcripts of what the child spoke, the reference texts read by the child, information about the fluency and number of mistakes committed is available through limited manual annotations. This dataset is a salient part of this work and the methods employed are explained next.

1.1 Focus of this work

In this work, the main objective is to build an ASR system for the detection of oral reading errors using a limited amount of target speech. For this, state of the art neural network based acoustic models are examined. These acoustic models, trained on relatively easily accessible adult speech, are used as a baseline for the children's speech. For now, the data used in both the children's samples and adult samples come from the same language (Hindi). Then, techniques of modifying the adult speech data to better suit the target data and data augmentation techniques are examined. Apart from this, acoustic model adaptation methods which tune the baseline model to work better on the target speech using limited amounts of adaptation data including transfer learning techniques are also examined. A block diagram describing these is shown in Figure 1.1.

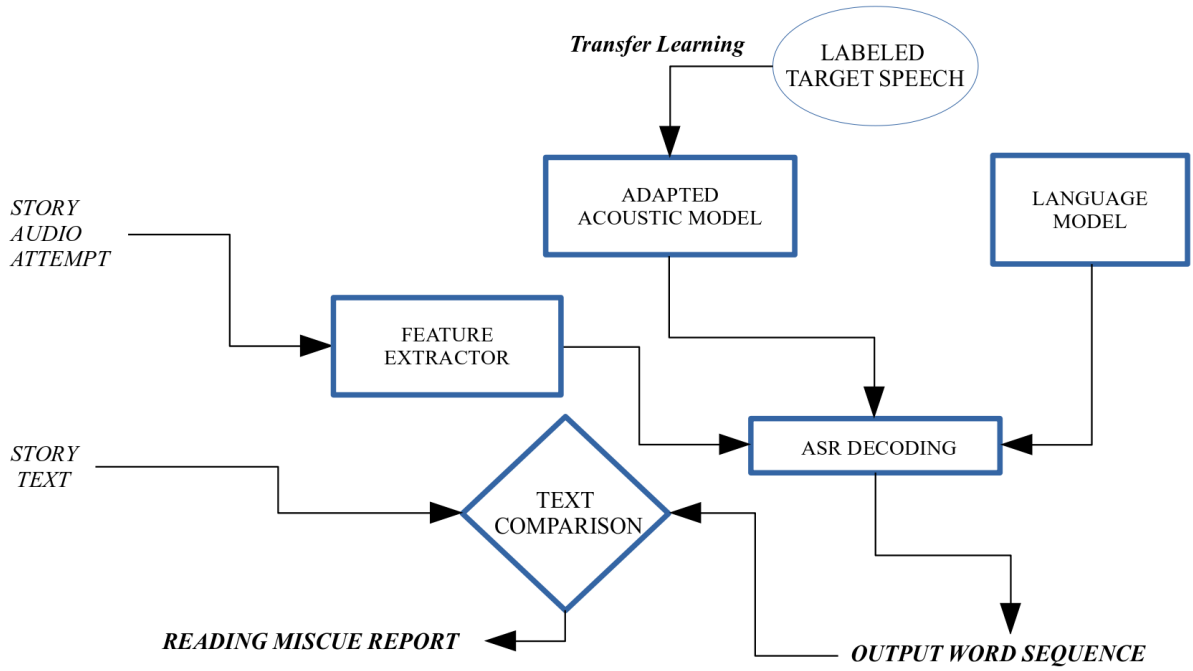

Figure 1.1: Block diagram of the proposed system

1.2 Report organization

There are seven more chapters in the rest of this report. Chapter 2 describes the training and adaptation datasets used in this work. Chapter 3 discusses the data augmentation strategies in use. Chapter 4 explains the acoustic model used in this work (kaldi TDNN model), its architecture and training methodologies. Chapter 5 details the transfer learning and adaptation techniques used in this work. Chapter 6 reports the experiments done, parameters used and the evaluation metrics used. Chapter 7 illustrates the results of the experiments and understandings developed. Chapter 8 concludes the work in this report along with future goals and directives.

Chapter 2

Dataset

As stated earlier, gathering and transcribing children's dataset is a challenging task. In this work a mixture of adult and children's speech data are used. Three datasets in Hindi are examined, two of which are children's speech and one consisting of adult speech:

2.1 IITM data

The IITM data set comprises of Hindi speech by volunteer adults who read predetermined segments chosen from newspapers. It was released as part of an (Automatic Speech Recognition) ASR challenge by the Speech Processing Lab of IIT Madras and was funded by the Ministry of Electronics and Information Technology (MeitY) [10]. Also provided are the corresponding transcriptions, a dictionary containing the lexicon and phone set and a recipe for building a baseline model. The text data covers a variety of genres like politics, sports, entertainment. Information about the gender, age and other characteristics of the speaker are unfortunately not available, however the region where the speaker came from is encoded as part of the name of the utterance recording (eg: cdg-Chandigarh, dli-Delhi, mum-Mumbai, pue-Pune).

All the recordings are sampled at 16 kHz sampling frequency. The recordings seem to be made in a closed room with very little noise in most of the recordings, although a very few of the recordings of some speakers have a low frequency background hum (in the 2-3 Khz range) that is noticeable. The IITM data set consist of three parts that were released as part of an ASR challenge:
Train Set: 40 hours
Dev Set: 5 hours
Eval Set: 5 hours

The train set is used to build a baseline TDNN model used in the challenge. There are 418 unique speakers in train set. Each utterance in the dataset is actually short segments from a larger set of recordings. These segments consist of one to three sentences (sometimes single words) lasting anywhere between 0.3 seconds to 15 seconds. The majority of the utterances are greater than one second, with only 57 out of the 27131 utterance segments being less than one second. Only the train set is used in this work and its stats are summarized in Table 2.2.

2.2 ASER data

The ASER data set [9] consists of recordings of children attempting the ASER (Annual Status of Education Report) literacy test which measures reading levels of children who are in the 6-14 years range. The recordings were obtained from children reading text of varying complexity in Hindi, Marathi and English which was displayed on a custom mobile app and recorded using a headset. The text complexity varies from individual letters and words to paragraphs and stories being read out. A sample is shown in Figure 2.1. Along

Figure 2.1: ASER sample survey [1]

with the recordings a JSON file is also available which contains information about the standard the student is studying in, the number of mistakes made while attempting the test and whether the intended text was spoken correctly or not as estimated by a trained 'surveyor'. More information about the survey process and how volunteers are trained can be found on the ASER website [1].

The recordings are sampled at 16Khz. Some of the recordings do have foreground speech by speakers other than the child taking the test, which adversely affects automatic reading assessment systems. There is also a variety of noises (stationary and non-stationary) in the recordings ranging from mic pops and bursts to vehicles, wind, babble and other noise types.

A subset of this ASER data set, consisting of Hindi story and paragraph recordings of children from Uttar Pradesh (UP) and Rajasthan (RJ) is used in this work. Furthermore, with the help of a baseline DNN-HMM [11] the recordings are categorized, based on the WER between DNN-HMM output and the story/paragraph canonical text, and those recordings with WER less than 80% have been transcribed to be useful for the speech recognition based experiments in this work.

Furthermore, the recordings are separated into skill-based categories in order to investigate the performance of the system in two different contexts. After the transcription process, those recordings with miscue rates (WER between the transcribed output and the story/paragraph canonical text) less than or equal to 20%, called a 'Ratable' set or High proficiency recording (HPR), and those with miscue rates greater than 20% and less than 80%, called a 'Non-ratable set' or Low proficiency recording (LPR) are created and used for experimentation.

From the above two sets (HPR and LPR) three subsets, each of which have no speaker overlap with the others, are derived for the transfer learning experiments in this work:
i) Train set, used for weight transfer/adaptation of the baseline TDNN model.
ii) Validation set, used for determining decoding parameters for the test set and reporting results. It is also used as a diagnostic during weight transfer experiments to tune the retraining parameters. Both the train and valid data sets are split at the sentence level using the manual transcriptions and sentences that contain irrelevant speech (IR)(regions which have distinct and audible speech sounds that are irrelevant to the story being read) as determined by the transcription process described in Section 2.2.1 are discarded from the train and valid set.
iii) Test set, which is also split at sentence level and sentences containing IR removed, to

report the results of the various experiments.

These subsets are created, after transcription, on both ASER UP an ASER RJ sets and their duration, speaker information and other information are summarized in Table 2.1. Figures 2.2 and 2.3 show the distribution of the number of recordings over the stories and miscue rates. A good representation of various proficiency levels and the stories/paragraphs spoken are present in each of the subsets.

