SPEECH RECOGNITION WITH PROBABILISTIC TRANSCRIPTIONS AND END-TO-END SYSTEMS USING DEEP LEARNING

BY

AMIT DAS

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor Mark A. Hasegawa-Johnson, Chair
Professor Thomas S. Huang
Associate Professor Paris Smaragdis
Professor Michael L. Oelze

Abstract

In this thesis, we develop deep learning models in automatic speech recognition (ASR) for two contrasting tasks characterized by the amounts of labeled data available for training. In the first half, we deal with scenarios when there are limited or no labeled data for training ASR systems. This situation is commonly prevalent in languages which are under-resourced. However, in the second half, we train ASR systems with large amounts of labeled data in English. Our objective is to improve modern end-to-end (E2E) ASR using attention modeling. Thus, the two primary contributions of this thesis are the following:

- **Cross-Lingual Speech Recognition in Under-Resourced Scenarios**

  A well-resourced language is a language with an abundance of resources to support the development of speech technology. Those resources are usually defined in terms of 100+ hours of speech data, corresponding transcriptions, pronunciation dictionaries, and language models. In contrast, an under-resourced language lacks one or more of these resources. The most expensive and time-consuming resource is the acquisition of transcriptions due to the difficulty in finding native transcribers. The first part of the thesis proposes methods by which deep neural networks (DNNs) can be trained when there are limited or no transcribed data in the target language. Such scenarios are common for languages which are under-resourced.

  Two key components of this proposition are *Transfer Learning* and *Crowdsourcing*. Through these methods, we demonstrate that it is possible to borrow statistical knowledge of acoustics from a variety of other well-resourced languages to learn the parameters of a the DNN in the target under-resourced language. In particular, we use well-resourced languages as cross-entropy regularizers to improve the generalization capacity of the target language. A key accomplishment of this study is that it is the first to train DNNs using noisy labels in the target language transcribed by non-native speakers available in

online marketplaces.

- **End-to-End Large Vocabulary Automatic Speech Recognition**

  Recent advances in ASR have been mostly due to the advent of deep learning models. Such models have the ability to discover complex non-linear relationships between attributes that are usually found in real-world tasks. Despite these advances, building a conventional ASR system is a cumbersome procedure since it involves optimizing several components separately in a disjoint fashion. To alleviate this problem, modern ASR systems have adopted a new approach of directly transducing speech signals to text. Such systems are known as E2E systems and one such system is the Connectionist Temporal Classification (CTC). However, one drawback of CTC is the hard alignment problem as it relies only on the current input to generate the current output. In reality, the output at the current time is influenced not only by the current input but also by inputs in the past and the future.

  Thus, the second part of the thesis proposes advancing state-of-the-art E2E speech recognition for large corpora by directly incorporating attention modeling within the CTC framework. In attention modeling, inputs in the current, past, and future are distinctively weighted depending on the degree of influence they exert on the current output. We accomplish this by deriving new context vectors using time convolution features to model attention as part of the CTC network. To further improve attention modeling, we extract more reliable content information from a network representing an implicit language model. Finally, we used vector based attention weights that are applied on context vectors across both time and their individual components. A key accomplishment of this study is that it is the first to incorporate attention directly within the CTC network. Furthermore, we show that our proposed attention-based CTC model, even in the absence of an explicit language model, is able to achieve lower word error rates than a well-trained conventional ASR system equipped with a strong external language model.

To my parents, professors, and mentors

Acknowledgments

I sincerely express my gratitude to many people for supporting me during the course of my research. This thesis would not have been possible without their help and support.

First, I would like to thank my advisor Prof. Mark Hasegawa-Johnson. Mark's broad knowledge across a wide range of disciplines and his passion for research have greatly motivated me in my research pursuits. I have enjoyed the discussions we have had over these years, the research environment, and the freedom in our group for pursuing our specific research interests. Thanks to Mark for his patience and kindness without which I would not have been able to get my conference papers reviewed and submitted at the eleventh hour. Mark's ingenuity and sharp acumen refined my thought process and often led me to consider multiple alternatives to the same problem. Beyond our lab, I will long cherish the rich academic setting provided by the Department of Electrical and Computer Engineering (ECE). It is both an honor and privilege to be associated with the faculty, students, and staff in this department. For these reasons, I feel the best learning experience I have had is at the University of Illinois at Urbana-Champaign (UIUC).

My earnest thanks to Prof. Thomas Huang, Prof. Paris Smaragdis, and Prof. Michael Oelze for their kind willingness to be a part of my defense committee and also for providing insightful comments and suggestions. These have helped greatly improve the overall quality of this work. In addition, I am thankful to Prof. Huang for providing multiple opportunities to discuss sequential modeling in his group meetings.

Next, I sincerely thank Prof. Scott Poole for his generous support during the formative years of my studies. Most of my PhD work was funded by grants from the National Science Foundation, Joan and Lalit Bahl Fellowship, Beckman Institute, and Microsoft internships. I thank these organizations and donors for their generosity.

I have greatly learned from my internship mentors. They are (in chronological

order) - Prof. Ivan Tashev, Dr. Jinyu Li, and Dr. Yifan Gong. It is because of them that I was able to expand my research work into broader domains. My past advisors, Prof. John Hansen (University of Texas at Dallas) and Prof. Umesh Srinivasan (Indian Institute of Technology Madras), have greatly helped me develop my research foundations in speech processing. I especially want to thank Prof. Hansen, who was instrumental in motivating and teaching me how to think like a researcher. Without his guidance and support, I may not have known what research is or even considered pursuing a PhD.

It is extremely hard to work through research problems without building the necessary foundations through formal coursework. I express my sincere gratitude to all the professors who have taught me during my days in graduate school. However, three professors were outstanding. They are Prof. Venugopal Veeravalli (ECE, UIUC), Prof. Rayadurgam Srikant (ECE, UIUC), and Prof. Xiaochun Li (Math, UIUC) for teaching Estimation and Detection Theory, Optimization, and Real Analysis respectively. Their lucid style of teaching and thought-provoking style of questioning made these courses quite fascinating. The materials that I learned from these courses facilitated my understanding of research articles.

Working at the UIUC has been enjoyable because of my friends and collaborators. I thank them for engaging discussions and valuable advice. I thank Po-Sen Huang, Sujeeth Bharadwaj, Xuesong Yang, Yang Zhang, Preethi Jyothi, Kaizhi Qian, Mary Pietrowicz, Sarah King, and Leda Sari. This list is incomplete without the mention of some my other entertaining friends with whom I made memorable trips in the US and beyond. They are Alok Goolya, Nek Sharan, Chaitanya Narayan, Purushottam Kumar, and Kartik Reddy.

I would like to thank Paritosh Garg, the System Administrator of our group, for his tireless efforts in keeping our servers running at all times despite his double appointments. I also thank Jennifer Carlson, ECE Assistant Director of Academic Programs, who made sure we kept track of our multiple deadlines during our busy schedules. She almost always answered my queries despite handling more than 500 graduate students in the department. I thank James Hutchinson, Publications Editor, for meticulously copyediting and proofreading this thesis. His efforts have greatly improved the overall quality of the text.

Finally, I want to thank my parents profusely for their steadfast support during my graduate studies. Despite their unfamiliarity with my work, they have always encouraged me to pursue my goals. Without this, I would not have the confidence to complete my graduate studies.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
AM	Acoustic Model
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CA	Content Attention
CD	Context Dependent
CE	Cross Entropy
CER	Character Error Rate
COMA	Component Attention
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DL	Deep Learning
DAE	Deep Auto-Encoder
DNN	Deep Neural Network
DT	Deterministic Transcript
E2E	End-to-End
fMLLR	Feature Space Maximum Likelihood Linear Regression
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HA	Hybrid Attention

HCI	Human-Computer Interaction
HMM	Hidden Markov Model
IPA	International Phonetic Alphabet
IVR	Interactive Voice Response
KD	Knowledge Distillation
KL	Kullback-Leibler
LDA	Linear Discriminant Analysis
LM	Language Model
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori Adaptation
MFCC	Mel Frequency Cepstral Coefficient
ML	Maximum Likelihood
MLLT	Maximum Likelihood Linear Transform
MLP	Multilayer Perceptron
MOE	Mixture of Experts
MSE	Mean Square Error
MT	Mismatched Transcript
MTL	Multitask Learning
OOV	Out of Vocabulary
PCA	Principal Component Analysis
PER	Phone Error Rate
PM	Pronunciation Model
PT	Probabilistic Transcript
RBM	Restricted Boltzmann Machines
RNN	Recurrent Neural Network
RNN-ED	Recurrent Neural Network Encoder Decoder
SAT	Speaker Adaptive Training

SBS	Special Broadcasting Service
SGD	Stochastic Gradient Descent
SHL	Shared Hidden Layer
SSP	Statistical Signal Processing
ST	Self-Training Transcript
TC	Time Convolution
TL	Transfer Learning
T/S	Teacher/Student
URL	Under-Resourced Language
WER	Word Error Rate
WFST	Weighted Finite State Transducer
WRL	Well-Resourced Language

Chapter 1

Introduction

1.1 Overview

Speech, vision, and text are three primary modalities through which humans communicate. Using cognition, logic, and memory, humans have learned to effectively communicate using these modalities. In other words, humans have the ability to use intelligence to process and infer information. Thus, intelligence is a complex biological phenomenon associated with humans and some animals and is still very actively studied in neuroscience and psychology.

Intelligence in machines is called artificial intelligence (AI) and is usually studied as a information processing technology. In the twentieth century, AI was mostly restricted to working within a well-defined set of formal rules. For example, the calculator was designed to perform well-defined tasks like addition, subtraction, multiplication, division etc. However, performing more human-like tasks is more challenging. For example, it is hard to define a set of formal rules to identify speakers through their voices or faces. This is where *deep learning* (DL) comes into play.

In the last decade or so, DL has become a burgeoning field of research in speech, vision, language, finance etc. In DL, machines learn about the world by observing simple concepts and building a hierarchy of more complicated concepts in a layer-wise fashion. The final layer is the most abstract layer and is usually the decision making layer. This approach of learning by experience, instead of formal rules, allows machines to perform human-like tasks like recognizing voices or faces. Therefore, this is a propitious time to study AI using the DL approach.

In this study, we focus on DL approaches for automatic speech recognition (ASR). ASR is the task of automatically converting speech into text by a machine without human intervention and is a key technology to enable human-computer interaction (HCI). Since the 1980s, ASR has been an area of active research falling

within the realms of statistical signal processing (SSP) and AI. For example, ASR is used in interactive voice response (IVR) systems to handle large volumes of telephone calls by automatically understanding callers' requests. In cellular devices, they act as dictation systems by converting a user's voice into text messages which can then be sent electronically to the desired destination. This reduces the user's effort of typing the entire message. In Global Positioning System (GPS) enabled devices, where hands-free communication is critical (for e.g., automobiles), an ASR system is able to convert a driver's commands into text which can then be processed by the GPS device to display the routing information. In the media, they are used to automatically transcribe large amounts of spoken news into text. Contemporary ASR devices include Microsoft Cortana, Apple Siri, Amazon Alexa, and Google Home.

1.2 Motivation

Trends in ASR research [2–4] have changed dramatically over the past decade. The traditional way of building ASR models using hidden Markov models (HMMs) has been revolutionized with the introduction of DL models such as deep neural networks (DNNs) [5], convolutional neural networks (CNNs) [6], and recurrent neural networks (RNNs) [7, 8]. Their popularity is mostly attributed to the fact that neural networks achieve much lower error rates than Gaussian mixture models (GMMs), especially with large training corpora. However, these systems are manifest only in a few countries where languages are well-resourced. A well-resourced language is a language (e.g. English) with an abundance of resources to support development of speech technology. Those resources are usually defined in terms of 100+ hours of speech data, corresponding transcripts, pronunciation dictionaries, and language models. Among these, the most expensive and time-consuming resource is the acquisition of transcripts. Primarily for this reason, more than 99% of 6900 languages in the world are still under-resourced [9]. As a result, one language dies every two weeks on an average [10]. Building ASR systems for such languages can help either slow down or even stop this decline since these systems will encourage people to continue using these languages in their daily lives.

In the first part of the thesis, we focus on developing ASRs for under-resourced languages in two scenarios. First, we build ASRs with very limited amounts of

transcriptions collected from native transcribers using transfer learning. In the second and more adverse scenario, we assume we do not have access to native transcribers at all. This is a realistic scenario since it is quite hard to find native transcribers in under-resourced languages. However, Turkers (crowd workers) available in online marketplaces can serve as valuable alternative resources by providing transcriptions in the target language. Since the Turkers may neither speak nor have any familiarity with the target language, their transcriptions are non-native by nature and are usually filled with incorrect labels. After some post-processing, these transcriptions can be converted to probabilistic transcriptions (PT). Conventional DNNs trained using PTs do not necessarily improve error rates over GMMs due to the presence of label noise. To alleviate this problem, we propose a variety of multi-task learning (MTL) training regimes by which we are able to train DNNs in the target language using noisy transcriptions.

In the second part of the thesis, we move our focus to building ASRs in English which is a well-resourced language. However, it is well-known that building a conventional ASR system in English is a cumbersome procedure since it involves optimizing several components separately in a disjoint fashion. To alleviate this problem, modern E2E systems such as the CTC framework [1, 11] directly transduce speech signals to text in a single model. However, one drawback of CTC is the hard alignment problem as it relies only on the current input to generate the current output. In reality, the output at the current time is influenced not only by the current input but also by inputs in the past and future. Thus, we propose advancing state-of-the-art E2E ASR for large corpora by directly incorporating attention modeling [12, 13] within the CTC framework. In attention modeling, inputs in the current, past, and future are distinctively weighted depending on the degree of influence they exert on the current output.