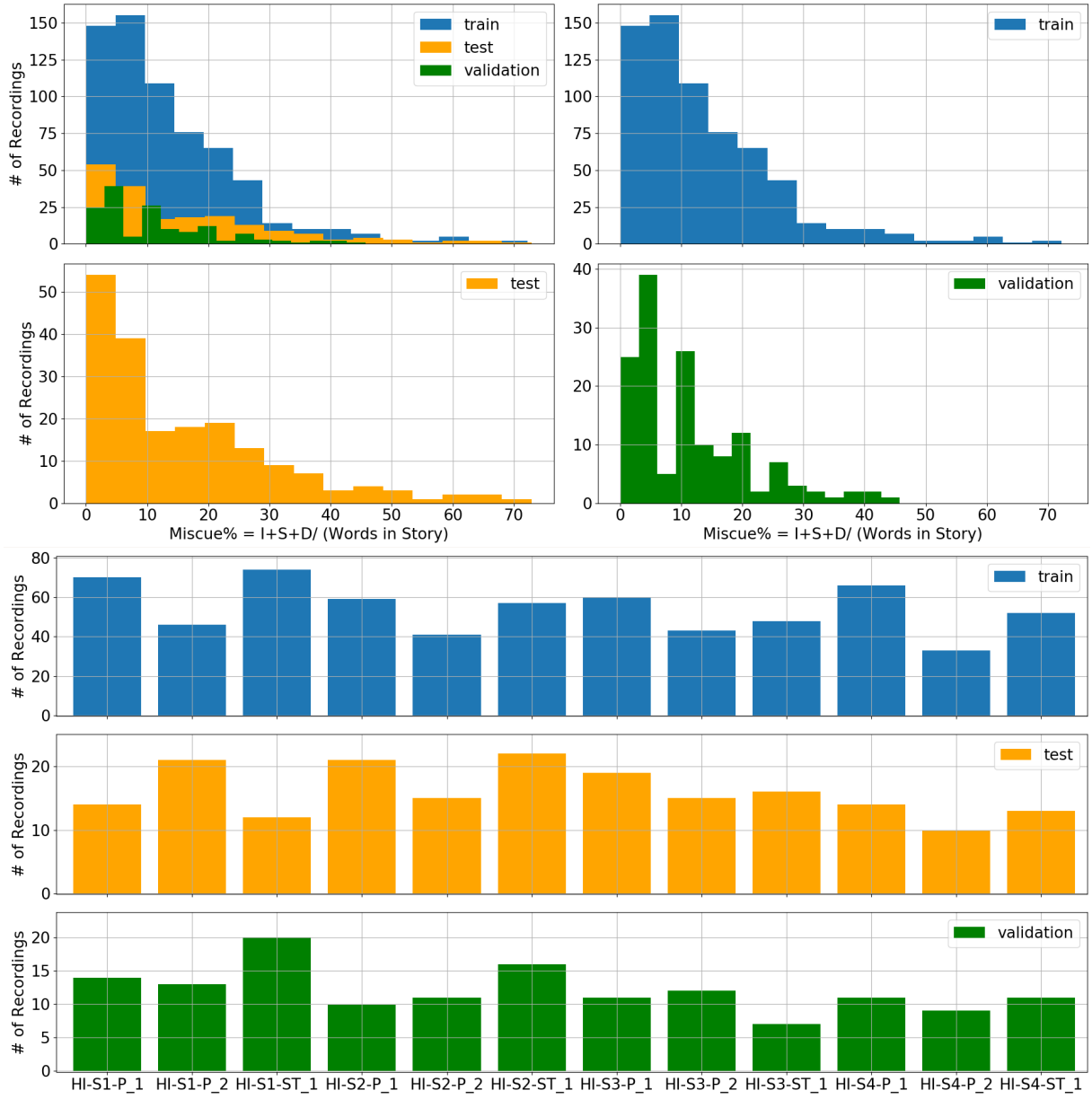

Figure 2.2: Distribution of UP recordings over miscue rate and Hindi story/paragraphs.

The following subsection goes into detail on the transcribing process in use and the other labels present in the transcription including the IR label described above.

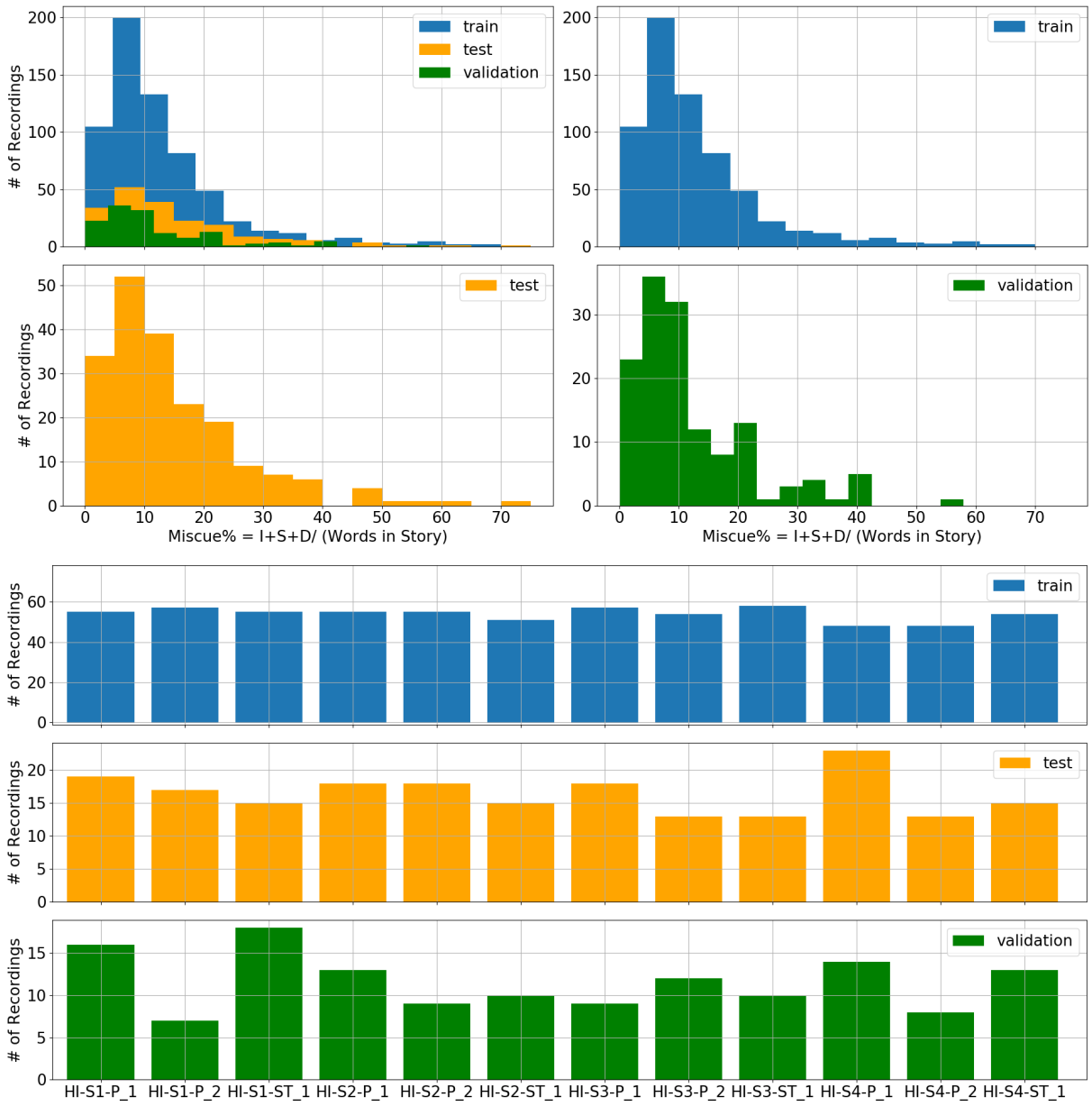

Figure 2.3: Distribution of RJ recordings over miscue rate and Hindi story/paragraphs.

Table 2.1: Summary of the ASER UP and RJ datasets

Dataset	# of Recordings Total(HPR, LPR)	# of Unique speakers	Duration (min)	# of non IR sentences	Non IR duration (min)
ASER UP train	649 (514,135)	489	302.62	3712	294.39
ASER UP test	192 (135,57)	164	102.07	1069	95.49
ASER UP valid	145 (120,25)	108	63.82	842	59.53
ASER RJ train	647 (539,108)	518	277.99	3696	266.42
ASER RJ test	197 (161,36)	159	81.65	1070	78.48
ASER RJ valid	139 (114,25)	115	62.45	817	60.24

2.2.1 Transcribing the ASER set

All audio recordings are first passed through a baseline DNN-HMM ASR [11] with a trigram language model trained on the specific story/paragraph text (along with a garbage model containing around 1500 common English words and all single phones) to get the decoded text output. Using the alignment of the decoded text with the canonical text, the number of miscues (insertions, deletions and substitutions) of the child are detected. This is then normalized by the number of words spoken to get the miscue rate. Using a threshold on the normalized miscues value, it is determined if a recording is Ratable (less than or equal to 20%) or Transcribable (greater than 20% and less than or equal to 80%) or 'WeakReader' (greater than 80%). Recordings having miscue rates greater than 80% could also have blank recordings where nothing is spoken.

This decoded text acts as a first level automated transcript for the recording to help the transcriber. This first level transcript is saved as an Audacity label track which also contains the timestamps aligned to the transcript. The first level transcript is slightly modified before passing it to the transcriber by removing isolated phones which get decoded due to the garbage model except for the following phones which are replaced by the tags mentioned below:

1. 1aa word in the decoded text(i.e. a single 'aa' phone) is replaced by FP (Filled pause).

2. 1s word in the decoded text (i.e. a single 's' phone) is replaced by BR (Breath Noise).

   The First level transcript label track and recording is split sentence wise based on its alignment with the canonical text. Then each line is aligned to the respective portion of its audio where the speaker started speaking the line to when the speaker stopped and is presented to the transcriber. Additionally, silences greater than 200ms are inserted as SIL labels as obtained from DNN-HMM CTM (time-marked conversation) outputs.

   The transcriber then modifies this first level transcript in the following ways in Audacity: The transcribers move the boundaries of sentences if needed. They modify the transcript using the Devanagari script in case an English word appears (because of the garbage model) or if they feel a slight modification to the decoded word is required or if they feel there is a mistake in the decoded text and a better transcrip-

tion is suited. The transcriber also adds in the following labels to the transcript by listening to the audio and modifying the label track:

3. Breath (inhalations/exhalations), sniffling or 's' sound between sentences are marked as BR.

4. Other noises (ON) if appearing in isolation are tagged as ON. Common noises observed include birds, mobile, vehicles, bell, mic noise, etc. When speakers clear their throat, it is also marked as an ON label.

5. Filled pauses like 'uh', 'hmm', 'umm' etc. are marked as FP.

6. For regions of unintelligible words or indecipherable mumbling, an MB symbol is used.

7. In case a section contains child whispering a word (this has been observed when the child has difficulties decoding the word), a WH label is used. Even if the words are audible, they are not transcribed and a WH label is used.

8. Irrelevant speech usually present at the start or end of the recording is transcribed by a separate label IR. This usually happens if the facilitator gives instructions to the child before the test has commenced or after it is over. These regions are distinct and the audible speech sounds made are irrelevant to the story being read.

   In the case where some labels overlap, the dominant source of a label is used as the only label.

The transcriber also ensures that the Audacity label tracks are contiguous and that additional SIL/BR/ON tags are used at the start/end of the sentence wherever necessary. Different disfluencies like Elongations, Stalling or Hesitations are marked as an additional (HS) tag in brackets along with the actual transcript. On finishing the transcription, the audio and label tracks are exported. Additional comments like School Noise, Loud background speaker, etc. may be entered in another field.

A stage of (Quality control/check) QC follows this transcription process ensuring the labels are marked accurately and the transcripts are satisfactory. These detailed levels of transcription and tags are treated as words whose pronunciation is the SIL (silence) phone from the IITM phone set while creating the train, test and valid datasets because the IITM baseline model is not trained on these. In evaluating the experimental results and student reading errors, these tags are removed from the ground truth transcription while

comparing with the decoded text. These tags are for future purposes and experiments where training separate phones for some of these tags will be quite useful for identifying and delivering specific feedback to the student speakers such as recognizing at what words hesitations (HS) commonly occurs or if the child whispers (WH) the word before pronouncing it with difficulty. It is also expected to help boost the ASR performance.

2.3 Campus School Hindi data

The Campus School (CS) Hindi data is a subset of a larger CS dataset which consists of children from the IIT Bombay campus school reading Hindi paragraph stories. They wear a headset while speaking and have the option to listen to a narrator reading the paragraph story presented to them on a tablet. The recordings are again sampled at 16Khz. The size of this dataset is very small compared to the other sets in this work (approximately 30 minutes in duration). These recordings have been transcribed from scratch unlike the procedure described in section 2.2.1 (because it was used to build the system in [11]) and the tags used were also not as extensive.

Some recordings and their transcriptions examples, involving the different tags and labels discussed, can be found in [12].

The CS Hindi datasets used and their statistics have been summarized in Table 2.2:

Table 2.2: Summary of the IITM and CS Hindi datasets

Dataset	# of Utterances	# of Unique speakers	Duration (min)
IITM train	27131	418	2400
CS Hindi	695	11	30

The phone sets of the ASER dataset and the CS Hindi dataset are different from the IITM data phone set. A manual mapping is created between the phone sets and the ASER and CS Hindi lexicons are converted to the IITM phone set. More information can be found in Appendix A. In the next chapter, a review of recent data augmentation methods is done and methods of augmenting and modifying the IITM dataset to suit the target set scenario of ASR is presented.

Chapter 3

Data augmentation

Data augmentation is a method of applying certain ecologically valid transforms on to the training data available. The transformed data can then be used along with the original training set to increase the count of the training data available or the original training set can remain unused with only the transformed data used for training. These methods are intended to groom the model towards certain test scenarios.

VTLP (Vocal tract length perturbation) was introduced by Jaitley et al. [13] where random warp factors chosen randomly from a normal distribution were used to 'corrupt' each utterance in the training data and it resulted in a slight reduction in the Phone Error Rate (PER) on the TIMIT task [14]. In [15] two different augmentation methods and their combined effects were considered on Assamese and Zulu recordings with only ~10hrs of training data available:
i) VTLP warping of the training data done three and seven times on Assamese and Zulu respectively using randomly sampled factors in the range [0.8, 1.2].
ii) Semi-supervised training (where a big pool of unsupervised data is decoded using an existing decoder and its output is used as the transcript for further training procedures). They observed a decrease in Token Error Rate (TER) in all augmentation cases, however combining the two methods did not yield the best result for Assamese, while it did for Zulu.

SpecAugment and SpecSwap are two recent methods of Data augmentation introduced by Park et al. [16] and Song et al. [17] respectively. Both of these methods involve spectrogram modifications and are tested using end-to-end ASR models which use Transformer networks and the Listen, Attend and Spell (LAS) networks respectively. In SpecAugment, three augmentation techniques inspired by computer vision are employed: time warping

(which deforms the spectrogram image along the time axis), time masking (where the entire spectrogram between certain time steps are made zero), frequency masking (where an entire band of frequencies are masked throughout the signal's length) [16]. In SpecSwap, inspired by the SpecAugment techniques and previous work by Song et al. [18] (where speech features were permuted), time swapping and frequency swapping techniques are examined. This involves two non-overlapping contiguous blocks of features, along time and frequency respectively, being swapped [17]. Both SpecAugment and SpecSwap when used along with the original data gave improvements over the baseline end-to-end models.

Speed perturbation (by re-sampling the signal), Tempo perturbation (corrupting the speech rate of the signal i.e. changing the tempo of the signal while keeping the pitch and the envelope of its amplitude spectrum constant) and VTLP as data augmentation techniques were compared by Ko et al. [19]. They obtained improvements using all techniques and concluded that the speed perturbation (which both slows down and speeds up the audio at 0.9x and 1.1x the original speed) when used along with the original train set was the best technique of the three methods.

In this work, the combined effects of speed perturbation and VTLP warping techniques of Data augmentation are examined. Since the aim of this work is to improve ASR systems for the target domain speech, the adult speech in the IITM data set is transformed in the following ways with the understanding that children tend to have higher formants than adults (which implies more energy concentration in higher frequency bands compared to adults) as shown in Huber et al. [20] and other works [21], [22]:

1) VTLP warp factors ($F_{original}/F_{warped}$) are chosen (from a uniform distribution for each utterance) such that their action maps frequency bins to only higher values. This mapping is a piece-wise linear map as shown in Figure 3.1.

2) The speed of each utterance is increased, at two levels, and used along with the original speed. This is done to both the VTLP warped and original IITM dataset. Intuitively, it is observed that sped up audios sound 'higher' than their normal counterparts. This is done because increase in the speed of a signal shifts both pitch and formants to higher values.

Having understood the reasons behind data augmentation and utilizing the above two augmentation strategies, these modified datasets are used in training acoustic models. The training procedure and architecture of TDNN acoustic models in kaldi are discussed in the next chapter.

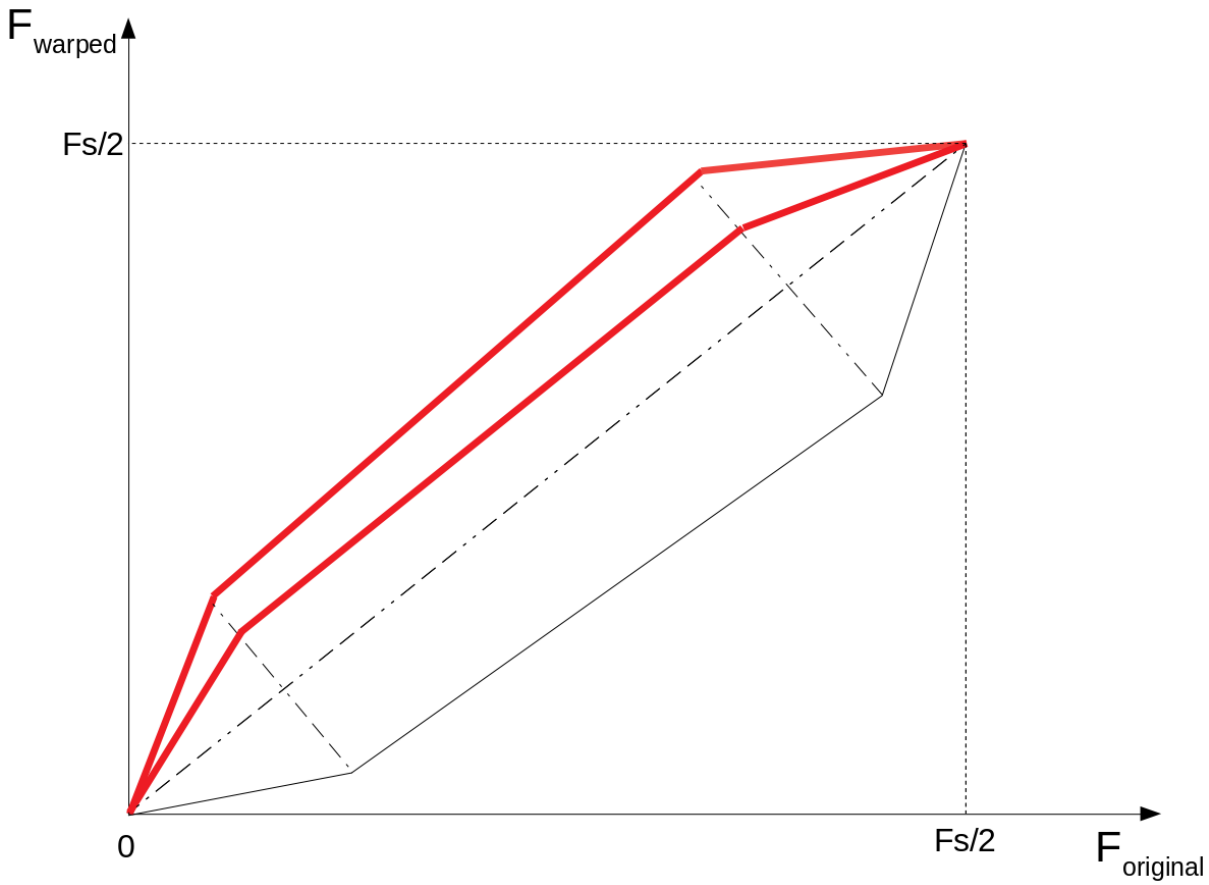

Figure 3.1: Representation of the VTLP warping function that maps frequencies to a new scale. In this work, the maps within the red perimeter are considered (corresponding to $0.8 <$ warp factors < 0.9).

Chapter 4

Kaldi TDNN chain models

Kaldi is a well known open source ASR toolkit introduced by Povey et al. [23] in 2011. Acoustic models in kaldi can be realized using Gaussian Mixture Models (GMMs) or other neural network based architectures which model the emission probabilities of a Hidden Markov Model (HMM). 'nnet3' models are a class of models that support the use of complex neural networks like Recurrent architectures (RNNs and LSTMs) and Time-delay Neural Networks (TDNNs) unlike the older 'nnet' setup which supported only simple deep feed-forward architectures. Chain models are a further subclass of these acoustic models which have innovative and unconventional HMM design and topologies. In this work, the baseline TDNN chain model recipe provided by the IITM ASR challenge organizers [10] is examined, which itself is a slight modification of the TDNN chain model recipe used in the WSJ (Wall Street Journal corpus) recipe provided by kaldi. Links to the recipes can be found here[1] [2]. First, a brief overview of the ASR problem is presented and then the TDNN chain model is described.

4.1 Overview of an ASR system

The broad goal of an automatic speech recognition system is to find a sequence of words that very closely matches the speech data present in an audio signal. It is defined as finding that sequence of words $\tilde{W}$, given a sequence of observation vectors O obtained from the audio signal, which maximizes the probability $P(W|O)$ where W is an arbitrary sequence of words. To put it mathematically, find $\tilde{W}$ such that

[1] WSJ recipe: https://git.io/JTH3n

[2] IITM baseline: https://git.io/JT9MZ

$$\tilde{W} = \arg\max_{W} P(W|O) \tag{4.1.1}$$

Using Bayes' rule and ignoring $P(O)$ since it is the same for all observation vectors this becomes

$$\tilde{W} = \arg\max_{W} P(O|W)P(W) \tag{4.1.2}$$

This product is now evaluated using two different modeling approaches. $P(O|W)$, the likelihood of observing O given that sequence of words spoken was W, is evaluated using an acoustic model. $P(W)$ (called a prior) models how likely the sequence W itself is irrespective of other information and is evaluated using a language model. Most modern day ASR systems use HMMs to do the Acoustic modeling excluding end to end models. A tutorial on HMMs and their role in training (learning the parameters of the HMM given a set of observation vectors and its transcripts) and decoding (as defined in Equation 4.1.1) for ASR applications can be found in [24]. Each HMM can be thought of as modeling a single phone or a context dependent phones (triphones).