1.3 Thesis Contributions

The main contributions of this thesis can be summarized as follows:

- In the case when there are very limited amounts of native transcriptions in the target language, we proposed cross-lingual adaptation using regularized cross-entropy training of DNNs [14]. Data from well-resourced languages act as regularizers during training.

- In the case when there are no native transcriptions in the target language, we trained ASRs in the target language using noisy non-native transcriptions collected from crowdworkers [15]. In particular, we proposed a MTL training regime which uses a mixture of noisy transcriptions in the target under-resourced language and clean transcriptions from several well-resourced languages [16, 17].

- We proposed another MTL using a deep auto-encoder (DAE) which is used as one of the sub-tasks in the MTL system [18]. The DAE uses the unlabled data in the target language as its ground truth targets and attempts to minimize the mean square error between its predictions and the ground truth targets.

- Furthermore, we proposed knowledge distillation and target interpolation as ways to improve the generalization capacity of the MTL system [19].

- Finally, for large corpora ASR in English, we proposed solving the hard alignment problem in CTC models by directly incorporating attention modeling [20–22].

1.4 Thesis Structure

The remainder of this thesis is organized as follows.

- In Chapter 2, we provide the necessary background for cross-lingual and end-to-end speech recognition.

- In Chapter 3, we propose training GMM-HMMs and DNNs in the target language when there are very limited amounts of transcribed data in the target language.

- In Chapter 4, we propose training DNNs in the target language when there are no transcribed data in the target language. Instead, we use train DNNs using transcripts generated by online non-native crowdworkers.

- In Chapter 5, we propose an attention-based CTC that directly transduces speech waveforms into characters or words by focusing on the most relevant parts of the utterance.

- In Chapter 6, we summarize the contributions of this study and discuss future directions for research.

Chapter 2

Background

2.1 Cross-Lingual Speech Recognition in Under-Resourced Scenarios

Deep neural network (DNN) based automatic speech recognition (ASR) systems achieve significantly lower error rates than Gaussian mixture models (GMMs) or hidden Markov models (HMMs), especially when large training corpora are available. However, these systems are manifest only in a few countries where languages are well-resourced. A well-resourced language (WRL) is a language with an abundance of resources to support development of speech technology. For example, English is the most well-resourced language. These resources can be defined in terms of the following attributes:

- 100+ hours of speech/acoustic data
- Transcripts corresponding to the speech data
- Pronunciation dictionaries and vocabulary lists
- Language models
- Strong presence on the web making the data accessible online

When these resources are available, it becomes possible to build *matched acoustic models*. The underlying structure of a matched acoustic model is illustrated in Fig. 2.1. Here, an utterance spoken in the target language (L1) is transcribed by a native transcriber in the same language (L1) using the native orthography of the target language. An acoustic model trained using the native transcript and the features extracted from the utterance is called a matched acoustic model.

On the other hand, an under-resourced language (URL) is a language with some (if not all) of the following resources: lack of electronic resources for speech and

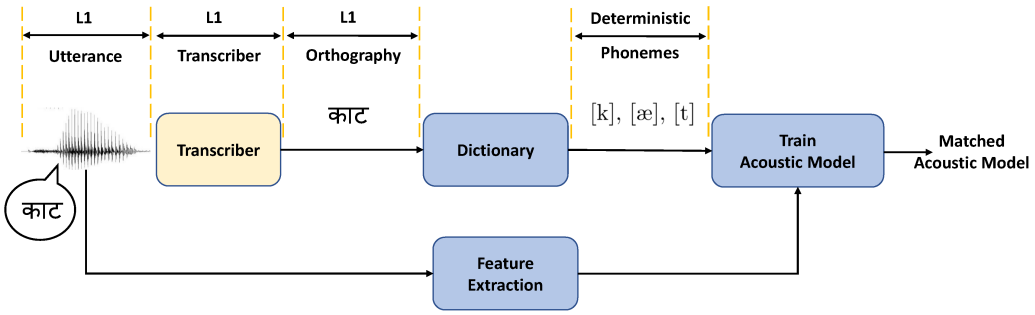

Figure 2.1: A matched acoustic model. An utterance in the target language, L1 (Hindi in this example), is transcribed by a native transcriber in L1 using the native orthography. A dictionary is used to convert the words in L1 to IPA phonemes. An acoustic model is trained using the IPA phonemes and the features extracted from the utterance.

language processing, such as monolingual corpora, bilingual electronic dictionaries, transcribed speech data, pronunciation dictionaries, and vocabulary lists; lack of a unique writing system or stable orthography; limited presence on the web; and lack of linguistic expertise [9]. Other factors include lack of infrastructure in the native country, and banking discrimination by western banks which makes it difficult to develop resources for URLs. Such languages are sometimes referred to as low-density languages, resource-poor languages, low-data languages, or less-resourced languages. However, a URL is not the same as a minority language, which is a language spoken by a minority of the population of a territory. Some URLs are actually official languages of their country and spoken by a very large population. For example, Bahasa Indonesia in Indonesia, Khmer in Cambodia, Amharic in Ethiopia, Dinka in South Sudan, and Uzbek in Uzbekistan are URLs but not minority languages. On the other hand, there are some minority languages that can be considered as WRLs. For example, the Catalan language is both a minority language and a WRL since resources for Catalan are available on Google Search and Google Translate. Consequently, URLs are not necessarily endangered (while the opposite is usually true) [9].

We will use the terms "*URL*" or "*target language*" interchangeably to refer to the language to be recognized. Similarly, we will use "*WRLs*" or "*source languages*" to refer to the auxiliary languages for which we have training data. However, the objective is not to recognize these languages.

Among all resources required for building ASR models, the most expensive and time-consuming resource is the acquisition of transcripts. Perhaps for this reason,

more than 99% of 6900 languages in the world still do not have well-developed ASR systems. This means that there are few or no natively transcribed transcripts easily available in URLs.

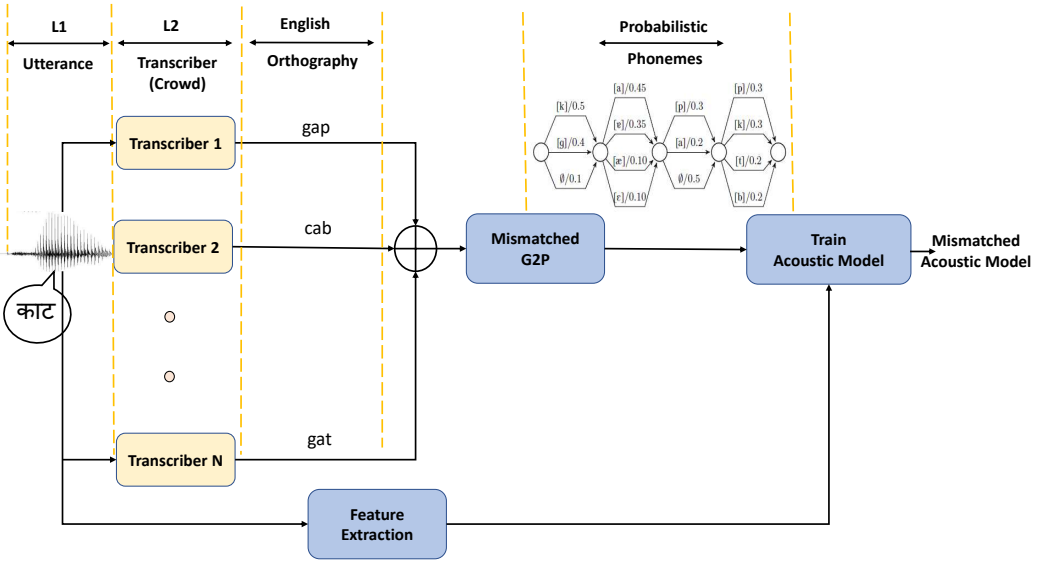

Figure 2.2: A mismatched acoustic model. An utterance in the target language, L1 (Hindi in this example), is transcribed by multiple non-native Turkers using the English orthography. A grapheme-to-phoneme (G2P) model is used to convert the English words to IPA phonemes. An acoustic model is trained using the lattice of IPA phonemes and the features extracted from the utterance.

To circumvent this difficulty, transcripts can be collected from online non-native crowd workers, or Turkers, who neither speak the target language nor have any familiarity with it. An acoustic model built from such non-native Turker transcripts is called a *mismatched acoustic model*. The underlying structure of a mismatched acoustic model is illustrated in Fig. 2.2. Briefly, a single utterance in the target language (L1) is transcribed by multiple Turkers who do not speak the target language. Due to this mismatch between the utterance language and the Turker's native language, no single Turker can generate the correct transcript. Instead, a collection of transcripts from multiple Turkers is constructed for a single utterance. After merging these transcripts and some post-processing, we get a lattice of transcripts which represent a probabilistic distribution over several transcripts. An acoustic model trained using this lattice of non-native transcripts and the features extracted from the utterance is called a mismatched acoustic model. Non-native transcripts are usually *noisy* or *inaccurate*. One of objectives of this study is to train DNNs using such noisy transcripts.

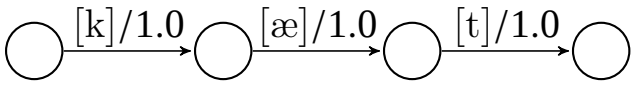

Figure 2.3: A deterministic transcription (DT) for the word *cat*.

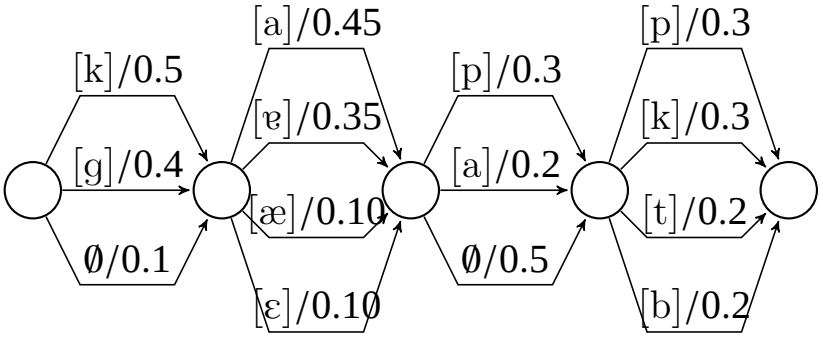

Figure 2.4: A probabilistic transcription (PT) for the word *cat*.

We now introduce some terminologies that will be frequently used in this study.

2.1.1 Deterministic Transcript (DT)

Consider the situation in Fig. 2.1. An utterance is played to a transcriber. If the utterance language is the same as the transcriber's native language, then there is a match between the languages. The native transcriber is likely to transcribe the contents of the utterance with a high degree of accuracy. Since there is no ambiguity in the ground truth labels (syllables, words etc.) that the transcriber provides, the labels are deterministic in nature. Transcripts of this kind are called *deterministic transcripts* (DTs). An example of a DT for the word "cat" is shown in Fig. 2.3. Each arc represents a label and a probability value which is 1.0 always. DTs are simply conventional transcripts which are part of many popular speech corpora like TIMIT, Wall Street Journal (WSJ) etc.

2.1.2 Mismatched Transcript (MT)

Consider the situation in Fig. 2.2. An utterance in Hindi (most widely spoken language in India) is played to a non-native transcriber. Non-native transcribers found in online marketplaces are called crowd workers or Turkers. Examples of popular online marketplaces are Amazon and Upwork. In the absence of native transcribers, these workers become valuable alternative resources for providing transcripts in the utterance language.

Since the utterance language (Hindi) is *not* the same as the Turker's native language, there is a mismatch between the utterance language and the Turker's language. Because of the Turker's unfamiliarity with Hindi, the Turker writes down non-sense syllables in English. Such a transcript is known as *mismatched transcript* (MT). For more details on the preparation of these transcripts, readers are encouraged to refer to [23]. The non-native Turker is unlikely to transcribe the contents of the utterance with a high degree of accuracy. In particular, the Turker is unlikely to distinguish all phone pairs in the utterance language. Consequently, an MT is likely to be *noisy*. However, these transcripts can be useful to train ASR systems in the absence of native transcribers.

2.1.3 Probabilistic Transcript (PT)

An MT can be post-processed and formed into a single confusion network consisting of labels and probability values associated with those labels. Such a confusion network is called a *probabilistic transcript* (PT) [24] and is shown in Fig. 2.4. The arc weight specifies the conditional probability that the phoneme was spoken, given the evidence in the transcripts. Because crowd workers cannot distinguish all phone pairs in the utterance language, these weights are usually less than 1.0. Therefore, a PT is, at best, a probability distribution over the labels provided by crowd workers. Unlike the DT in Fig. 2.3 which has a single sequence of symbols, the PT has 3×4×3×4 = 144 possible sequences, one of which could be the right sequence. In this case, it is "k æ ∅ t" (∅ is the empty symbol). There is another useful interpretation of DTs and PTs. The DTs can also be thought of as 1-hot alignments that are frequently observed in conventional transcripts. How-

Table 2.1: Deterministic transcript (DT) vs Probabilistic transcript (PT).

	DT	PT
Transcribers	Native	Non-native
Transcript Structure	Single stream	Lattice
Probability	1.0	$[0, 1]$
Label Noise	Low	High
Availability	Difficult	Easy
Cost	Expensive	Cheap

ever, in the case of PTs, the alignments are soft since a single frame could have multiple labels with non-zero probabilities. In the illustrated example, the 1-hot alignment (DT) for the word "cat" is $[1.0\ \mathrm{k}], [1.0\ \mathrm{æ}], [1.0\ \mathrm{t}]$. Here, *a* and *b* in [*a b*] denote the probability of the label and the label respectively. On the other hand, the soft alignment (PT) is $[0.5\ \mathrm{k}, 0.4\ \mathrm{g}, 0.1\ \emptyset]$, $[0.45\ \mathrm{a}, 0.35\ \mathrm{ɐ}, 0.1\ \mathrm{æ}, 0.1\ \mathrm{ɛ}]$, $[0.3\ \mathrm{p}, 0.2\ \mathrm{a}, 0.5\ \emptyset]$, $[0.3\ \mathrm{p}, 0.3\ \mathrm{k}, 0.2\ \mathrm{t}, 0.2\ \mathrm{b}]$. An overview of the differences in DTs and PTs are summarized in Table 2.1.