The emission probabilities of an HMM can be modeled using a mixture of gaussians (GMM-HMM model) or a neural network. One conventional approach for building an ASR system is to first build a GMM-HMM system. Estimating the GMM's parameters is done simultaneously along with the HMM training using either the Baum-Welch algorithm or the faster Viterbi training. This GMM-HMM system act as a buffer for the neural network training by providing frame level phone labels for the neural network to train on. The objective function optimized in the neural network might be a frame level objective function (like the cross entropy between the HMM hidden state (triphone) and the observed feature vector for each frame).

This method of neural network training is an MLE (Maximum Likelihood Estimation) approach which maximizes the likelihood of the correct word sequence:

$$F_{MLE}(\theta) = \sum_{r=1}^{R} \log P_\theta \left(O_r \mid M_{w_r}\right) \tag{4.1.3}$$

where θ represents the parameters of the model, R is the total number of training utterances, w_r is an individual utterance's word transcript (which might be a sentence or two), M_{w_r} represents the hidden state sequence of the HMMs for that particular word sequence w_r. O_r represents the input feature sequence for that word sequence w_r.

Another well known approach which directly tries to maximize the posterior probability in Equation 4.1.1 is called the MMI (Maximizing Mutual Information) estimation

(which is part of other discriminative training approaches like MPE (Minimum Phone Error) and MWE (Minimum Word Error) [25].

The MMI objective function is:

$$F_{MMI}(\theta) = \sum_{r=1}^{R} \log \frac{P_\theta\left(O_r \mid M_{w_r}\right) P\left(w_r\right)}{\sum_{\hat{w}} P_\theta\left(O_r \mid M_{\hat{w}}\right) P(\hat{w})} \tag{4.1.4}$$

where the symbols have the same meaning as before in Equation 4.1.3 and $\hat{w}$ represents an arbitrary word transcript summed over all possible word transcripts. Unlike the MLE approach which only maximizes the likelihood of the correct word sequence from the training transcripts, MMI tries to maximize that likelihood while at the same time also minimizing the likelihood of all wrong word sequences.

In the next section the nnet3 TDNN chain model recipe is discussed which uses both the MMI training criterion and Cross entropy training.

4.2 TDNN chain model

The baseline TDNN chain model recipe from the IITM ASR challenge [10] is examined in this section. A GMM-HMM model provides lattices that align the training data with its transcripts which is required for the TDNN training. The TDNNs used in the baseline recipe use a 140D (dimensional) input vector for each frame (100D i-vectors and 40D LDA MFCCs) and have additional constraints on its layers. They are a factored form of TDNNs i.e. each matrix corresponding to a TDNN layer is factored into a product of two matrices, with one of the factors constrained to be semi-orthogonal [26]. The size of the factored matrices can also be adjusted using variables, called 'bottleneck-dim', 'big-dim' and 'small-dim', in the kaldi code depending on the type of layer.

The TDNN is trained using both the MLE estimation of cross entropy labels for each time frame and the MMI criterion between input feature sequence and output word sequence, discussed in section 4.1, on two different output layer blocks. This is conventionally referred to as multi-task training, although the cross entropy output (output-xent) block is unused while decoding. The output layer trained on cross entropy has a different learning rate, scaled to the MMI output learning rate, which can be set by the user. A snippet of this multi-output network's code can be seen in Figure 4.1, with another example in Figure 4.2 showing a final TDNN block which feeds two output layers.

In the following sections the TDNN architecture is discussed first, then the details of the chain model training (with its unconventional HMM topologies) which uses the

```
prefinal-layer name=prefinal-chain input=prefinal-l $prefinal_opts big-dim=1024 small-dim=192
output-layer name=output include-log-softmax=false dim=$num_targets $output_opts

prefinal-layer name=prefinal-xent input=prefinal-l $prefinal_opts big-dim=1024 small-dim=192
output-layer name=output-xent dim=$num_targets learning-rate-factor=$learning_rate_factor $output_opts
```

Figure 4.1: Kaldi code snippet of the two output layers used in chain models. The input to both of them is the same.

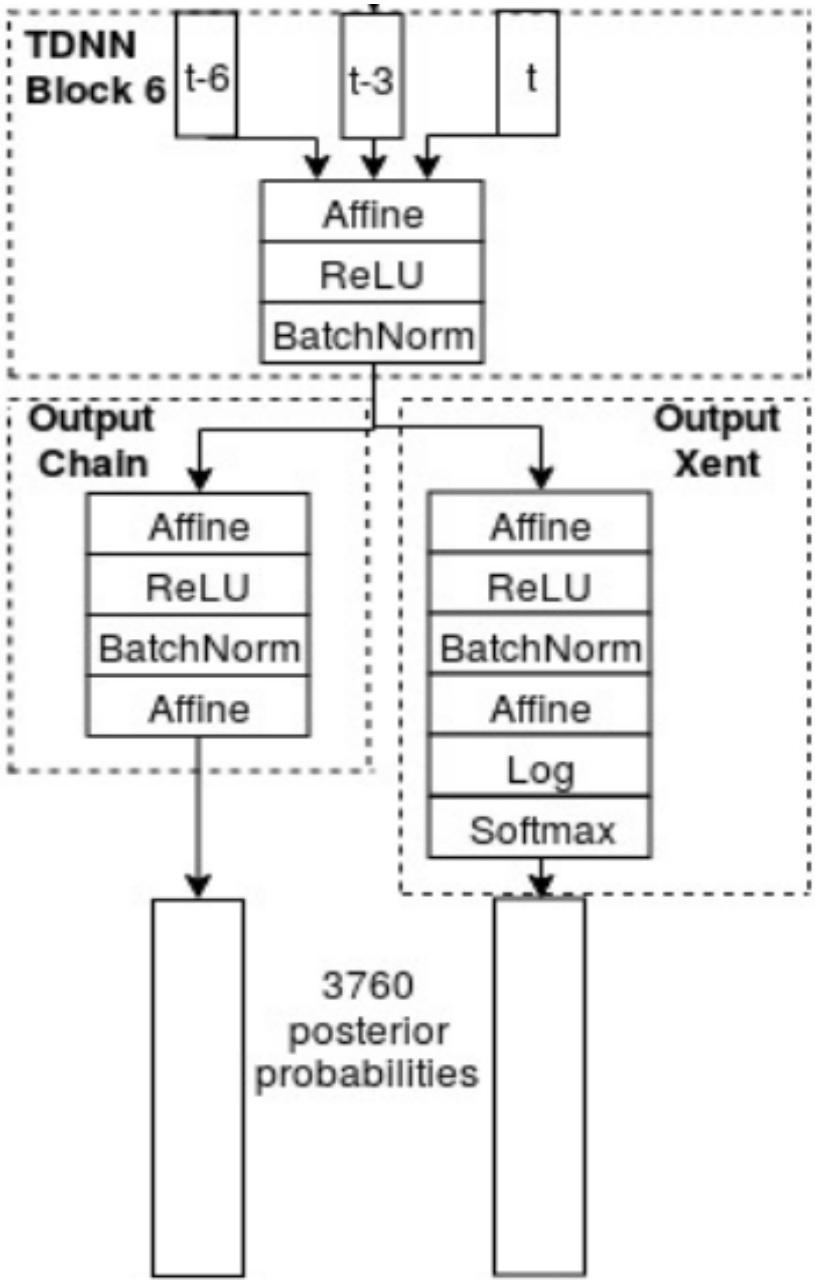

Figure 4.2: Final blocks of a TDNN architecture, representing the two output layers. [2]

TDNN outputs for training are discussed. Of course, TDNNs can be used to model emission probabilities of conventional HMMs as well.

4.2.1 TDNN

Time-delay neural networks were introduced to the Kaldi ecosystem in Peddinti et al. [3]. Figure 4.3 shows an example of a TDNN architecture. TDNNs are used to model long term contexts in the data using features calculated in the short term without explicitly modifying the features. The TDNN architecture in Kaldi has some other special properties and is similar to a 1D CNN (Convolutional neural network) in many ways. Each node (represented by the rectangular box) in a subsequent layer is computed using only the inputs within a given context from a previous layer. Activations at subsequent layers are not found at all time frames since there is a correlation between two adjacent nodes. So a sub-sampling of the nodes at each layer is done. The lines and boxes in red represent the sub-sampling, characteristic of the network. The blue lines represent a conventional TDNN. This sub-sampling leads to a huge reduction in training time compared to conventional TDNNs.

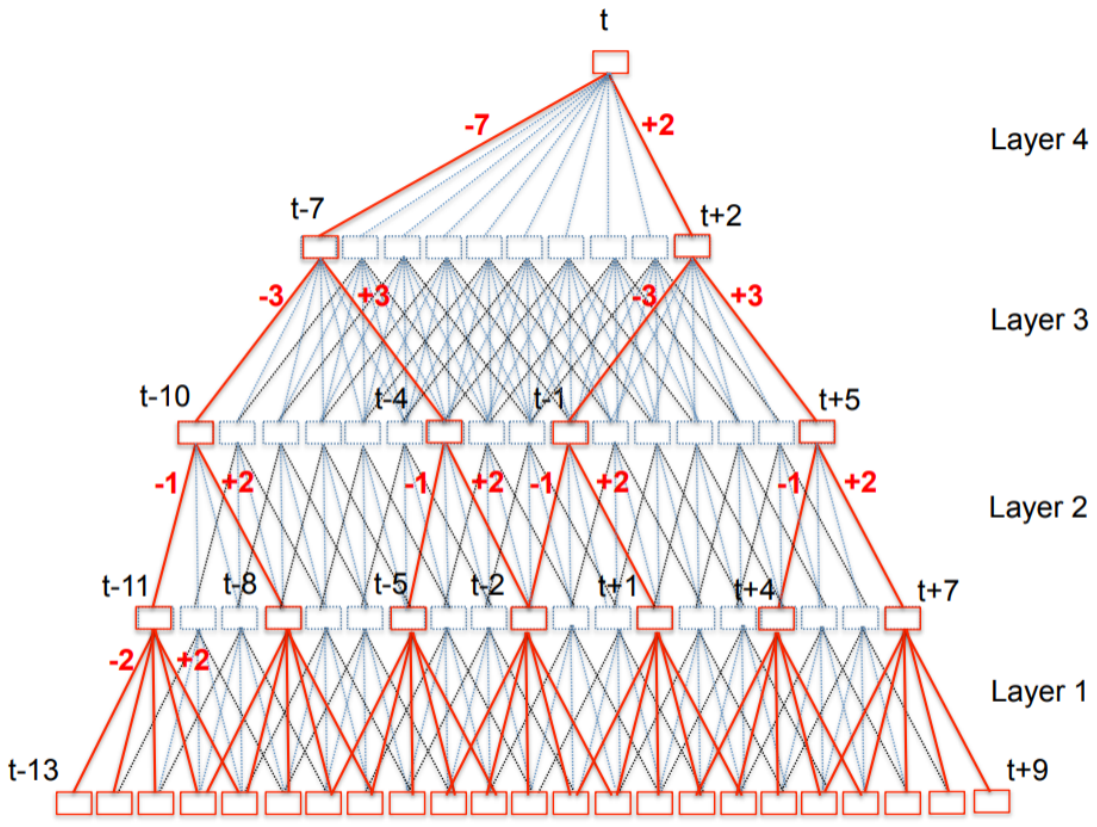

Figure 4.3: Example of a TDNN architecture [3]