2.2 End-to-End Models for Large Vocabulary Automatic Speech Recognition

Recent advances in ASR have been mostly due to the advent of DL algorithms such as deep neural networks (DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). Despite these advances, building a conventional ASR system is a cumbersome procedure since it involves training several components in the ASR pipeline in a disjoint fashion. A conventional ASR system is shown in Fig. 2.5.

In ASR, we are given a sequence of feature vectors $\mathbf{x}$ which is a compact representation of the speech waveform in an utterance. The objective is to decode the sequence of words $\mathbf{y}$ from $\mathbf{x}$ with minimum probability of error. This translates to the maximum a posteriori (MAP) problem,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \ P(\mathbf{y}|\mathbf{x}; \Theta_{\text{ASR}}) \tag{2.1}$$

$$= \arg\max_{\mathbf{y}} \ P(\mathbf{x}|\mathbf{y}; \Theta_{\text{AM}}) P(\mathbf{y}; \Theta_{\text{LM}}) \tag{2.2}$$

$$\approx \arg\max_{\mathbf{y},\mathbf{l}} \ P(\mathbf{x}|\mathbf{l}; \Theta_{\text{AM}}) P(\mathbf{l}|\mathbf{y}; \Theta_{\text{PM}}) P(\mathbf{y}; \Theta_{\text{LM}}), \tag{2.3}$$

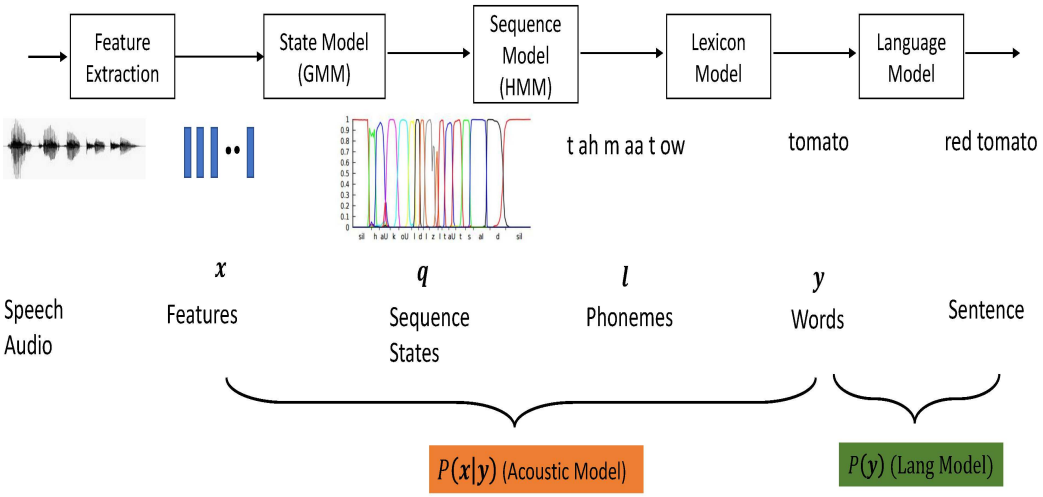

Figure 2.5: A conventional ASR system.

where $\Theta_{\text{ASR}} = \{\Theta_{\text{AM}}, \Theta_{\text{PM}}, \Theta_{\text{LM}}\}$ is the set of parameters to be estimated and l is a sequence of phonemes. The first term $P(\mathbf{x}|\mathbf{l}; \Theta_{\text{AM}})$ in Eq. (2.3) is the likelihood of the features given the phoneme sequence and is obtained from an acoustic model (AM). The second term $P(\mathbf{l}|\mathbf{y}; \Theta_{\text{PM}})$ is the likelihood of the phoneme sequence given the word sequence and is obtained from a lexicon or pronunciation model (PM). The third term $P(\mathbf{y}; \Theta_{\text{LM}})$ is the prior probability of the word sequence and is obtained from a language model (LM).

In practice, the AM, PM, and LM models are trained separately. Thus, the ASR problem becomes a complex disjoint learning problem. Apart from this, the decoding process during test time involves a complex graph search step and fine-tuning other empirical parameters such as the scaling factor of AM likelihood and the word-insertion penalty.

In contrast, an end-to-end (E2E) ASR system, shown in Fig. 2.6, directly models the posterior distribution $p(\mathbf{y}|\mathbf{x}; \Theta_{\text{ASR}})$ by transducing an input sequence of acoustic feature vectors to an output sequence of words (or more generally tokens). The output sequence of tokens is better known as a transcription. Thus, this makes it possible for all the components to be jointly trained as in Eq. 2.1 instead of Eq. (2.3).

More specifically, for an input sequence of feature vectors $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$ of length T with $\mathbf{x}_t \in \mathbb{R}^m$, an E2E ASR system transduces the input sequence to an intermediate sequence of hidden feature vectors $\mathbf{h} = (\mathbf{h}_1, \cdots, \mathbf{h}_L)$ of length L with $\mathbf{h}_l \in \mathbb{R}^n$. Usually, $L = T$. The sequence $\mathbf{h}$ undergoes another transduction resulting

in the posterior probability of the transcription, $\tilde{p}(\mathbf{y}|\mathbf{x})$, where $\mathbf{y} = (y_1, \cdots, y_U)$ is a transcription of length U with $y_u \in \mathbb{L}$, $\mathbb{L}$ being the label set. Here, $K = |\mathbb{L}|$ is the cardinality of the label set $\mathbb{L}$. In ASR, the labels could be senones, graphemes, letters, words etc. depending on the desired granularity of outputs. Usually $U \leq T$ which means that an E2E system is able to convert input to output sequences of different lengths. Thus, an E2E neural network, parameterized by $\mathbf{W}$, learns a many-to-one functional $\mathbf{f_W} : \mathbf{x} \mapsto \tilde{p}(\mathbf{y}|\mathbf{x})$ where $\tilde{p}(\mathbf{y}|\mathbf{x})$ closely resembles the true $p(\mathbf{y}|\mathbf{x})$.

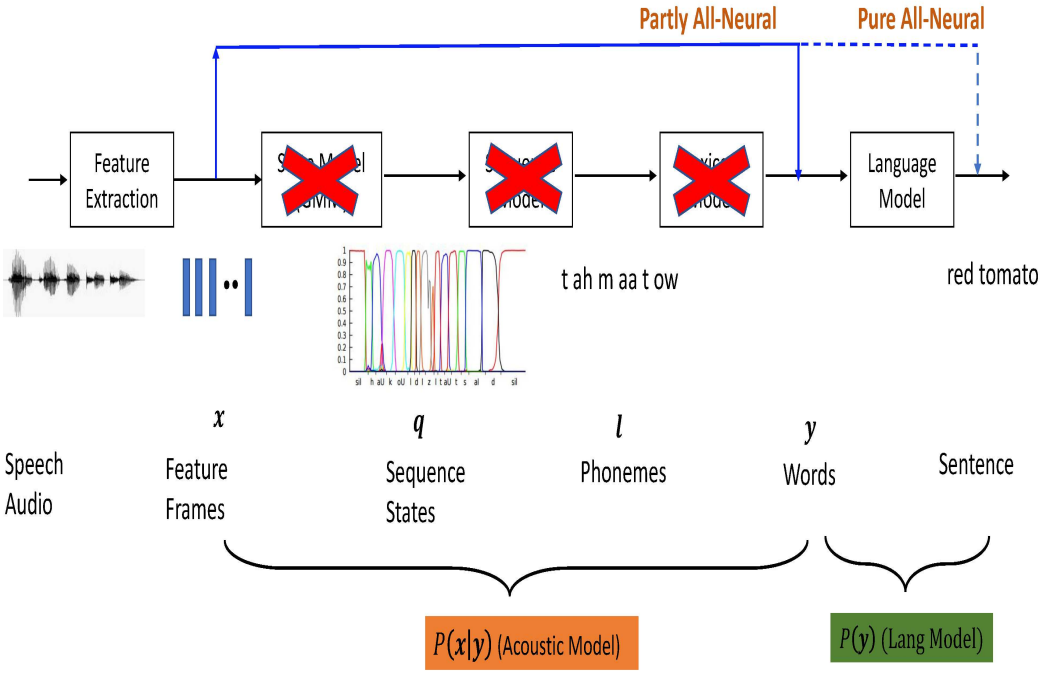

Figure 2.6: An end-to-end ASR system.

2.2.1 Connectionist Temporal Classification (CTC)

RNNs used in ASR optimize the Kullback-Leibler (KL) divergence between the probability distributions of frame predictions and ground truth labels. This forces the network to align its frame predictions with the ground truth alignments. A ground truth alignment is a sequence of labels, one label per frame. The labels in these alignments are usually phonemes. Alignments are usually obtained as a result of the HMM based forced alignment procedure (constrained Viterbi decoding). However, for ASRs, the desired outputs are larger linguistic units such as characters or words rather than smaller units such as phonemes. The CTC [1, 11]

error criterion directly optimizes prediction of larger linguistic units thereby circumventing the need for generating smaller linguistic units such as phonemes.

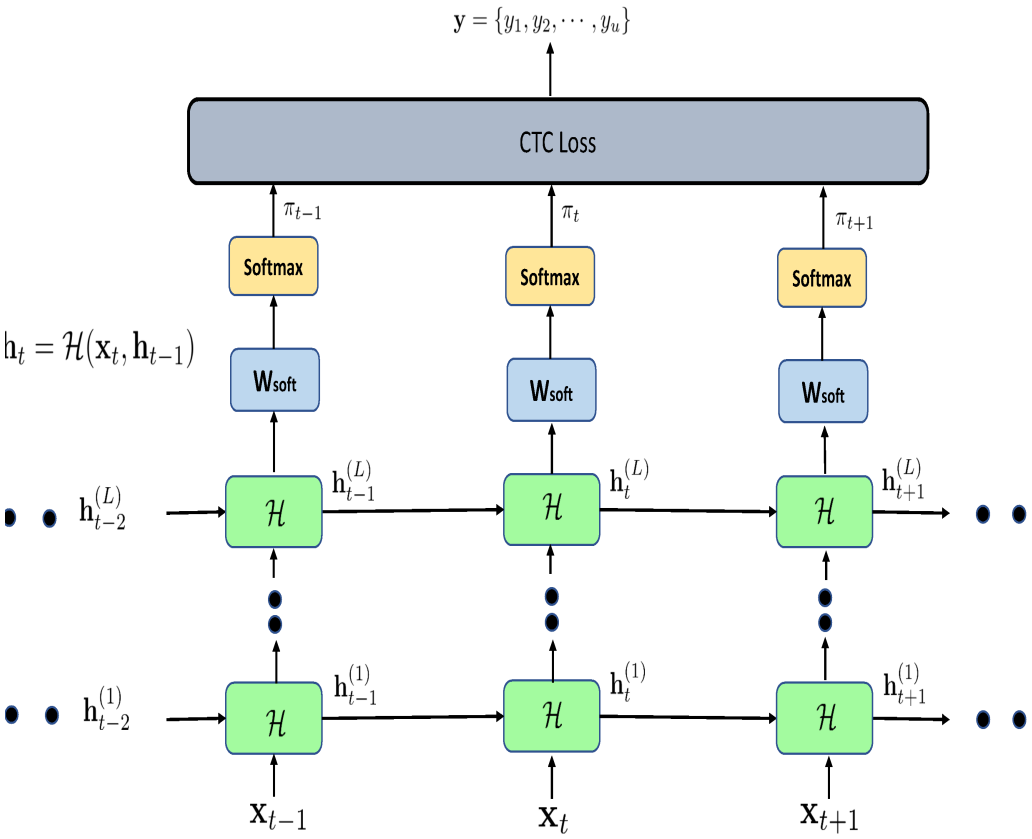

Figure 2.7: An example of a CTC network.

A CTC network uses a recurrent neural network (RNN) and the CTC error criterion [1, 11] which directly optimizes the prediction of a transcription sequence. The basic structure of a CTC network is shown in Fig. 2.7. As the length of the transcription is shorter than the length of input acoustic vectors, CTC introduces an intermediate label representation called a *path* denoted by $\boldsymbol{\pi} = \{\pi_1, \cdots, \pi_T\}$. The intermediate label at time t is denoted by π_t. A CTC path $\boldsymbol{\pi}$ has the same length as the input sequence $\mathbf{x}$ which is made possible by adding a blank symbol $\{\varnothing\}$ as an additional label and allowing repetition of labels. Thus, after the addition of $\varnothing$, the lengths of input $\mathbf{x}$, hidden $\mathbf{h}$, and intermediate output sequences $\boldsymbol{\pi}$ are the same. Because of the addition of $\varnothing$, we have an extended label set $\mathbb{L}'$, where $\mathbb{L}' = \mathbb{L} \cup \{\varnothing\}$ with cardinality $K + 1$. Because of this, $\pi_t \in \mathbb{L}'$.

The advantage of adding a blank label is that it does not force the network to make non-blank predictions for frames whose predictions are weak. The posterior

path probability of $\boldsymbol{\pi}$ is defined as

$$p(\boldsymbol{\pi}|\mathbf{x}) \overset{\text{CI}}{=} \prod_{t=1}^{T} p(\pi_t|\mathbf{x}), \tag{2.4}$$

where the equality is based on the assumption that the network output at time t, and $p(\pi_t|\mathbf{x})$ is conditionally independent of outputs at other times $p(\pi_{\neq t}|\mathbf{x})$. In other words, $\pi_t \perp\!\!\!\perp \pi_{\neq t}|\mathbf{x}$. The network output $p(\pi_t|\mathbf{x})$ is the RNN softmax activation of the intermediate label π_t such that $\sum_{\pi_t \in \mathbb{L}'} p(\pi_t|\mathbf{x}) = 1$ and noting that t is fixed in the summation.