Furthermore, as the network gets deeper it was found that using wider and wider contexts was beneficial. This can be seen with the increased context range in Layer 3 compared to Layer 2. The context range was explored as a hyper-parameter in [3] and

it was found that after increasing the contexts beyond a certain range, the WERs were adversely affected. All these observations seem to be in agreement with certain intuitive notions held about speech and phones. Also, it can be seen that the left context (past) is usually larger than the right context (future) which doesn't exceed three frames in the baseline model. This was done to support real time decoding used in other recipes. The baseline TDNN model used in this work has 12 TDNN layers followed by one linear layer (along with two linear output layers and one input layer) and the contexts used are: one frame for the first three layers, no context on the fourth layer and three frames of context for the remaining layers.

With the TDNN architecture section, the next section describes how a TDNN is trained in kaldi using the MMI and MLE criteria. This is referred to as chain training.

4.2.2 Chain training

Training the TDNNs involves the use of a previously trained conventional GMM-HMM model that provides the alignments of the TDNN training data with its transcripts. The GMM-HMM model uses a LDA + MLLT + SAT tri-phone GMM model (with 2750 tri-phones and approximately 50 gaussians per tri-phone) trained on the IITM data set. More information about GMM-HMM model training can be found in [11]. The input to the TDNN is a 140D vector (concatenating the 100D i-vector and the 40D LDA MFCCs). Considering Equation 4.1.4, two terms i.e. the numerator and denominator terms have to be evaluated. They are treated as graphs during training.

The denominator term is a constant for all utterances and need only be evaluated once and composed with every new training utterance. It is evaluated on a graphics processing units (GPU) as a phone graph, which is very similar to the construction of decoding graphs, using a 4gram phone LM (with no back-off) obtained from all training transcripts. There is no lexicon FST (L) here so the denominator is an HCP FST (Finite state transducer) containing only the HMM states (H), the context dependency (C) and the phone LM (P). More information about the determinization, minimization and other phone LM characteristics can be found in [27]. Even with these adjustments of using a phone graph the model size becomes prohibitively large, so another sub-sampling approach is taken up where the TDNN outputs are computed every 30 milliseconds (ms) and then passed to be composed with the denominator FSTs, which is then evaluated to find the denominator in the loss function.

However, because the TDNN outputs are at 30 ms hops, the older HMM topolgies of three states per phone cannot be used. So a new topology where an HMM can be traversed in one transition between two states is used. The first state outputs the context dependent tri-phone and the second state emits a blank symbol. This was inspired by the work of Sak et al. [28]. It is expected that the emission probabilities and the MMI objective function make up for this crude topology. No lattices are constructed in the denominator and instead the 4gram phone LM models all possible utterance transcripts, hence the name lattice-free MMI.

Next, the numerator term is also evaluated as an FST. The numerator term gives the probability of the utterances' feature given its correct transcript. It is evaluated as an FST using the GMM-HMM alignments of the training data with its transcripts converted into lattices (containing alternate phone pronunciations for each word as well). Even for a single pronunciation, the lattice alignments used (instead of a single best forced alignment) while training offer extra variability, a simple example can be an alignment where there are only three feature frames present in an utterance and if the corresponding word W of that utterance has a phone transcript x-y, then possible alignments are x,x,y or x,y,y [29]. These lattice alignments are relaxed a bit; allowing a phone to occur on either side of its alignment by adding 50ms of slack. It gives the chain model more 'freedom' while training, as expressed in the kaldi code comments. However since the G FST (Language model) is just the words in the transcript for that utterance it is just composed as an acceptor. The HCLG fst is evaluated on the central processing unit (CPU) [30].

Another optimization used to reduce the size of the memory occupied while evaluating both FSTs is to split the training utterances into chunks of 1400ms i.e. (140 frames). These chunks are obtained by splitting the utterances and also using the lattice alignment information to modify the phone transcripts and get exact 1.4 second chunks for the numerator FST. The same splits are used for the denominator FSTs. More information and optimizations involved in the FSTs of the split chunks can be found in [30], [29]. Codes that create the denominator FST[3] and numerator FST[4] with explanations in the comments can be found in the links.

With the numerator and denominator FSTs computed, their difference in log domain gives the MMI loss function. This loss is then minimized by the TDNN training. No early stopping criteria is provided. Instead, two subsets (300 utterances each) of the

[3]denominator.fst: https://git.io/JT78E

[4]numerator.fst: https://git.io/JT74f

TDNN training data are formed called i) validation diagnostics (which has no utterance overlap with the actual data used to train the TDNN) and ii) train diagnostics (which has utterance overlap). The loss functions on each of these subsets at each training iteration can be used to determine whether under-fitting or over-fitting occurred.

Once the TDNN training is done an HCLG FST graph is available and the TDNN is available as a final.mdl file. Usual kaldi decoding procedures can then be followed with some slight modifications to the acoustic weight parameter [31]. Reducing beam-widths while creating decoding lattices can significantly reduce decoding time with little to no degradation in WER. The TDNN outputs during decoding are also computed at 30ms hops and provide this improved decoding speed.

With the IITM adult speech trained TDNN chain model in hand, the next focus is on adapting these models to target domain speech. Works on Transfer learning and Adaptation are reviewed next.

Chapter 5

Transfer learning and Adaptation

The aim of this work is to explore the baseline TDNN model (trained on adult speech) and to improve its decoding ability on target speech. To this end, techniques of adapting and modifying the baseline model for the ASER data sets are explored.

5.1 Acoustic model adaptation

In this section, acoustic model adaptations explored in [11] are reviewed. Batch supervised adaptation techniques are considered since the requirements of this work right now do not need real time outputs (so online adaptations for each new utterance is not done) and the transcriptions obtained for the speech are manually done by a transcriber. There are broadly three adaptation methods:

1. **Feature Adaptation**

   In feature adaptation, the input features used in the ASR system are modified to either remove or control speaker effects. Common methods include Vocal Tract Length Normalization (VTLN) which tries to curtail the effects of the different vocal tract length the speakers have [32]. VTLP warping of the feature set can also be considered a form of adaptation although it is unguided and not optimal like VTLN techniques. fMLLR (feature maximum likelihood linear regression) methods find affine transforms of the features to remove variability and normalize across speakers. More information on applying feature transforms in kaldi can be found in [33]. Identity vector (i-vector) features are commonly used in neural network systems to capture information about the speaker identity and channel acoustics for speaker verification tasks but have also found use in speech recognition systems

[34], although in recent ASR systems they are an essential feature that are almost always used.

2. **Model space adaptation**

   Model space adaptations as the name suggests modify the model parameters instead of the features to suit the adaptation data and its speakers. Maximum a posteriori (MAP) adaptation is a well known adaptation for GMM-HMM models. It maximizes the likelihood of the adaptation data by modifying the parameters of the GMM-HMM model [35]. Maximum likelihood linear regression (MLLR) of GMM parameters is similar to fMLLR but the affine transform operates on the model parameters (instead of the features) to maximize likelihood adaptation data. These transforms can be found for specific pronunciation variations and can be applied for specific gaussian components (regression classes) as done by Yoo Rhee Oh et al. [36].

   For neural network models re-training specific layers of the network with the adaptation data gives improvements. Transfer learning is an extension of this and is explored deeply in Section 5.2.

3. **Speaker space adaptation**

   In speaker space adaptations, separate acoustic models are trained for each speaker type. Speaker types are estimated using clustering techniques or is done manually (for example building gender/age dependent models). Once the speaker clustering is done and different sets of parameters for each model are available, new speakers are estimated as weighted means of the cluster models and decoding is done. A generalization of this is to find the eigenspace of the speakers which reduces model size and complexity [37].

5.2 Transfer Learning

Transfer learning is a model-space adaptation method as discussed in Section 5.1. This is the main focus of experimentation in this work. Transfer learning methods in kaldi were investigated extensively by Ghahremani et al. [38] which will be the reviewed in this section. Other kaldi transfer learning works like Manohar et al. [39] mainly refer to this.

Transfer learning methods are used when there is a small amount of domain specific

data (target speech) compared to larger amounts of out of domain data. It involves using the domain specific data to improve a model trained on the out of domain data which will be called the 'source' model. This is achieved by using the source model and either modify its training mechanisms or re-train the model using the domain specific data to obtain a 'target' model that performs better on test sets from the domain specific data. The alignments of the training/valid data with its transcript, required for weight transfer, are obtained from the source TDNN chain model as well.

Lee et al. [40] showed that the intermediate layers of a network are not task specific by using a pre-trained model using some of the bottom layers and fine tuning them for a variety of audio classification tasks including speaker identification, phone classification, speaker gender classification and separately for music genre and music artist classification. This gives credence to the idea that transfer learning can be used extensively when the domain specific data is in shortage. In many cases of multi-lingual DNNs it was found that language similarity (between source and target) and the amount of data also played a part [41] [42]. A general block diagram of transfer learning is shown in Figure 5.1.