To produce the final output sequence $\mathbf{y}$ (transcription), CTC defines a many-to-one function $B : \boldsymbol{\pi} \mapsto \mathbf{y}$ which maps multiple CTC paths to a single transcription.

The path $\boldsymbol{\pi}$ represents an intermediate sequence of labels at every frame. However, the final desired output sequence is a human-readable transcription $\mathbf{y}$. To this end, CTC defines a many-to-one map $\mathbf{B} : \boldsymbol{\pi} \mapsto \mathbf{y}$ which compresses the path $\boldsymbol{\pi}$ of length T to a transcription $\mathbf{y}$ of length $U \leq T$. This is achieved by first removing the repeated labels from the path $\boldsymbol{\pi}$ and then removing the blanks. For example, $\mathbf{B}(cc - aa - -t) = \mathbf{B}(c - a - t) = cat$. With this, the transcription probability of $\mathbf{y}$ given $\mathbf{x}$ is the sum of probabilities of all those paths which can be compressed using $\mathbf{B}$ to generate $\mathbf{y}$. This can be written as

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in \mathbf{B}^{-1}(\mathbf{y})} p(\boldsymbol{\pi}|\mathbf{x}), \tag{2.5}$$

where $\mathbf{B}^{-1}(\mathbf{y})$ is the pre-image of $\mathbf{y}$. The CTC loss function can then be defined so that the network learns to maximize the transcription probability (or to minimize the negative log probability) of the ground truth transcriptions in the training set. Thus,

$$\mathcal{L}_{\text{CTC}} = -\sum_{\mathbf{l} \in \text{ train}} \ln p(\mathbf{y}|\mathbf{x}). \tag{2.6}$$

This training criterion directly optimizes the probability of the transcription rather than frame level path or alignment. For decoding, it is very simple to generate the transcription using greedy decoding: simply concatenate the tokens corresponding to posterior spikes in CTC to generate the transcription.

However, CTC has some limitations.

- First, CTC is harder to train than a standard long short-term memory (LSTM)

network since it is sensitive to initialization. In [25], CTC training was initialized from a LSTM network trained with large amounts of data using the frame level cross entropy criterion.

- Second, the conditional independence assumption in Eq. (2.4) for speech data, in general, is not true. Due to this constraint, CTC does not model inter-label dependencies very well although it can be argued that the recurrent structure in RNN implicitly models time dependencies. Therefore, during decoding, the CTC framework relies on external language models to achieve good ASR accuracy. More details about CTC training are covered in [1, 11].

2.2.2 RNN Encoder-Decoder (RNN-ED)

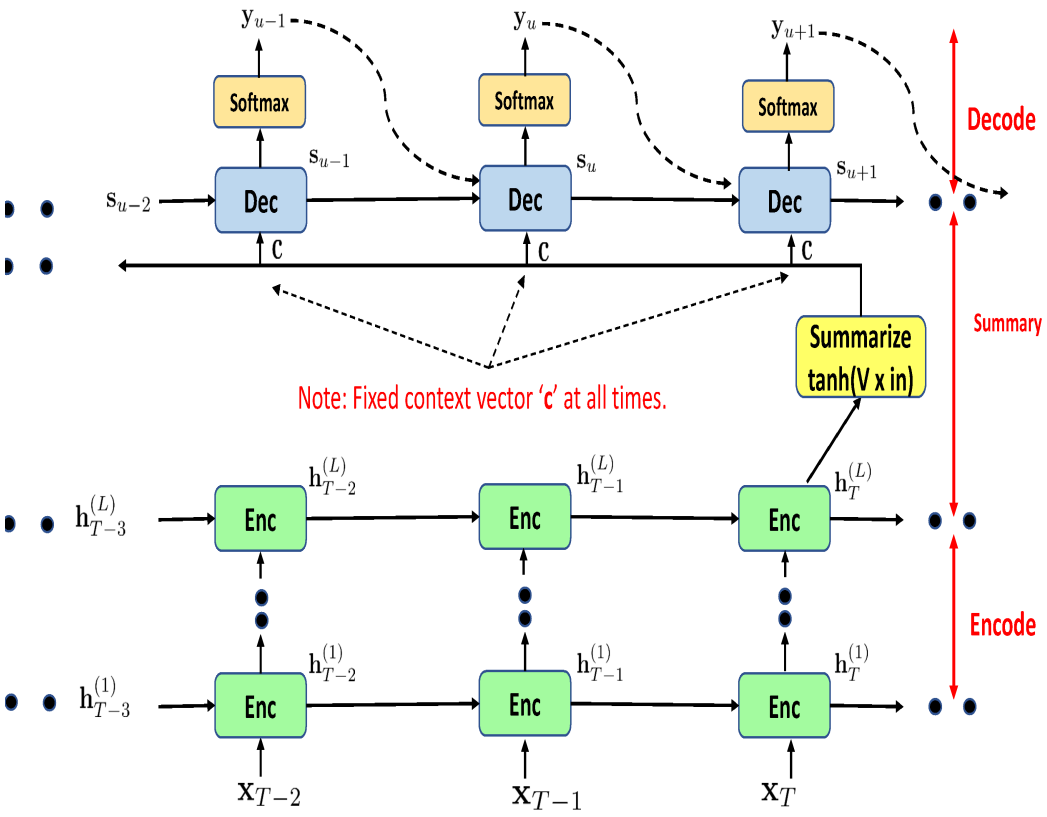

Figure 2.8: An example of an RNN-ED network.

An RNN-ED [12, 13] uses two distinct RNNs: an RNN encoder that transforms **x** to **h** and an RNN decoder that transforms **h** to **y**. The basic structure of an

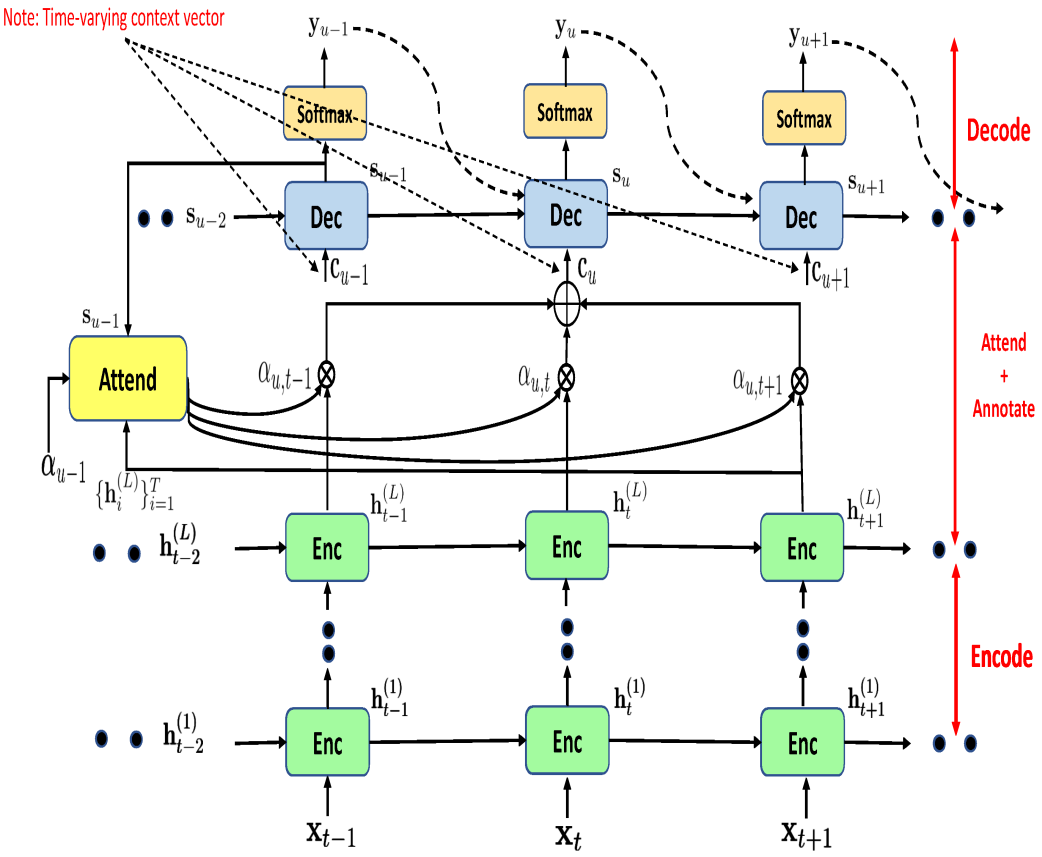

Figure 2.9: An example of an attention-based RNN-ED network.

RNN-ED network is shown in Fig. 2.8. It models $p(\mathbf{y}|\mathbf{x})$ as

$$p(\mathbf{y}|\mathbf{x}) = \prod_{u=1}^{U} p(\mathbf{y}_u|\mathbf{c}_u, \mathbf{y}_{1:u-1}), \tag{2.7}$$

where $\mathbf{c}_u$ is a function of $\mathbf{x}$ and is sometimes called as *context vector* or *soft alignment*. Here, the subscript u in $\mathbf{c}_u$ is a time step. The context vector can be a constant across all time steps (thus, $\mathbf{c}_u = \mathbf{c}, \forall u$) or more generally can be time-dependent in the case of attention-based RNN-ED [26, 27]. An attention-based RNN-ED network is shown in Fig. 2.9.

The RNN-ED encoder computes

$$\mathbf{h}_t = \text{Encode}(\mathbf{x}_t, \mathbf{h}_{t-1}). \tag{2.8}$$

Encode(.) function is simply a unidirectional or bidirectional RNN.

The RNN-ED decoder has two components: a multinomial distribution generator Eq. (2.9), and an RNN decoder Eq. (2.10). In addition, an attention-based RNN-ED [26, 27] is equipped with an attention network Eq. (2.11)-Eq. (2.16) as follows:

$$p(\mathbf{y}_u|\mathbf{y}_{1:u-1}, \mathbf{s}_u, \mathbf{c}_u) = \text{Generate}(\mathbf{y}_{u-1}, \mathbf{s}_u, \mathbf{c}_u), \tag{2.9}$$

$$\mathbf{s}_u = \text{Recurrent}(\mathbf{s}_{u-1}, \mathbf{y}_{u-1}, \mathbf{c}_u), \tag{2.10}$$

$$\mathbf{c}_u = \text{Annotate}(\boldsymbol{\alpha}_u, \mathbf{h}) = \sum_{t=1}^{T} \alpha_{u,t}\mathbf{h}_t \tag{2.11}$$

$$\boldsymbol{\alpha}_u = \text{Attend}(\mathbf{s}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{h}). \tag{2.12}$$

Here, $\mathbf{h}_t, \mathbf{c}_u \in \mathbb{R}^n$ and $\boldsymbol{\alpha}_u \in \mathbb{U}^T$, where $\mathbb{U} = [0, 1]$, such that $\sum_t \alpha_{u,t} = 1$. Also, for simplicity $\mathbf{s}_u \in \mathbb{R}^n$. Generate(.) is a feedforward network with a softmax operation [13, Appendix A.2.2] generating the probability of the target output $p(\mathbf{y}_u|\mathbf{y}_{u-1}, \mathbf{s}_u, \mathbf{c}_u)$. Recurrent(.) is an RNN decoder and is similar to the recurrency in *Encode*(.). However, Recurrent(.) operates on the output time axis indexed by u and its hidden state is $\mathbf{s}_u$. Annotate(.) computes the context vector $\mathbf{c}_u$ (soft alignment) using the attention probability vector $\boldsymbol{\alpha}_u$ and the hidden sequence $\mathbf{h}$. The scalar weight $\alpha_{u,t} \in \mathbb{U}$ determines the influence of $\mathbf{h}_t$ in generating $\mathbf{c}_u$. Attend(.)

computes the attention weight $\alpha_{u,t}$ using a single-layer feedforward network as

$$e_{u,t} = \text{Score}(\mathbf{s}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{h}_t), \quad t = 1, \cdots, T \tag{2.13}$$

$$\alpha_{u,t} = \frac{\exp(e_{u,t})}{\sum_{t'=1}^{T} \exp(e_{u,t'})}, \tag{2.14}$$

where $e_{u,t} \in \mathbb{R}$. Score(.) can either be content-based attention or hybrid-based attention. The latter encodes both content ($\mathbf{s}_{u-1}$) and location ($\boldsymbol{\alpha}_{u-1}$) information. Score(.) is computed using

$$e_{u,t} = \begin{cases} \mathbf{v}^T \tanh\left(\mathbf{U}\mathbf{s}_{u-1} + \mathbf{W}\mathbf{h}_t + \mathbf{b}\right), & \text{(content)} \\ \mathbf{v}^T \tanh\left(\mathbf{U}\mathbf{s}_{u-1} + \mathbf{W}\mathbf{h}_t + \mathbf{V}\mathbf{f}_{u,t} + \mathbf{b}\right), & \text{(hybrid)} \end{cases} \tag{2.15}$$

$$\text{where,} \quad \mathbf{f}_{u,t} = \mathbf{F} * \boldsymbol{\alpha}_{u-1}. \tag{2.16}$$

The operation $*$ denotes convolution. The bias term $\mathbf{b}$ is optional. Attention parameters $\mathbf{U}$, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{F}$, $\mathbf{b}$, $\mathbf{v}$ are learned while training RNN-ED.

When only the content function of Eq. (2.15) is used, Eq. (2.12) is usually referred to as the *content attention model* [27]. On the other hand, when the hybrid function of Eq. (2.15) is used, Eq. (2.12) is usually referred to as the *hybrid attention model* [27].

There are two key differences between CTC and RNN-ED. First, $p(\mathbf{y}|\mathbf{x})$ in Eq. (2.7) is generated using a product of ordered conditionals. Thus, RNN-ED relaxes the conditional independence constraint of Eq. (2.4) in CTC. Second, there is no intermediate label representation $\boldsymbol{\pi}$ in RNN-ED.

Chapter 3

Cross-Lingual Adaptation Using Limited Native Transcriptions

3.1 Introduction

Often there are situations when the target language that needs to be recognized has very limited amounts of transcribed data. When limited amounts of transcribed data are available in the target language, training ASR systems with the limited data often leads to poor generalization. To alleviate this issue, transfer learning techniques can be used to transfer the acoustical knowledge from the source languages to the target language. This scenario of leveraging knowledge from source languages (WRLs) to build ASR systems in the target language (URL) is usually referred to as *cross-lingual adaptation* or *cross-lingual recognition*. This is the main focus of this chapter.

The remainder of the chapter is organized as follows. In Section 3.2, we provide a summary of past work. In Section 3.3, we introduce some common notations that will be used throughout this chapter. In Section 3.4 and Section 3.5, we explain the proposed cross-lingual adaptation using regularized ML training of GMM-HMM and regularized CE training of DNN respectively. Finally, in Section 3.6, we describe the experiments and outline the results.

3.2 Background

Many interesting research studies have improved the performance of state-of-the-art cross-lingual speech recognition. One of the earlier approaches includes bootstrapping target language acoustic models based on phonetic similarity either using existing monolingual [28], or multilingual models [29], [30]. Recently, DNNs have spurred interest in the speech recognition community due to their superior discriminative modeling capabilities compared to GMM-HMM based modeling

techniques. In [5], the outputs of a hybrid DNN-HMM system were used to represent posterior probabilities of shared context-dependent states (senones). DNNs have been used in cross-lingual recognition through tandem or hybrid approaches. In the class of tandem approaches, either (a) posteriors as the final layer outputs of DNNs are Gaussianized [31,32], or (b) the outputs of an intermediate layer (bottleneck extractions) [33, 34], followed by dimensionality reduction using principal component analysis (PCA) are used as distinctive features for training GMM-HMM classifiers. In the class of hybrid approaches, the alignments from GMM-HMM systems are treated as ground truth labels to train DNNs using the CE criterion. After completing training, the posteriors from the trained DNN are used for classification of test data. It has been shown that, when few target language data are available, unsupervised pre-training of DNN hidden layers with multilingual data [35] can outperform hidden layers trained with monolingual data [36], [37]. In [38], DNNs were used for knowledge transfer with zero training data using an "open-target MLP" - an MLP designed to generate posteriors for all possible monophones in the IPA table. DNNs have been effective since the hidden layers are able to learn complex feature transformations. The complex features are then classified using a logistic regression classifier at the final layer.

Transfer learning has been successfully implemented for semi-supervised learning [39, 40] and supervised learning [41] of GMMs. In this study, we focus on knowledge transfer from WRLs to an URL in two supervised settings - while training GMM-HMM and DNN-HMM.

3.3 Notations

Let $\mathbf{x}$ be a sequence of feature vectors, one feature vector per frame, of a language indexed by the superscript. The sequence $\mathbf{x}$ can be represented as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$ where the subscript indicates the time index, $\mathbf{x}_t \in \mathbb{R}^d$, and N is the number of frames. Associated with each $\mathbf{x}_t$ is a label. In speech recognition, labels are usually states (monophones, context-dependent (CD) phones, senones etc.), or graphemes, or words depending on the granularity of the features. In the current context, since the granularity is at the frame level, states are used as labels. Assume there is a total of C states. Then the set of states for this language is $\mathcal{S} = \{1, 2, \cdots, C\}$.

In ASR, however, the speech corpora usually do not provide any state infor-

mation associated with each frame. An utterance can be considered as a sequence of frames $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$ and the corresponding sequence of states $\mathbf{q} = (q_1, \cdots, q_N)$ where $q_t \in \mathcal{S}$ is a random variable denoting the state at time t. A sequence of states is popularly referred to as an *alignment*. Since the true alignments are unknown, these need to be estimated during training.

The objective is to estimate the parameters of an HMM or a DNN so that the HMM/DNN is adapted to the target language. Since the data available in the target language is sparse, we make use of both the target language and a pool of source languages (or multilingual data).

3.4 Cross-Lingual Adaptation Using Regularized Maximum Likelihood Training of GMM-HMM

The modeling parameters of the HMMs for the target language are given by $\{\boldsymbol{\Theta}_c\}_{c=1}^{C}$ where each $\boldsymbol{\Theta}_c$ corresponds to a set of parameters for the state c. Each HMM consists of three CD states, arranged left-to-right, with each state modeled by a GMM with diagonal covariance matrices. The individual states are connected by non-zero transition probabilities. Thus, each HMM consists of the parameters $\boldsymbol{\Theta}_c = \left\{\pi_1, a_{ij}, \left\{\omega_m, \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}\right\}_{m=1}^{M}\right\}_{i,j=1}^{3}$ where M is the total number of mixtures of a GMM. The objective is to learn $\{\boldsymbol{\Theta}_c\}_{c=1}^{C}$ by using limited training data from the target language and large amounts of multilingual data from source languages. To learn the parameters of an HMM, the objective function to be maximized is the log-likelihood function of the training data. Since the training data consist of both the target and source data, the likelihood of the target data is regularized with the weighted likelihood of the source data. Hence, the new objective is to maximize the total likelihood which is given by

$$\mathcal{L}(\mathbf{x}; \boldsymbol{\Theta}_c) = \mathcal{L}(\mathbf{x}^{(1)}; \boldsymbol{\Theta}_c) + \rho \mathcal{L}(\mathbf{x}^{(2)}; \boldsymbol{\Theta}_c), \tag{3.1}$$

where $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ represent the training data of the target and source languages respectively, and

$$\mathcal{L}(\mathbf{x}^{(l)}; \boldsymbol{\Theta}_c) = \frac{1}{N^{(l)}} \sum_t \log p(\mathbf{x}_t^{(l)}; \boldsymbol{\Theta}_c), \quad l = 1, 2, \tag{3.2}$$

and ρ is a regularization constant taking values within $[0, 1]$. The optimal parameter set is given by

$$\boldsymbol{\Theta}_c^{\star} = \arg\max_{\boldsymbol{\Theta}_c} \mathcal{J}(\boldsymbol{\Theta}_c), \quad c = 1, \cdots, C. \tag{3.3}$$

The corresponding auxiliary function for the new objective function becomes

$$\begin{aligned} Q(\boldsymbol{\Theta}_c, \boldsymbol{\Theta}_c^0) = & \frac{1}{N^{(1)}} \sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x}^{(1)}; \boldsymbol{\Theta}_c^0) \log p(\mathbf{x}^{(1)}, \mathbf{q}; \boldsymbol{\Theta}_c) \\ & + \frac{\rho}{N^{(2)}} \sum_{\mathbf{q}} p(\mathbf{q}|\mathbf{x}^{(2)}; \boldsymbol{\Theta}_c^0) \log p(\mathbf{x}^{(2)}, \mathbf{q}; \boldsymbol{\Theta}_c), \end{aligned} \tag{3.4}$$

where $\boldsymbol{\Theta}_c^0$ is the initial model and the summation is taken over all possible alignments $\mathbf{q}$. Given an initial model $\boldsymbol{\Theta}_c^0$, the maximum likelihood (ML) parameters, under the constraints $\sum_{m=1}^{M} \omega_{jm} = 1$ and $\boldsymbol{\Sigma}_{jm} \succ 0$ (j^{th} state and m^{th} mixture in class c), are found using the expectation-maximization (EM) algorithm. Finding the parameters using EM is straightforward and is given by

$$\pi_i = \frac{\frac{1}{N^{(1)}} \sum_t \alpha_t^{(1)}(i)\beta_t^{(1)}(i) + \frac{\rho}{N^{(2)}} \sum_{t'} \alpha_{t'}^{(2)}(i)\beta_{t'}^{(2)}(i)}{\frac{1}{N^{(1)}} \sum_{t,i} \alpha_t^{(1)}(i)\beta_{t'}^{(1)}(i) + \frac{\rho}{N^{(2)}} \sum_{t',i} \alpha_{t',(1)}^{(2)}(i)\beta_{t',(1)}^{(2)}(i)}, \tag{3.5}$$

$$a_{ij} = \frac{\frac{1}{N^{(1)}} \sum_t \xi_t^{(1)}(i, j) + \frac{\rho}{N^{(2)}} \sum_{t'} \xi_{t'}^{(2)}(i, j)}{\frac{1}{N^{(1)}} \sum_t \gamma_t^{(1)}(i) + \frac{\rho}{N^{(2)}} \sum_{t'} \gamma_{t'}^{(2)}(i)}, \tag{3.6}$$

$$\omega_{j,m} = \frac{\frac{1}{N^{(1)}} n_{jm}^{(1)}(1) + \frac{\rho}{N^{(2)}} n_{jm}^{(2)}(1)}{\frac{1}{N^{(1)}} \sum_m n_{jm}^{(1)}(1) + \frac{\rho}{N^{(2)}} \sum_m n_{jm}^{(2)}(1)}, \tag{3.7}$$

$$\boldsymbol{\mu}_{jm} = \frac{\frac{1}{N^{(1)}} n_{jm}^{(1)}(\mathbf{x}) + \frac{\rho}{N^{(2)}} n_{jm}^{(2)}(\mathbf{x})}{\frac{1}{N^{(1)}} n_{jm}^{(1)}(1) + \frac{\rho}{N^{(2)}} n_{jm}^{(2)}(1)}, \tag{3.8}$$

$$\boldsymbol{\Sigma}_{jm} = \frac{\frac{1}{N^{(1)}} n_{jm}^{(1)}(\mathbf{x}^2) + \frac{\rho}{N^{(2)}} n_{jm}^{(2)}(\mathbf{x}^2)}{\frac{1}{N^{(1)}} n_{jm}^{(1)}(1) + \frac{\rho}{N^{(2)}} n_{jm}^{(2)}(1)}, \tag{3.9}$$

where

$$n_{jm}^{(l)}(1) = \sum_t \gamma_t^{(l)}(j,m),$$
$$n_{jm}^{(l)}(\mathbf{x}) = \sum_t \gamma_t^{(l)}(j,m)\mathbf{x}_t^{(l)},$$
$$n_{jm}^{(l)}(\mathbf{x}^2) = \sum_t \gamma_t^{(l)}(j,m)(\mathbf{x}_t^{(l)} - \mu_{jm})(\mathbf{x}_t^{(l)} - \mu_{jm})^T.$$

The quantities $\gamma_t^{(l)}(j,m), \xi_t^{(l)}(i,j)$ are defined in [42, Eq. (27, 37)].

3.5 Cross-Lingual Adaptation Using Regularized Cross-Entropy Training of DNN

A DNN takes an input frame (or a feature vector) $\mathbf{x}_t$, which then undergoes multiple layers of successive affine transformations followed by element-wise non-linearities, to output a vector of posterior probabilities $\mathbf{y}_t$. Thus, the DNN models the posterior probabilities of predicting the state given $\mathbf{x}_t$. Thus, corresponding to the sequence $\mathbf{x}$, there is a sequence of vectors of posterior probabilities $\mathbf{y} = (\mathbf{y}_1, \cdots, \mathbf{y}_N)$ where $\mathbf{y}_t \in \mathbb{U}^C$ with $\mathbb{U} = [0,1]$. The k^{th} component of $\mathbf{y}_t$ is the posterior probability of the occurrence of state k at time t given that $\mathbf{x}_t$ was observed. This can be explicitly stated as

$$y_t(k) = p(q_t = k|\mathbf{x}_t), \quad k = 1, \cdots, C. \tag{3.10}$$

Since $\mathbf{y}_t$ represents a probability distribution over states, $\sum_{k=1}^{C} y_t(k) = 1$.

The output of layer l, denoted by $\mathbf{u}^l$, is obtained by applying the affine transform (using $\mathbf{W}^l, \mathbf{b}^l$) on the outputs of the previous layer $\mathbf{u}^{l-1}$ followed by a sigmoid activation $\sigma(.)$. This can be represented as

$$\mathbf{u}^l = \sigma(\mathbf{W}^l\mathbf{u}^{l-1} + \mathbf{b}^l), \quad 1 \leq l < L. \tag{3.11}$$

Here, $\mathbf{W}^l$ is the weight matrix between layers $l-1$ and l, $\mathbf{b}^l$ is the bias vector at layer l. For the first layer ($l = 1$), $\mathbf{u}^0 = \mathbf{x}_t$. For the final layer L (softmax layer),

the output at node k, $u^L(k)$, is given by

$$u^L(k) = \frac{\exp(\mathbf{w}_k^{L^T}\mathbf{u}^{L-1} + b_k^L)}{\sum_j \exp(\mathbf{w}_j^{L^T}\mathbf{u}^{L-1} + b_j^L)}, \tag{3.12}$$

where $\mathbf{w}_k^{L^T}$ is the k^{th} row of matrix $\mathbf{W}^L$. The output $u^L(k)$ is simply the posterior probability $p(q_t = k|\mathbf{x}_t)$ as defined in Eq. (3.10). The emission probability $p(\mathbf{x}_t|q_t)$ is obtained using Bayes' theorem as

$$p(\mathbf{x}_t|q_t) = \frac{p(q_t|\mathbf{x}_t)p(\mathbf{x}_t)}{p(q_t)}, \tag{3.13}$$

where the state priors $p(q_t)$ are obtained by simply counting the state labels from the HMM based forced alignments of the training data. The term $p(\mathbf{x}_t)$ can be ignored since during Viterbi decoding (max. operation) it is treated as a constant for all values of t. The DNN is trained to minimize the negative log posterior probability of the training examples in the sequence $\mathbf{x}$. This is given as

$$\begin{aligned}
E &= -\sum_t \log\, p(\mathbf{d}_t|\mathbf{x}_t) \\
&= -\sum_t \log \prod_{k=1}^{C}\, p(q_t = k|\mathbf{x}_t)^{d_t(k)} \\
&= -\sum_t \sum_{k=1}^{C} \log\, p(q_t = k|\mathbf{x}_t)^{d_t(k)} \\
&= -\sum_t \sum_{k=1}^{C} d_t(k) \log\, p(q_t = k|\mathbf{x}_t) \\
&= -\sum_t \sum_{k=1}^{C} d_t(k) \log\, y_t(k),
\end{aligned} \tag{3.14}$$

where $d_t(k)$ is the ground truth label for $\mathbf{x}_t$. The second step is due to the frame independence assumption. Eq. (3.14) is the cross-entropy (CE) between the desired target vector $\mathbf{d}_t$ and the DNN output vector $\mathbf{y}_t$. The desired target $d_t(k)$ is constrained to $d_t(k) \in [0, 1]$ such that $\sum_{k=1}^{C} d_t(k) = 1$ and is obtained from HMM based forced alignments. The DNN output $y_t(k)$ is obtained from Eq. (3.12).