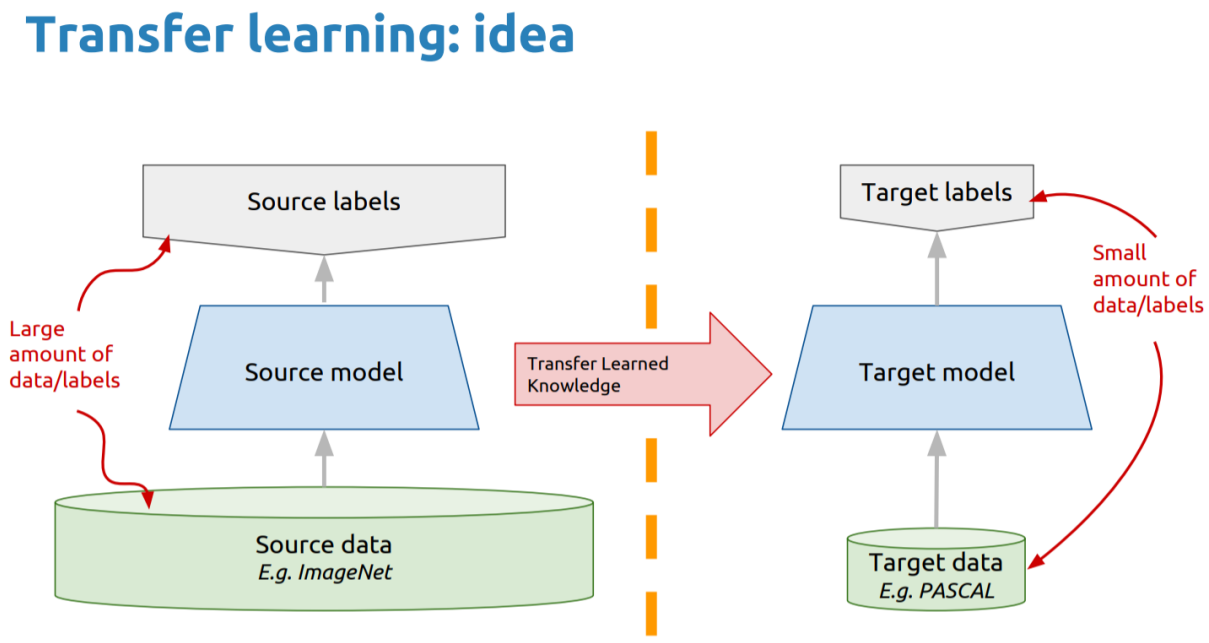

Figure 5.1: A general block diagram of transfer learning [4]

In Ghahremani et al. [38], two methods of transfer learning are investigated:

1. **Joint multi-task training:**

   In this method, the initial layers of the network are trained on data from all domains in parallel, and specific final layers (all of which share the output of the last initial

layer) are trained only on their respective domain specific data. Scaling the gradients from the domain specific data depending on their size has been found to help.

2. **Weight transfer:**

   Weight transfer closely mirrors the work of Lee et al. [40]. A source model trained on out of domain data is taken and the first 'x' number of source layers are retained. Then either new task specific layers are added as required and the model is re-trained using the domain specific data. This re-training also allows different strategies. In single stage tuning the new task specific layers are trained at higher learning rates compared to the source layers, whereas in two-stage training the source layers are frozen and the task specific layers are trained on the domain specific data followed by fine tuning both source and task specific layers using a small learning rate for a very small number of epochs.

From these experiments, Ghahremani et al. [38] concluded that:

1. Joint multi-task training is preferable to weight transfer, although both outperform the baseline and the multi-tasking takes more time to train.

2. Single-stage tuning is preferable to two stage tuning in weight transfer experiments

3. When the phone sets, type of speech (spontaneous or read speech) of the domain specific and out of domain data are the same, transferring all layers from the source model was more effective than some other partial transfer that does not include all layers.

During single-stage tuning, where the transferred source layers are trained at smaller learning rates compared to the task specific layers, it was found that increasing the ratio between the learning rates of the source and global learning rate (α) helped reduce the WER on a test set up to a degree.

In Zhang et al. [43], an alternative school of thought in transfer learning was presented which freezes task specific layers and trains lower layers. The output layer and top most one, two or three BLSTM (Bidirectional Long Short-Term Memory) layers of an end to end CNN-BLSTM model (with four BLSTM layers on top of the CNN) were frozen and the lower layers are retrained using the adapted data. This is justified with the premise that although the intermediate layers of a network aren't task specific (they capture a useful representation of the input), the final layers are task specific, so retraining the

intermediate layers while keeping the final layers frozen modifies the representation learnt from the target domain data. The best result they obtained was for freezing the top two BLSTM layers and the output layers. They also examine scaling the learning rates of the topmost layers by two factors of 0.1 and 0.5 with the global learning rate. The best results were obtained for the factor 0.5.

5.3 Parameters in transfer learning

Some of the commonly tuned parameters in transfer learning are explained here:

1. **Regularization:**

   In kaldi, it was empirically found (through experiments on various corpora) that a form of multi-task training i.e. having two or more output layers trained independently on different objective functions (cross entropy and MMI) helped decrease the WER. These output layers are called 'output-xent' (Short for cross(x) entropy(ent)) and 'output' (chain). The 'output' layer is the one that is finally used while decoding while the 'output-xent' is only useful while training. The 'output-xent' layer also acts as a form of regularization [27]. Another regularization method used is the L2 norm regularization. The squared L2 norm of the 'output' layer of the TDNN (not 'output-xent') is penalized. This involves adding $-0.5c * ||y||_2$ to the loss function, where y is the output of any node in the 'output' layer. The variable 'c' is tuned and determines the degree of regularization. [27]. A separate term that scales the output of the 'output-xent' layer also exists called chain.xent-regularize. These two parameters can help tune the degree of regularization.

2. **Number of epochs:**

   The number of epochs help determine the total number of training iterations. The dataset is divided into mini-batches before passing it through the network. Each mini-batch consists of 'x' utterances where x is called the batch size. These two parameters together determine the number of iterations. These mini-batches are constructed using the 1.4 second kaldi training chunks and treating them as utterances as explained in Chapter 4.2.2.

3. **Number of jobs:**

This determines whether multiple GPUs are used which can do simultaneous training on different subsets of the training data. Although no significant differences are expected, due to the size of the dataset using one GPU is optimal and recommended to reduce the effects of model averaging (combining the training/model parameters across GPUs).

4. **Global initial and final effective learning rates:**

   These rates determine the learning rate of the network at each iteration. The learning rate starts at the initial learning rate on the first iteration and decreases at each iteration until it reaches the final learning rate. Together, the rates control the rate at which the training loss reduces.

5. **Differential learning rates:**

   Differential learning rates are used to train the lower layers of the baseline model and the two output layers. The learning rate of the lower layers and the two output layer are represented using the convention 0.0005(13) - 0.005(2) which indicates the initial learning rate of all 13 lower layers = 0.0005 while the initial learning rate of 2 output layers = 0.005.

. The other parameter not fiddled with is the mini-batch size at 64, which is kept fixed across all experiments. In the next section, the experiments carried out based on the understandings from these parameters are explained.

Chapter 6

Experiments

The following observations are made, pertaining to the children's ASER test data, for the adaptation and data augmentation experiments in this work:

1. The VTLP warped and original IITM dataset are considered for training. Along with this the speed perturbation factors are also changed from the default baseline used in kaldi (0.9x and 1.1x) to higher speeds which also shift the data to higher frequencies (1.1x and 1.15x, 1.1x and 1.2x) as discussed in Chapter 3.

2. For the weight transfer experiments, the UP and RJ sets in Table 2.1 are used for re-training, validation and testing. The training and validation sets are also augmented by adding the speed perturbed versions of these sets at 0.9x and 1.1x the original speed of the audio. Since the size of these sets are relatively small compared to the original train set and there is no language mismatch, no new layers are added and all source layers are transferred. The final two output layers of the source model are treated as task specific layers and are retrained at different learning rates. The best model (based on valid set WER) obtained from experiment 1 is used in the weight transfer experiment.

6.1 Evaluation metrics and Decoding parameters

This section outlines the evaluation metrics to measure the ASR system performance keeping in mind that it is inherently measuring (or rather should measure) the reading skills of the child. A modified version of the standard Word Error Rate (WER) is presented that is based on observations made from the recordings and type of errors from the ASR output.

While decoding, two parameters are involved in determining the decoded text. These parameters influence the cost associated with each word. The decoding cost, consisting of an acoustic cost, language model cost and an insertion penalty is:

$$Total\_decoding\_cost = \log P(O|W) + LMWT * \log(P(W)) + WIP * |W| \quad (6.1.1)$$

$|W|$ represents the total number of words in an utterance. LMWT represents the language model weight and WIP represents word insertion penalty. In all experiments LMWT is an integer ranging from 10 to 50 and the WIP values used are -0.5,0.0,0.5,1.0.

6.1.1 Evaluation metrics

The experiments made are evaluated using WER metric and the modified version called mod_WER. To calculate the WER on the test sets, a tri-gram language model (LM) which is trained on the canonical text of all the ASER Hindi stories is used. In addition to this, a uni-gram garbage model with a fixed cost = 0.2 (tuned from other experiments) is appended to the language model to provide words from outside the stories' text. The garbage model is region specific and contains all unique words from the transcriptions of the recordings of that region (UP, RJ) that occur at least twice in the train and validation set. It also contains all single phones present in the lexicon. This slightly restrictive LM is expected to provide a good performance for speakers who are good readers (because of the restrictive text), while the garbage model allows other words to come in. However, the performance of the LPR set might diminish slightly because the garbage model weights are the same for both the HPR and LPR set. As stated before in Section 2.2.1, the other tags like ON, FP, BR etc are removed from the ground truth while evaluating these metrics.

Based on some observations of the common mistakes children in the ASER set make, a slight modification to the WER, called the mod_WER, is made. It treats all contiguous insertions as one insertion and is merged to one insertion. After this, if there is a sub-

stitution on either side of an insertion it is treated as one substitution. For example, if the sequence of mistakes when comparing a decoded text to the ground truth transcript was say DCIICSIIICC, where C is a correct word, I is an Insertion, D a deletion and S a Substitution the mod_WER would treat this as DCICSCC. These changes were made by observing that children hesitate and stutter before saying a word correctly(C) or saying something else entirely(S). To get a more accurate picture of their reading performance contiguous insertions are not penalized so.

6.1.2 Selection of decoding parameters

The WER of the test sets on all experiments is reported using the best performing LMWT and WIP found on the validation set. The decoded texts corresponding to these parameters are also used to report the mod_WER on the test sets. These are the metrics on which improvement is desired.