In this study, a modified CE error criterion is used that takes into account the CE error of both the target data and source data similar to Eq. (3.1). The modified

CE error criterion is

$$E = E^{(1)} + \rho E^{(2)}, \tag{3.15}$$

where $E^{(1)}, E^{(2)}$ are the CE errors of the form (3.14) for target and source languages respectively with ρ being a regularization constant taking values within $[0, 1]$. The subscripts of E given inside the parenthesis indicate the language index.

A DNN trained using Eq. (3.15) has a slightly modified weight update rule. The CE error E is a result of the training errors of individual frames from either of the languages. The errors backpropagated to node k of the final layer (L) of the DNN given $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ are

$$\delta_k^{(1),L} \triangleq \frac{\partial E_t^{(1)}}{\partial a_k^L} = (y_t^{(1)}(k) - d_t^{(1)}(k)), \quad k = 1, \cdots, C \tag{3.16}$$

$$\delta_k^{(2),L} \triangleq \frac{\partial \rho E_t^{(2)}}{\partial a_k^L} = \rho(y_t^{(2)}(k) - d_t^{(2)}(k)), \quad k = 1, \cdots, C, \tag{3.17}$$

where $a_k^L = \mathbf{w}_k^{L^T}\mathbf{u}^{L-1} + b_k^L$ is the output of the affine transformation at the softmax layer before going through the softmax activation. These errors are backpropagated to the layers below the L^{th} layer. During backpropagation, the errors at the layers below ($\delta_k^{L-1}, \delta_k^{L-2}, \ldots$ etc.) are computed as a linear combination of the errors at the layer above with the weights being the connection weights between two successive layers. Thus the effect of having a scaling term ρ in Eq. (3.17) is reflected as scaled errors at the lower layers. This leads to

$$\frac{\partial E_t^{(1)}}{\partial w_{kj}^l} = \delta_k^{(1),l} u_j^{l-1}, \tag{3.18}$$

$$\frac{\partial \rho E_t^{(2)}}{\partial w_{kj}^l} = \rho \delta_k^{(2),l} u_j^{l-1}, \tag{3.19}$$

where $w_{k,j}^l$ is the weight connecting j^{th} input node to k^{th} output node at the l^{th} layer. From Eq. (3.19), it is clear that the error gradient with respect to the weight w_{kj}^l at the l^{th} layer is directly proportional to δ_k^l scaled by ρ. No such scaling occurs for the error in Eq. (3.18). The error gradient matrix due to the target language can be

defined as

$$\nabla E^{(1)} = \begin{bmatrix} \frac{\partial E_t^{(1)}}{\partial w_{11}^l} & \frac{\partial E_t^{(1)}}{\partial w_{12}^l} & \cdots & \frac{\partial E_t^{(1)}}{\partial w_{1J}^l} \\ \frac{\partial E_t^{(1)}}{\partial w_{21}^l} & \frac{\partial E_t^{(1)}}{\partial w_{22}^l} & \cdots & \frac{\partial E_t^{(1)}}{\partial w_{2J}^l} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E_t^{(1)}}{\partial w_{K1}^l} & \frac{\partial E_t^{(1)}}{\partial w_{K2}^l} & \cdots & \frac{\partial E_t^{(1)}}{\partial w_{KJ}^l} \end{bmatrix}. \tag{3.20}$$

For the error gradient matrix due to source languages, the only change is setting the subscript of E to 2. During training, frames from both target and source languages are presented in a randomized fashion. Hence, the weight update rule using gradient descent will contain gradients from both languages as follows:

$$\mathbf{w}(\tau) = \mathbf{w}(\tau - 1) - \eta \nabla E^{(1)} - \eta\rho \nabla E^{(2)}, \tag{3.21}$$

where τ is the iteration step, and η is the learning rate. Thus the effect of the regularization constant ρ with $E^{(2)}$ in Eq. (3.15) is a reduced learning rate $\rho\eta$ for frames belonging to the source languages as given in Eq. (3.21).

3.6 Experiments and Results

In this section, we present the results for the methods described in Section 3.4 and Section 3.5. We used Turkish as the target language and English as the source language.

3.6.1 Data

The Turkish corpus in [43] was used. Its training set consists of a total of 3974 utterances (4.6 hours) spoken across 100 speakers. Data for the training speaker *s1012* was discarded due to lack of transcriptions.

On an average, each training utterance is about 4.12 seconds long. Its full test set consists of 752 utterances spoken across 19 speakers. In this study, 558 utterances from 14 randomly selected speakers constitute the test set. The remaining utterances across 5 speakers constitute the development set. For English, the TIMIT training set consists of 3696 (462 speakers, 3.14 hours). The Turkish corpus follows the METUBET based phonetic representation [43]. Since the

phonetic systems are different for Turkish and TIMIT, it is important that both the systems be mapped to a single system prior to running any experiment. In this study, the WORLDBET [44] system was used since its alphabets cover a wide range of multilingual phones and it is represented in the amicable ASCII format. A summary of Turkish and English phone inventories is given in Table 3.1. Turkish has a more compact phone set than English. There are only 4 vowels that are common to both the languages; Turkish distinguishes rounded vs. unrounded vowels at every place, whereas English distinguishes tense vs. lax vowels. Hence the vowel coverage of Turkish using English is only 40% (4/10).

Table 3.1: Turkish and English phone set. M = Monophthongs, D = Diphthongs, NS = Non-syllabics, S = Syllabics.

Language	Vowels		Consonants		Total
	M	D	NS	S	
Turkish	10	0	28	0	38
English	13	5	27	3	48
Common	4	0	20	0	24

3.6.2 Baseline HMM

Context-dependent GMM-HMM acoustic models for Turkish and English were trained using 39-dimensional Mel frequency cepstral coefficients (MFCCs) which include the delta and acceleration coefficients. Temporal context was included by splicing 7 successive 13-dimensional MFCC vectors (current +/- 3 frames) into a high dimensional supervector and then projecting the supervector to 40 dimensions using linear discriminant analysis (LDA). Using these features, a maximum likelihood linear transform (MLLT) [45] was computed to transform the means of the existing model. The final model is the LDA+MLLT model. For the English recognition system, the forced alignments obtained from the LDA+MLLT model were further used for speaker adaptive training (SAT) by computing feature-space maximum likelihood linear regression (fMLLR) transforms [46]. This is the LDA+MLLT+SAT model. The forced alignments from this model were used for training Turkish models which is discussed next. The resulting phone error rates (PER) from a total of 27K phones are given in Table 3.2. The results for Turkish show the error rate that would be achieved by a monolingual system if the full

Table 3.2: PERs of CD GMM-HMM models using full training sets of Turkish and TIMIT.

GMM-HMM Models	PER (%)
Turkish (LDA+MLLT)	24.25
TIMIT (LDA+MLLT+SAT)	19.6

training set were to be available. The results for TIMIT are based on a reduced phone set of cardinality 39 [47]. All experiments were conducted using the Kaldi toolkit [48].

3.6.3 Regularized Maximum Likelihood Training of GMM-HMM

Phones sharing the same WORLDBET symbol were mapped between the two languages. This work differs from previous works [49] involving such hard semantic maps in that we do not completely rely on the knowledge transfer involving such maps. This is evident from the setting $\rho < 1$ in Eq. (3.1). This is justified because the phonetic variations associated with a phone in one language can be different from the phonetic variations in another language, even though the two language-dependent phones are canonically transcribed using the same WORLDBET symbol. Second, we also map some phones from English to Turkish even though they do not share the same WORLDBET symbols. This many-to-one mapping was based on the degree of similarity in articulation between the two sounds. This is important in the context of limited availability of data in the target language. For example, English vowels were mapped to those Turkish vowels that were closest in terms of tongue height followed by fronting. Since Turkish does not have any diphthong and English has falling diphthongs, only the first vowel of the diphthongs due to their higher prominence were mapped to the closest vowel in Turkish. After these mappings, there were still 8 Turkish phones which could not be mapped. Therefore, the minimum, maximum, and average number of English phones mapped to Turkish phones were 0, 4, and 1.23 respectively.

We converted the triphone alignments of English to Turkish using the above mapping rules *before* proceeding for monophone training of Turkish HMMs. Monophones were trained using the criterion in Eq. (3.1). For triphone training, we build a decision tree for each central phone with the leaves representing a variety of senones for that central phone. Since each senone can represent multiple contexts, differences in contexts between Turkish and English are easily addressed

through these senones. Therefore, cross-lingual knowledge transfer occurs both at the monophone and triphone stages using Eq. (3.1). However, the transfer is more effective at the triphone stage since learning triphone models contain a larger number of model parameters and hence require more data to be learned. At the LDA+MLLT stage of training, there is no knowledge transfer because the LDA transform cannot be shared between languages. However, knowledge transfer during the triphone stage helps in generating better forced alignments thereby leading to better models at any subsequent stage of training.

In Table 3.3, the PERs are shown for varying amounts of Turkish training data (100 to 1000 utterances, out of the available 3974). The first row is the baseline (BL) LDA+MLLT system trained using only the limited Turkish training set. There is no knowledge transfer from English in this system. In the second row is the transfer learned (TL) LDA+MLLT system that uses data from *both* the languages. Compared to the BL LDA+MLLT system, the relative improvement in performace of TL LDA+MLLT system is in the range 0.95–2.35%. Expectedly, with increasing amounts of training data the difference in performance begins to shrink. The value of ρ is determined using the dev set in each case. We used $\rho = 10^{-2}$ for the first two cases (100, 200) and decreased this by an order of magnitude with each further doubling of the amount of data.

Based on the relative increase in PER, it is clear that these improvements due to transfer learning at the HMM stage are marginal. However, when forced alignments obtained from the TL LDA+MLLT system are used to train DNNs, signficant improvements can be obtained as discussed in the next section.

Table 3.3: PERs for LDA+MLLT models trained with limited Turkish utterances and the entire TIMIT set.

# Turkish Utterances	100	200	500	1000
	PER (%)			
(a) BL LDA+MLLT	44.75	39.50	33.65	29.47
(b) TL LDA+MLLT	43.70	38.57	32.92	29.19
Relative PER $\downarrow$ (%)	2.35	2.35	2.17	0.95

3.6.4 Regularized Cross-Entropy Training of DNN

In the first step, we build multilingual shared hidden layers (MSHLs) by using greedy layer-wise unsupervised training of stacked restricted Boltzmann machines

(RBMs). We do not build monolingual SHLs since it is well known that they are outperformed by MSHLs [36], [37]. Hence, all DNN experiments, inlcuding the baseline, use MSHLs.

We obtained multilingual audio files from the Special Broadcasting Service (SBS) network which publishes multilingual radio broadcasts in Australia. These data include include over 1000 hours of speech in 70 languages. We used about 20 hours of data divided equally between all 70 languages since choice of languages is not important for pre-training and larger amounts of data may not necessarily yield significant gains [35]. We use 6 layers to build the MSHLs with 1024 nodes per layer. The input features to the bottom layer, the Gaussian-Bernoulli RBM, included 5 neighboring frames containing 39-dimensional MFCC vectors spliced together and globally normalized to zero mean and unit variance. The learning rate was set to 0.01. For all subsequent layers, the Bernoulli-Bernoulli RBMs, we used a learning rate of 0.4. Mini-batch size was set to 100 for all layers. All layers were randomly initialized.

After training the MSHLs, we proceed for supervised training of the Turkish DNN by adding a randomly initialized softmax layer. Therefore, all DNNs reported in Table 3.4 use 6 MSHLs and a randomly initialized softmax layer to classifiy senones. The DNNs in Table 3.4 differ in the type of training and labeled utterances used during the fine-tuning stage. These differences are explained in the next three paragraphs. The learning rate was fixed at 0.008 until cross-validation accuracy between two successive epochs fell below 0.5%. The learning rate was halved for all subsequent epochs until the overall accuracy failed to increase by at least 0.1%. At this point, the algorithm terminates.

The PER results for various DNNs are given in Table 3.4. The first DNN is the baseline (BL) DNN trained on alignments generated by the BL LDA+MLLT system (no knowledge transfer from English) in Table 3.3, part (a). The second DNN is trained on alignments generated by the TL LDA+MLLT system (knowledge transfer from English) in Table 3.3, part (b). The relative improvement of PERs are in the range 0.36-6.18%. Both the DNNs are trained in the same way: MSHLs, then add random softmax, then use forced alignments in Turkish to fine-tune. The only difference is in the quality of Turkish alignments that were used to train the two DNNs. The alignments were generated by the HMMs in Table 3.3. The quality of alignments generated by the TL LDA+MLLT HMM in Table 3.3 is much better than the BL LDA+MLLT HMM which leads to training better DNNs.

In the third DNN, the DNN is trained using the modified training error crite-

Table 3.4: PERs for DNN models trained with HMM state alignments obtained from Table 3.3.

MSHL + rand softmax +	# Turkish Utterances			
	100	200	500	1000
	PER (%)			
BL DNN (No Transfer):				
(a) Train using 3.3(a) ali	45.98	38.75	31.73	26.63
TL DNN (Transfer):				
(b) Train using 3.3(b) ali	43.14	38.61	30.96	26.10
Relative PER ↓ (%) (b-a)	6.18	0.36	2.43	1.99
TL DNN (Transfer):				
(c) Joint	42.11	37.81	30.55	26.23
Relative PER ↓ (%) (c-a)	8.42	2.43	3.72	1.50
TL DNN (Transfer):				
(d) Seq: L2 (2 iter)	39.90	35.98	29.78	25.73
(e) Seq: L2 (6 iter)	39.57	35.61	**29.44**	**25.37**
(f) Seq: L2 (10 iter)	**39.25**	**35.51**	29.56	25.39
Best relative PER ↓ (%)	14.64	8.36	7.22	4.73

rion shown in Eq. (3.15). This requires using alignments from *both* Turkish and a limited number (about 100 utterances) from English. While Turkish alignments were obtained from TL LDA+MLLT system, English alignments were obtained from the TIMIT LDA+MLLT+SAT system and converted to alignments in terms of Turkish phones using English to Turkish mapping rules as was explained in Section 3.6.3. We refer to this type of supervised training as "joint" training in Table 3.4. The relative PERs improve further except for the last case (1000 utterances). The relative improvements in PER in Table 3.4 are always computed with respect to the PER of BL DNN (first row of Table 3.4).

In the next set of DNNs, we again use alignments from *both* Turkish and English as before, although in a sequential manner. First, we train the DNN using English alignments converted to Turkish phones using early stopping and then retrain the same DNN using Turkish alignments until the termination criterion determined by cross-validation accuracy is met. We refer to this type of supervised training as "sequential" training where we first train the DNN using the source language (English or L2) for a few iterations and then retrain the same DNN using the target language (Turkish). We also observed that early stopping while training in L2 leads to better PERs. Here, the early stopping criterion is to train the DNN for a fixed number of epochs in L2 (2-10) epochs. For cases where target

data was very limited (100 or 200), the number of L2 epochs was 10. Otherwise, 6 epochs were sufficient. More epochs do not guarantee better accuracies. As demonstrated in Table 3.4, PER improves in each case by 4.73–14.64% relative (1.26–6.73% absolute), with an average improvement of 8.74% relative (3.38% absolute). Through these experiments, it is clear that knowledge transfer can also occur at the supervised training stages.

We think that initializing weights by sequential training is closest to the work on MLP initialization schemes of Vu et al. [38]. In [38], they use the weights of a multilingual MLP to initialize the weights of a target language MLP. Their target language MLP used monophone based posteriors and the hidden layer weights were initialized using the multilingual MLP whereas the softmax layer weights were initialized randomly. The key differences in this work are: (a) the DNNs are deeper than the MLPs, (b) the DNNs use CD phones instead of monophones in [38], and (c) the DNNs are able to leverage the knowledge of the phonetic structure of the CD space by using source language senones. This is helpful especially in under-resourced scenarios.

3.7 Summary

In this study, cross-lingual transfer learning methods using supervised training were investigated for limited resource scenarios. A regularized maximum likelihood training criterion was proposed for training GMM-HMMs using labeled data from both target and source languages. Next, a regularized cross-entropy training criterion was proposed for training DNNs which also uses labeled data from both languages. Finally, it was shown that DNNs could also be trained sequentially using both languages.

Chapter 4

Cross-Language Transfer Using Crowdsourced Non-Native Transcriptions

4.1 Introduction

In Chapter 3, we presented cross-lingual adaptation in scenarios where transcribed data in the target language are limited. In this chapter, we extend this further to an even more adverse scenario where there are no target language data at all.

As in Chapter 3, we use the terms "*URL*" or "*target language*" interchangeably to refer to the language to be recognized. Similarly, we use "*WRLs*" or "*source languages*" to refer to the auxiliary languages for which we have training data but the objective is not to recognize these languages.

When there are no transcribed data (i.e., transcriptions) available in the target language, it is hard to build ASR systems in the target language that can perform reasonably well. However, transfer learning techniques can be used to transfer the acoustical knowledge from the source languages to the target language. In the absence of transcribed data in the target language, leveraging knowledge from source languages (WRLs) to build ASR systems in the target language (URL) is usually referred to as *cross-language transfer*.

Lack of transcribed data in the target language can be attributed to the difficulty of finding native transcribers. Another reason is the lack of large government funded programs. However, there exist alternative resources for collecting transcribed data. For example, transcriptions can be collected from online non-native crowd workers, or Turkers, who neither speak the target language nor have any familiarity with it. We briefly outline this procedure. A single utterance in some target language L is transcribed by multiple Turkers who do not speak L. Due to this, no single Turker can generate the correct transcription. Instead, a collection of transcriptions from multiple Turkers is constructed for a single utterance in L. This collection, after merging and some post-processing, can be represented as a confusion network which we refer to as *probabilistic transcript* (PT). In contrast,

the correct transcription generated by a native speaker can be represented as a single sequence of labels. We will refer to this sequence as a *deterministic transcript* (DT). DTs are simply conventional transcriptions that we frequently encounter in large vocabulary speech corpora. In Chapter 2, we outlined the differences between PTs and DTs. A key difference between PTs and DTs is that the labels in PTs are noisy. This main focus of this chapter is on training DNNs using PTs in the target language (URL). At no point do we use DTs to train these DNNs. Thus, in this sense, this chapter deals with *zero resource speech recognition*.

The remainder of the chapter is organized as follows. In Section 4.2, we provide a summary of past work. In Section 4.3 and Section 4.4, we propose the models to adapt to PTs using weakly supervised and semi-supervised learning respectively. In Section 4.5, we describe the experiments and outline the results followed by a summary in Section 4.6.

4.2 Background

In [15, 50], it was shown that it is possible to adapt HMMs (pre-trained using DTs in WRLs) to an URL using PTs. In this study, the objective is to investigate DNN training techniques that can adapt to the URL. Forced alignments obtained from PT adapted HMMs are treated as "ground truth" labels for DNN training. Since these alignments are based on PTs and not DTs, the "ground labels" are soft and noisy rather than 1-hot and clean.

Self Training: One possibility is to ignore the soft labels in PTs since they are noisy and instead use a self-training method. Here, a well-trained ASR system decodes the utterances in the URL and then uses the decoded labels and its confidences to adapt itself to the URL. This was earlier used in monolingual [51] and multilingual scenarios [52]. In [52], the multilingual ASR system was used to decode the utterances in an unseen target language and was then retrained using the decoded labels to adapt to the target language. This is illustrated in Fig. 4.1. However, this method does not make use of PTs in the URL.

Vanilla DNN: In order to make use of the PTs, a better way is to use the conventional approach to adapt a multilingual DNN to a new language. This is achieved by preserving the shared hidden layers (SHLs) [36] of an existing multilingual DNN and then replacing the weights in the multilingual trained softmax layer with randomly initialized weights to form a new softmax layer. The new soft-

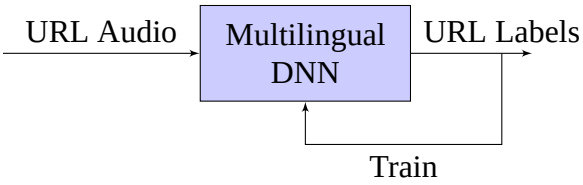

Figure 4.1: Self-training ASR.

max layer is fine-tuned using the labels of only the target language [37]. This is illustrated in Fig. 4.2.

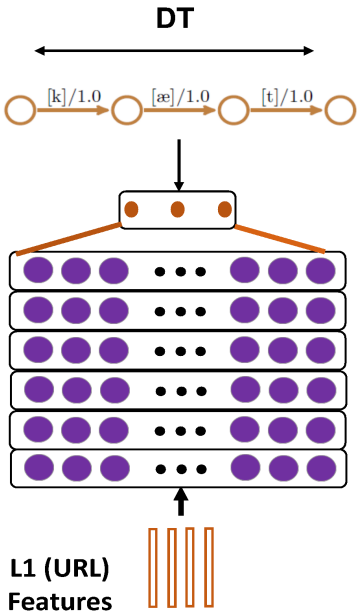

Figure 4.2: Vanilla DNN trained with DTs.

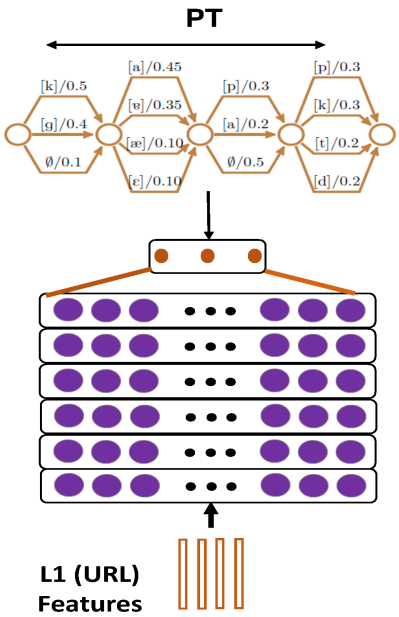

Figure 4.3: Vanilla DNN trained with PTs.

However, in the current scenario, there are no DTs. Hence, an obvious step is to use the PTs to fine-tune the softmax layer. This is the Vanilla DNN training as illustrated in Fig. 4.3. Since CE training of DNN attempts to minimize the Kullback-Leibler (KL) divergence between the distributions of ground truth labels (which are noisy for PTs) and DNN posterior outputs, the posteriors simply learn the noisy distribution of the PTs. This degrades the performance of the DNN, sometimes even worse than a GMM-HMM system. This is shown from the experiments in Section 4.5. There are two reasons for this.

- First, discriminative training is more sensitive to the accuracy of labels compared to ML training [53].

- Second, DNNs do not generalize well if the training and test data are generated from two different distributions. In [54], this was shown for the case when a DNN was trained using wideband data but tested on narrowband data. In the current context, the distributions are different during training and testing. During training, we train the network to match the PT distributed labels. However, during testing, we expect the network outputs to generate DT distributed labels.

4.3 Weakly Supervised Learning

In this section, we explore methods by which we are able to train DNNs which perform consistently better than GMM-HMM systems. In particular, we explore multi-task learning (MTL) methods [55, 56].

4.3.1 Multi-Task Learning (MTL)

To take advantage of the PTs while at the same time alleviating the effect of noisy labels, we explore the MTL approach. Here, multiple related tasks are trained together with all tasks having a set of shared hidden layers (SHLs). However, each task has its own softmax layer which is trained using the labels for that particular task. The first softmax layer is trained using PTs of the target language whereas the second layer is trained on DTs of all the available source languages. This is the MTL system as illustrated in Fig. 4.4.

There could be a third softmax layer trained using self-training transcripts (ST). A self-trained system was earlier described in Section 4.2. In the absence of supervised data, the STs can be used as "ground truth" labels to retrain a well-trained ASR system. Thus, the ASR system retrains itself using its own predictions.

For all the MTLs, during test time, only the PT softmax layer is retained for decoding while discarding the other softmax layers.

Now, we define the objective function of the MTL framework. The objective function for the MTL system illustrated in Fig. 4.4 is

$$\mathcal{L}(\mathbf{W}) = \mathcal{L}_{\text{CE-PT}}(\mathbf{W}) + \lambda_{\text{DT}}\mathcal{L}_{\text{CE-DT}}(\mathbf{W}), \tag{4.1}$$

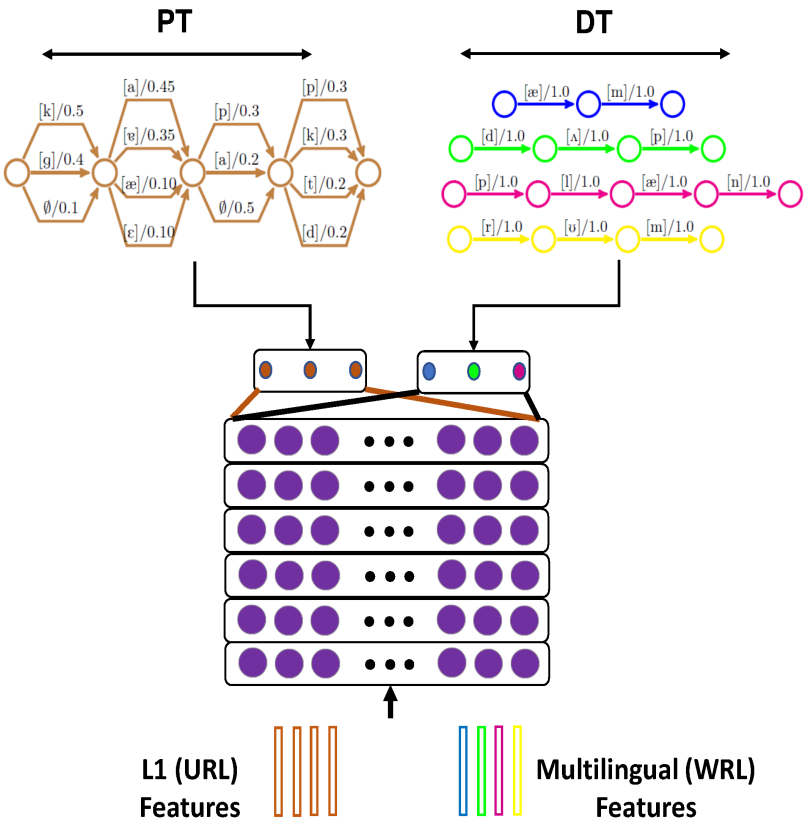

Figure 4.4: MTL DNN trained with PTs and DTs.

where

$$\mathcal{L}_{\text{CE-PT}}(\mathbf{W}) = \sum_t \sum_k p_t^{(1)}(k) \log y_t^{(1)}(k), \tag{4.2}$$

$$\mathcal{L}_{\text{CE-DT}}(\mathbf{W}) = \sum_t \sum_k d_t^{(2)}(k) \log y_t^{(2)}(k). \tag{4.3}$$

The term $p_t^{(1)}(k) \in [0, 1]$ is the ground truth probability of senone k provided by the PTs in the target language (URL). The term $d_t^{(2)}(k) \in \{0, 1\}$ is the ground truth probability of senone k provided by the DTs in the source languages (WRL). The term $y_t^{(l)}(k)$ is the output of the softmax node k in the l^{th} task of the MTL system. The weights of the neural network are represented by $\mathbf{W}$.