The next chapter presents the results of these experiments using the WER and the mod_WER metric defined above.

Chapter 7

Results

For the Data augmentation experiments, minor inconsistent improvements in the WER and mod_WER were observed. The overall WER on the both the validation and test set along with the mod_WER for the UP and RJ test sets are presented in Tables 7.1 and 7.2 respectively. It shows minor improvements on the valid set while the improvements on the test are inconsistent for the data augmentation based experiments.

Table 7.1: Data augmentation experiment results on ASER UP dataset

Model	WER% valid	WER% test	mod_WER% test	LMWT	WIP
Baseline (original IITM data) — 10 epochs — (sp 0.9,1.1)	16.34	29.97	19.75	34	0.5
VTLP warped + original — 20 epochs — (sp 0.9,1.1)	16.42	29.32	19.32	28	0.0
VTLP warped + original — 20 epochs — (sp 1.1,1.15)	16.13	29.56	19.45	31	0.0
VTLP warped + original — 20 epochs — (sp 1.1,1.2)	15.97	30.16	19.62	37	0.0

The best performing tuned weight transfer parameters for the UP and RJ set obtained by observing the validation loss in the single stage tuning experiments are in Table 7.3. For the single stage weight transfer experiments, a plot of the MMI loss functions for different learning rate values (the differential learning rates in weight transfer parameters discussed in Chapter 6) of the parameter combination from Table 7.3 on the UP valid set is shown in Figure 7.1. The results of single stage tuning experiments with the UP and RJ valid sets are described in Table 7.4 and Table 7.6 respectively. The LMWT and WIP

Table 7.2: Data augmentation experiment results on ASER RJ dataset

Model	WER% valid	WER% test	mod_WER% test	LMWT	WIP
Baseline (original IITM data) — 10 epochs — (sp 0.9,1.1)	18.06	19.94	14.79	35	0.0
VTLP warped + original — 20 epochs — (sp 0.9,1.1)	18.41	19.49	14.79	36	0.0
VTLP warped + original — 20 epochs — (sp 1.1,1.15)	18.04	19.99	15.41	33	0.0
VTLP warped + original — 20 epochs — (sp 1.1,1.2)	17.46	19.57	14.63	30	0.0

Table 7.3: Best performing retraining parameter combinations

Parameter	UP	RJ
–chain.xent-regularize	0.0001	0.00001
–chain.l2-regularize	0.0	0.0
–trainer.num-epochs	4	4
–trainer.optimization.initial-effective-lrate (initial global learning rate)	0.0005	0.0005
–trainer.optimization.final-effective-lrate (final global learning rate)	0.00005	0.00005

found for each experiment is used to report the results on the test set. The results are discussed in the next section.

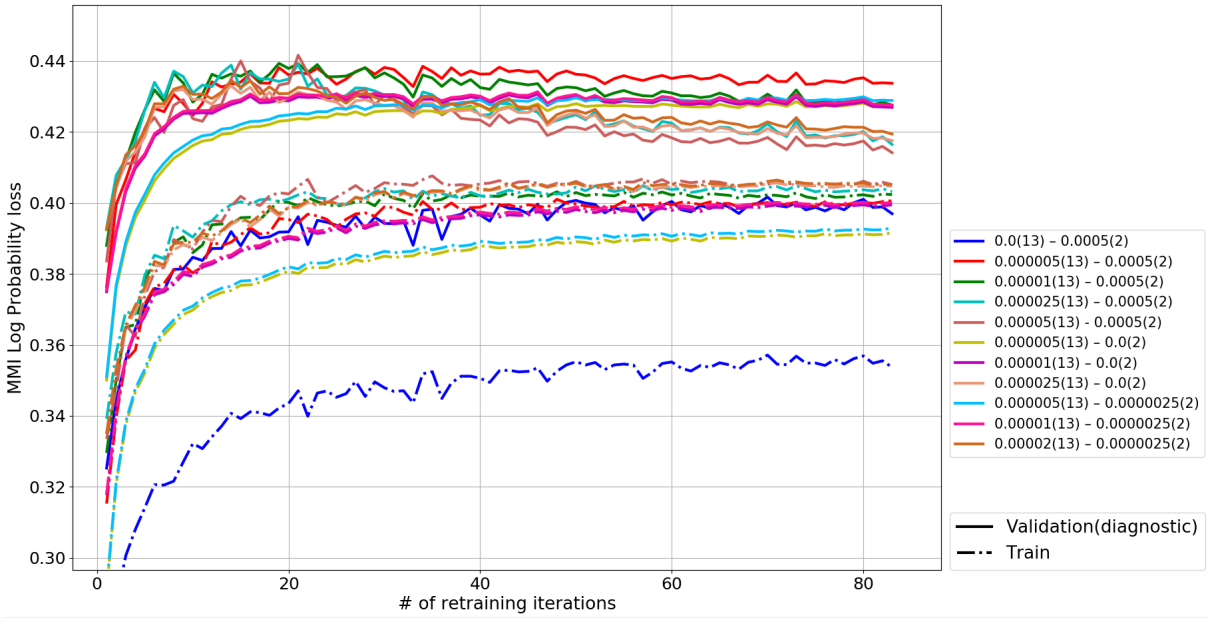

Figure 7.1: MMI Log probability loss of train and valid sets for various differential learning rates.

Table 7.4: Weight transfer results on UP valid dataset

Model	HPR WER%	HPR LMWT, WIP	LPR WER%	LPR LMWT, WIP
Baseline	12.20	39, 0.0	30.80	24, 1.0
0.0(13) – 0.0005(2)	13.19	43, -0.5	31.00	25, 1.0
0.000005(13) – 0.0005(2)	10.83	39, -0.5	25.86	25, 0.5
0.00001(13) – 0.0005(2)	11.05	33, -0.5	26.26	30, 0.5
0.000025(13) – 0.0005(2)	10.91	27, -0.5	27.44	25, 1.0
0.00005(13) - 0.0005(2)	11.17	39, -0.5	27.15	25, 1.0
0.000005(13) – 0.0(2)	10.10	25, -0.5	25.27	33, 0.0
0.00001(13) – 0.0(2)	10.25	23, 0.0	25.37	29, 0.5
0.000025(13) – 0.0(2)	10.54	24, 0.0	26.46	21, 1.0
0.000005(13) – 0.0000025(2)	10.10	31, -0.5	24.88	33, 0.0
0.00001(13) – 0.0000025(2)	10.35	23, 0.0	25.77	25, 0.5
0.000025(13) – 0.0000025(2)	10.49	38, -0.5	26.16	20,1.0

Table 7.5: Weight transfer results on UP test dataset

Model	HPR WER%	LPR WER%	HPR mod_WER%	LPR mod_WER%
Baseline	9.79	48.54	8.31	35.22
0.0(13) – 0.0005(2)	9.61	46.75	8.36	37.84
0.000005(13) – 0.0005(2)	8.26	34.89	7.81	32.61
0.00001(13) – 0.0005(2)	8.29	35.19	7.96	32.99
0.000025(13) – 0.0005(2)	7.96	36.01	7.69	33.25
0.00005(13) - 0.0005(2)	8.31	35.93	7.76	32.95
0.000005(13) – 0.0(2)	7.59	34.81	7.16	32.05
0.00001(13) – 0.0(2)	8.01	35.56	7.41	32.99
0.000025(13) – 0.0(2)	7.89	36.23	7.41	33.13
0.000005(13) – 0.0000025(2)	7.39	34.78	7.01	32.05
0.00001(13) – 0.0000025(2)	8.11	35.34	7.51	32.69
0.000025(13) – 0.0000025(2)	8.01	35.93	7.61	32.87

Table 7.6: Weight transfer results on RJ valid dataset

Model	HPR WER%	HPR LMWT, WIP	LPR WER%	LPR LMWT, WIP
Baseline	12.25	37, 0.0	43.63	29, 0.5
0.0(13) – 0.0005(2)	14.30	48, -0.5	40.77	25, 1.0
0.000005(13) – 0.0005(2)	10.44	49, -0.5	33.30	32, 0.0
0.00001(13) – 0.0005(2)	10.36	40, -0.5	33.74	42, -0.5
0.000025(13) – 0.0005(2)	10.96	35, -0.5	33.30	43, 0.0
0.00005(13) - 0.0005(2)	10.93	36, 0.0	34.40	42, 0.0
0.000005(13) – 0.0(2)	10.01	28, -0.5	31.10	28, 0.0
0.00001(13) – 0.0(2)	10.04	27, -0.5	32.31	31, -0.5
0.000025(13) – 0.0(2)	10.34	32, -0.5	33.85	28, 0.0
0.000005(13) – 0.0000025(2)	10.01	46, -0.5	31.32	26, 0.0
0.00001(13) – 0.0000025(2)	10.04	30, -0.5	32.20	28, -0.5
0.000025(13) – 0.0000025(2)	10.39	33, 0.0	34.40	26, 0.5

Table 7.7: Weight transfer results on RJ test dataset

Model	HPR WER%	LPR WER%	HPR mod_WER%	LPR mod_WER%
Baseline	12.80	45.68	11.38	35.53
0.0(13) – 0.0005(2)	14.80	46.15	12.63	37.99
0.000005(13) – 0.0005(2)	10.83	37.51	10.26	33.94
0.00001(13) – 0.0005(2)	11.50	39.18	10.75	34.02
0.000025(13) – 0.0005(2)	11.34	38.94	10.73	34.73
0.00005(13) - 0.0005(2)	12.68	38.22	12.13	34.89
0.000005(13) – 0.0(2)	10.55	36.48	9.96	32.51
0.00001(13) – 0.0(2)	10.87	39.02	10.28	33.86
0.000025(13) – 0.0(2)	11.05	38.14	10.53	33.47
0.000005(13) – 0.0000025(2)	9.98	36.24	9.47	32.20
0.00001(13) – 0.0000025(2)	10.79	38.86	10.16	33.54
0.000025(13) – 0.0000025(2)	11.44	37.59	10.93	33.70

7.1 Discussion of results

In both the ASER UP and RJ test sets, the WER% of the LPR set is much higher than that of the HPR set. Improvements are obtained from the weight transfer experiment for both subsets. However, the performance of the system on the LPR recordings is still relatively poor. Examining this test set, they are found to have much higher concentration of other tags including other noise (ON), described in Section 2.2.1. These noises lead to some amount of insertions and substitutions. Another main point that explains the higher WER of the LPR set is that it also has words which are not present in the Language model(LM) or the Garbage model(GM). Many of the substitutions and insertions observed were from words that are absent in both the LM and GM. Decoding the LPR and HPR sets separately does help in reducing the WER results as the LMWTs and WIPs are more tuned to the nature of these test sets. These observations and understandings help explain the relatively poor WER and mod_WER obtained on the LPR set for the experiments done in this work.

The best performing model overall for the UP test set is the (VTLP warped + original) + (speed perturbed at 1.2x and 1.1x original speed) (IITM data) trained model retrained using '0.000005(13) – 0.0000025(2)' which gives 7.01% mod_WER, an improvement over baseline mod_WER of 8.31% for the HPR subset. The best model overall for the RJ test set is also the (original) + (speed perturbed at 0.9x and 1.1x original speed) (IITM data) trained model retrained using '0.000005(13) – 0.0000025(2)' which gives 9.47% mod_WER, an improvement over baseline mod_WER of 11.38%.

Chapter 8

Conclusion

With the goal of building an automatic assessment system for children taking literacy tests a few challenges arose. These included the vast variety in speaking style and speech characteristics of children, the scarcity of labeled data available. Since, there are a large number of adult speech datasets and ASR systems available, modifications to these systems and datasets were considered a viable opportunity. A state of the art TDNN ASR system was discussed and two strategies for improving them were considered. Data augmentation techniques were examined to increase the quantity of training data and also modify it to suit particular test scenarios for the target speech. Furthermore, with a small amount of labelled target speech data, weight transfer techniques were examined to fine tune the system for the target domain speech. Both these techniques gave improvements in detecting reading miscues on the test sets and improved WER by 2.2, 2.8% absolute on the HPR UP and RJ test subsets respectively.

8.1 Future Work

- The first pain point discovered in the test sets is the presence of many unknown substitutions which are not present in the LM or garbage model. As more and more transcriptions of the ASER dataset become available, the garbage model used will expand which should also help reduce the number of substitution errors the occur in the ASR output, especially in the LPR set which have a lot of substitution errors.

- Further experiments, on English and Marathi datasets can be conducted using the other transfer learning experiments which involve transferring only part of the source model (not all layers). Freezing more and more of the TDNN layers and treating

them as task specific layers along with the output layers is another avenue to explore.

- Other data augmentation techniques like SpecSwap and SpecAugment can be used to increase the amount of target adaptation data available for further improvements in the transfer learning experiments.

References

[1] ASER Centre. ASER Survey Process. http://www.asercentre.org/p/231.html, last accessed: October 2020.

[2] Alexandru-Lucian Georgescu, Horia Cucu, and Corneliu Burileanu. Kaldi-based dnn architectures for speech recognition in romanian. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pages 1–6. IEEE, 2019.

[3] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] Kevin McGuinness. 2nd Workshop on Deep Learning for Multimedia, Insight Dublin City University . https://github.com/telecombcn-dl/2018-dlmm/raw/master/D2L02_Transfer.pdf, Last accessed October 2020.

[5] ASER. ASER Report for the year 2018. http://img.asercentre.org/docs/ASER2018/ReleaseMaterial/aser2018nationalfindingsppt.pdf, Last accessed October 2020.

[6] The Hindu. Basic literacy, numeracy skills of rural Class VIII students in decline. https://www.thehindu.com/news/national/basic-literacy-numeracy-skills-of-rural-class-viii-students-on-a-decline-aser-2018/article26004114.ece, Last accessed October 2020.

[7] Shreeharsha B S. Spoken language assessment on mobile device. https://www.ee.iitb.ac.in/student/~shreeharsha/mng630_report_shreeharsha.pdf, 2019.

[8] Raymond D Kent. Anatomical and neuromuscular maturation of the speech mechanism: Evidence from acoustic studies. *Journal of speech and hearing Research*, 19(3):421–447, 1976.

[9] Dolly Agarwal, Jayant Gupchup, and Nishant Baghel. A dataset for measuring reading levels in india at scale. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9210–9214. IEEE, 2020.

[10] Speech Processing Lab IIT Madras. Hindi ASR challenge. `https://sites.google.com/view/asr-challenge`, last accessed: September 2020.

[11] Prakhar Swarup. Acoustic model training and adaptation for children's read speech recognition. *MTP report, Dept. of Electrical Engineering, IIT Bombay*, 2017.

[12] Nagesh Nayak. Transcription Examples of tags and labels used. `https://docs.google.com/presentation/d/1Id54pZMuGPpDf7LbSWViF42_yKGYmo92IF2HZq-nfIY/`, last accessed: October 2020.

[13] Navdeep Jaitly and Geoffrey E Hinton. Vocal tract length perturbation (vtlp) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117, 2013.

[14] John S Garofolo. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

[15] Anton Ragni, Katherine Knill, Shakti P Rath, and Mark Gales. Data augmentation for low resource languages. In *Interspeech*, 2014.

[16] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[17] Xingchen Song, Zhiyong Wu, Yiheng Huang, Dan Su, and Helen Meng. Specswap: A simple data augmentation method for end-to-end speech recognition. In *Interspeech*, 2020.

[18] Xingchen Song, Guangsen Wang, Zhiyong Wu, Yiheng Huang, Dan Su, Dong Yu, and Helen Meng. Speech-xlnet: Unsupervised acoustic model pretraining for self-attention networks. *arXiv preprint arXiv:1910.10387*, 2019.

[19] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[20] Jessica E Huber, Elaine T Stathopoulos, Gina M Curione, Theresa A Ash, and Kenneth Johnson. Formants of children, women, and men: The effects of vocal intensity variation. *The Journal of the Acoustical Society of America*, 106(3):1532–1542, 1999.

[21] Suco Eguchi. Development of speech sounds in children. *Acta Otolaryngol*, 257:1–51, 1969.

[22] James Hillenbrand, Laura A Getty, Michael J Clark, and Kimberlee Wheeler. Acoustic characteristics of american english vowels. *The Journal of the Acoustical society of America*, 97(5):3099–3111, 1995.

[23] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[24] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[25] Daniel Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005.

[26] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, pages 3743–3747, 2018.

[27] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence trained neural networks for asr based on lattice free mmi (author's manuscript). Technical report, The Johns Hopkins University Baltimore United States, 2016.

[28] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015.

[29] Desh Raj. Blog on lattice free mmi and chain models. `https://desh2608.github.io/2019-05-21-chain/`, last accessed: January 2020.

[30] Kaldi. Chain Model Doc. `https://kaldi-asr.org/doc/chain.html`, last accessed: October 2020.

[31] Kaldi. Decoding using a chain model Doc. `https://kaldi-asr.org/doc/chain.html#chain_decoding`, last accessed: October 2020.

[32] Luís Felipe Uebel and Philip C Woodland. An investigation into vocal tract length normalisation. In *Sixth European Conference on Speech Communication and Technology*, 1999.

[33] Kaldi. Applying feature transforms. `https://kaldi-asr.org/doc/transform.html#transform_apply`, last accessed: October 2020.

[34] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59. IEEE, 2013.

[35] J-L Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994.

[36] Yoo Rhee Oh and Hong Kook Kim. Mllr/map adaptation using pronunciation variation for non-native speech recognition. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 216–221. IEEE, 2009.

[37] Patrick Nguyen, Roland Kuhn, Jean-Claude Junqua, Nancy Niedzielski, and Christian Wellekens. Eigenvoices: a compact representation of speakers in model space. In *Annales des télécommunications*, volume 55, pages 163–171. Springer, 2000.

[38] Pegah Ghahremani, Vimal Manohar, Hossein Hadian, Daniel Povey, and Sanjeev Khudanpur. Investigation of transfer learning for asr using lf-mmi trained neural networks. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 279–286. IEEE, 2017.

[39] Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur. Jhu kaldi system for arabic mgb-3 asr challenge using diarization, audio-transcript alignment and transfer learning. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 346–352. IEEE, 2017.

[40] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[41] Reza Sahraeian and Dirk Van Compernolle. Using weighted model averaging in distributed multilingual dnns to improve low resource asr. *Procedia Computer Science*, 81:152–158, 2016.

[42] František Grézl, Ekaterina Egorova, and Martin Karafiát. Study of large data resources for multilingual training and system porting. *Procedia Computer Science*, 81:15–22, 2016.

[43] Shucong Zhang, Cong-Thanh Do, Rama Doddipatla, and Steve Renals. Learning noise invariant features through transfer learning for robust end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7024–7028. IEEE, 2020.

Appendix A

Miscellaneous information about transfer learning experiments

To use the IITM baseline model for retraining experiments, the phone sets used in the baseline model and the ASER UP and RJ transcription data must be mapped. A mapping between the ASER transcription phone set to the IITM phone set is derived manually by looking at examples in both the lexicons. Many of the mappings are straightforward but for some phones in the ASER phone set multiple phones in the IITM phone set had to be used. For example, 'oy' (as used in the word 'Voice' with pronunciation 'w oy s') in the ASER phone set is mapped to two phones in the IITM phone set 'ou y'.

Another change made to the lexicon is providing alternate pronunciations to nasalized words. This is done because the IITM phone set has two nasal phones 'q' and 'n'. To make use of all the phone information in the IITM phone set, nasalized words have two pronunciations. This was done automatically by identifying the unicode values of nasalized words/vowels in the lexicon. The complete mappings can be found here[1].

The other labels and tags made in the transcriptions of the ASER set are treated as words in the lexicon with SIL (silence) phone as its pronunciation. This is to provide better alignments between the ASER retraining sets and their transcriptions. There are 47 speech phones in the ASER transcription lexicon all of which are mapped to the 57 speech phones in the IITM phone set. The non-speech phones in the ASER transcription are mapped to the single SIL phone in the IITM set which is it's only non-speech phone.

While decoding all these phones are filtered out. Another point in decoding is treating the LPR and HPR recordings as separate data folders in kaldi. This optimizes the LMWTs

[1]Mapping of phones: https://rb.gy/jcjr90

and WIPs for both the sets and provides better results. The LMWT of the HPR recordings is usually much higher compared to the LMWT of the LPR recordings. To tune the retraining parameters while doing transfer learning, the validation loss is used. The parameters (number of epochs, learning rates, regularization factors) are adjusted by getting the best possible loss with the setting of freezing the 13 lower layers and tuning the final two output layers (0.0(13)-1.0(2)). Then, further experiments are carried out by modifying the differential learning rates only.