Why does MTL work? Our motivation for using multiple softmax layers stems from encouraging results obtained in previous studies for multilingual training [57], [58], [36] and MTL [56]. In [57], Scanzio *et al.* were the first to propose the multiple softmax architecture for training an artifical neural network (ANN) simultaneously using multilingual data. Later, it was used in [58], [36] for multilingual training and in [56] for MTL. Their ANN was used as a front-end discriminant features generator that were later used to train language dependent hidden Markov model (HMM) based speech recognizer. In [58], this approach was further extended by Vesely *et al.* by including a bottleneck intermediate layer. They showed that such bottleneck features generalize well even in mismatched cross-lingual settings, i.e., when training languages are different from the test language. In [59], Tuske *et al.* propagated this idea further by concatenating such bottleneck features with mel-frequency cepstral coefficients (MFCCs) in mismatched cross-lingual scenarios. Furthermore, it was established in [36] that the SHLs trained using multiple softmax layers over multilingual data outperform monolingual SHLs. In [60], it was used for semi-supervised training. The key advantages offered by training DNNs with multiple softmax layers are:

- Improved SHLs.
- SHLs are language independent provided the amount of training data is uniform across all languages. Thus, the SHLs represent a "global" view of the multilingual data.
- Each softmax layer may be fine-tuned to a specific language thus making it language dependent. Thus, the softmax layer represents a "local view" of the multilingual data.

- It obviates the need to explicitly map the phones to a common phone set.

In this work, our conjecture is that simultaneous training of PTs along with DTs offers multiple advantages.

- First, the *spurious* or incorrect error gradients backpropagated by the noisy PT labels are partially corrected by the *true* or correct error gradients backpropagated by the high quality DT labels. Therefore, due to strong supervision of highly reliable DT labels, the gradients become less noisy. The net result is an improved non-linear transformation learned by the SHLs and hence better feature separation. This advantage is clearly lost with the single softmax Vanilla DNNsystem trained using PTs since the training steps are sequential in nature - first train using multilingual DTs and then fine-tune using monolingual PTs. The noise introduced by PTs in the second stage cannot be corrected.

- Since the output nodes of the DNN have one-to-one correspondence with a multilingual senone decision tree, the output nodes of each softmax layer represent multilingual senones. By exclusively training the PTs in the first softmax layer, we train only those softmax weights which are connected to the nodes representing the senones in the target language. The weights for the other senones remain untrained. This is expected to reduce the entropy of the output activation vectors. In addition, if the quality of the PTs improves, it will further lead to improved softmax weights.

- Unlike [57] where each language was assigned its own softmax layer, we assign all source languages with DTs to only one softmax layer since the primary role of DTs is to fix SHLs. This reduces the complexity of the network structure.

4.3.2 Knowledge Distillation (KD)

In this section, we make improvements over the MTL system, described in Section 4.3.1, using knowledge distillation (KD).

We provide a brief overview of KD first before describing the framework in detail. In [61], the authors describe KD as the process of transferring knowledge from a large cumbersome model (or an ensemble of models) to a small distilled

model. The cumbersome and distilled model are sometimes referred to as the teacher and student models. Hence, KD is also known as teacher-student (TS) learning. In [62], KD was used for reducing the size of deep networks.

If $\mathcal{D}$ is a data set on which the student model is to be trained, then the DNN training procedure involves the following steps.

- In the first step, feedforward $\mathcal{D}$ through a prior well-trained teacher DNN to generate the posterior outputs (or teacher labels). The teacher labels form a soft target distribution for each training example in $\mathcal{D}$.

- In the second step, train the student DNN by minimizing the cross-entropy (CE) loss between the teacher labels and posterior outputs of the student DNN. Thus, the student DNN attempts to mimic the behavior of the teacher DNN by trying to match its own outputs with those of the teacher labels.

To improve the generalizability of the student DNN, the teacher labels could be generated by using a high temperature T in the softmax of the teacher DNN. The same temperature T is then used at the softmax of the student DNN during CE training. It can be shown that when $T \rightarrow \infty$ (high temperature limit), CE training is equivalent to minimizing the mean square error (MSE) of the logits (pre-softmax activations) between the teacher and student DNNs [61].

Several studies [63–73] in the past have used KD to improve DNNs. In [63], a small DNN was trained using teacher labels generated by feedforwarding a large number of untranscribed data through a large DNN. In other studies, the authors transfer the knowledge from a large RNN to a small DNN [64] or from a large DNN to a small highway DNN [65]. In [66, 67], KD was used to improve robustness of DNNs to noisy data. The one that is most relevant to our work is in [68] where KD was used for adaptation to under-resourced Japanese dialects and children's speech.

Now, we explain the KD framework in detail. Consider an input feature vector $\mathbf{x}$. A generalized softmax is a softmax function operating on logits $z_k(\mathbf{x})$ and a temperature $T \in \mathbb{R}^+$. Here, $k \in \{1, \cdots, K\}$, where K is the total number of labels. We will denote $z_k(\mathbf{x})$ as simply z_k and assume the dependence on $\mathbf{x}$ is implicit. The output $y_k(T)$ of the generalized softmax is given by

$$y_k(T) = \frac{\exp(z_k/T)}{\sum_{j=1}^{K} \exp(z_j/T)}. \tag{4.4}$$

There are two extreme cases for Eq. (4.4). Let $\mathbf{y}(T) = [y_1(T) \cdots y_K(T)]'$. For very hot and cold temperatures, $\mathbf{y}(T)$ approaches the uniform and 1-hot distribution respectively. Thus, $\lim_{T\to\infty} y_k(T) = \frac{1}{K}$ and $\lim_{T\to 0} y_k(T) = \mathbb{1}_{[k=\arg\max_{1\le j\le K} y_j]}$. In the KD framework, the student model is trained to minimize the loss,

$$E_{\text{KD}} = \rho C(\mathbf{p}, \mathbf{y}(1)) + (1-\rho) C(\mathbf{q}(T), \mathbf{y}(T)), \tag{4.5}$$

where

$$C(\mathbf{p}, \mathbf{y}(1)) = -\sum_{k=1}^{K} p_k \log y_k(1), \tag{4.6}$$

$$C(\mathbf{q}(T), \mathbf{y}(T)) = -\sum_{k=1}^{K} q_k(T) \log y_k(T). \tag{4.7}$$

The term p_k is the posterior probability of label k given the feature vector $\mathbf{x}$. Since this is generated from the PTs, it need not be a binary value 0 or 1. Thus, $\mathbf{p}$ need not be a 1-hot vector. Likewise, $q_k(T)$ is the posterior probability of label k generated by feedforwarding $\mathbf{x}$ through a teacher DNN equipped with a generalized softmax with temperature T. In other words, it is a teacher label. In the under-resourced scenario, the teacher DNN is a multilingual DNN trained with DTs from WRLs. The term $y_k(T)$ is the posterior probability of label k generated by feedforwarding $\mathbf{x}$ through a student DNN equipped with a generalized softmax with temperature T as in Eq. (4.4). The student DNN is the target language DNN to be trained with PTs from the URL. The outputs $y_k(1)$ of the student DNN in Eq. (4.6) are constrained to a temperature of one whereas in Eq. (4.7) the temperature can be any $T \in \mathbb{R}^+$. Finally, ρ is a weight that balances the losses in Eq. (4.6) and Eq. (4.7).

During backpropagation, the gradient of Eq. (4.7) with respect to the student logit z_k, i.e., $\frac{\partial C(\mathbf{q},\mathbf{y})}{\partial z_k}$, is artificially scaled by T^2. This is because the gradient itself is a function of $1/T^2$. Thus, the artificial scaling removes the dependence on T. As a result, the individual backpropagation errors from Eq. (4.6) and Eq. (4.7) have similar scales and can be added meaningfully.

Knowledge distillation specializes to several interesting cases.

- When $\rho = 1$, Eq. (4.5) is same as the standard CE loss.
- When $0 < \rho < 1$ and $T = 1$, Eq. (4.5) is equivalent to regularizing the CE loss with Kullback-Leibler divergence (KLD) [74].

- When $\rho = 0$ (indicating the absence of ground truth labels), Eq. (4.5) can be used for unsupervised adaptation. For example, in the case of $\rho = 0$, $T = 1$ and when the student DNN is *not* initialized from a teacher DNN, Eq. (4.5) was used for unsupervised adaptation using the teacher labels obtained from a large teacher DNN [63].

- When $\rho = 0$, $T = 1$ and the student DNN is initialized from the teacher DNN, training using Eq. (4.5) is equivalent to self-training. Here, the teacher labels $\mathbf{q}(1)$ are identical to the outputs $\mathbf{y}(1)$ of the student DNN only before training begins. However, once training begins, the teacher labels are kept constant whereas the student outputs are allowed to change with every weight update.

4.3.3 Target Interpolation (TI)

In this section, we make improvements over the MTL system, described in Section 4.3.1, using target interpolation. The key idea here is that we interpolate the confidences of the labels provided by PTs with the confidences of the target language DNN. The DNN is then trained using the new interpolated confidence values. Intuitively, we emphasize the beliefs of the learner rather than completely relying on noisy "ground truth" labels.

Now, we explain the TI framework in detail. We will omit the dependence on T since in this section $T = 1$ always. First, we define $C(f(\mathbf{y}), \mathbf{y})$ as

$$C(f(\mathbf{y}), \mathbf{y}) = -\sum_{k=1}^{K} f(y_k) \log y_k, \tag{4.8}$$

where $f(.)$ is an element-wise function of $\mathbf{y}$ such that $f(y_k) \in [0, 1]$ and $\sum_k f(y_k) = 1$. The DNN is trained to minimize the loss,

$$\begin{aligned} E &= \rho C(\mathbf{p}, \mathbf{y}) + (1 - \rho) C(f(\mathbf{y}), \mathbf{y}), \\ &= C(\rho \mathbf{p} + (1 - \rho) f(\mathbf{y}), \mathbf{y}), \end{aligned} \tag{4.9}$$

where $C(\mathbf{p}, \mathbf{y})$ is as defined in Eq. (4.6). The second step in Eq.(4.9) is due to the linearity of $C(.,.)$ in the first argument. We consider two among several choices

of $f(.)$. They are

$$f(y_k) = \begin{cases} y_k, \text{ (soft)} \\ \mathbb{1}_{[k=\arg\max_{1\leq j\leq K} y_j]}. \text{ (hard)} \end{cases} \tag{4.10}$$

The loss functions corresponding to these choices are

$$E_{\text{soft}} = -\sum_{k=1}^{K} (\rho p_k + (1-\rho) y_k) \log y_k, \tag{4.11}$$

$$E_{\text{hard}} = -\sum_{k=1}^{K} (\rho p_k + (1-\rho) \mathbb{1}_{[k=\arg\max_{1\leq j\leq K} y_j]}) \log y_k. \tag{4.12}$$

And the error gradients are

$$\frac{\partial E_{\text{soft}}}{\partial z_k} = \rho(y_k - p_k) + (1-\rho) y_k (I(y_k) - H(\mathbf{y})), \tag{4.13}$$

$$\frac{\partial E_{\text{hard}}}{\partial z_k} = \rho(y_k - p_k) + (1-\rho)(y_k - \mathbb{1}_{[k=\arg\max_{1\leq j\leq K} y_j]}), \tag{4.14}$$

where

$$I(y_k) = -\log y_k,$$

$$H(\mathbf{y}) = -\sum_{k=1}^{K} y_k \log y_k.$$

The motivation behind the choices in Eq. (4.10) is that we use the label confidences of the DNN instead of completely relying on the noisy PT labels. Hence, we modify the PT confidence p_k with a new confidence which is an interpolation between p_k and $f(y_k)$. For the soft case, we use the entire output distribution of the DNN. Then the loss in Eq. (4.11) becomes the standard CE loss with entropy regularization. A DNN trained using this loss function will find a balance between minimizing the CE loss $C(\mathbf{p}, \mathbf{y})$ and lowering the entropy of its outputs $C(\mathbf{y}, \mathbf{y})$. Since PTs are prone to high entropies, lowering the entropies of the DNN outputs is desirable. For the hard case, we simply binarize the DNN outputs to a 1-hot distribution. Compared to the soft case, the hard case ignores the cross-correlations between different classes. In both cases, the new interpolated confidences still form a valid probability distribution since they sum to one when summed over the K labels.

4.4 Semi-Supervised Learning

In this section, we make improvements over the MTL system, described in Section 4.3.1, using semi-supervised learning. The key idea here is that we make use of unlabeled data, along with labeled data, while training the MTL to discover additional useful hidden layer representations.

4.4.1 Deep Auto-Encoder

We use $(\mathbf{x}, \mathbf{y})$ to denote labeled examples and $\mathbf{x}$ to denote unlabeled examples. In the semi-supervised learning paradigm, both unlabeled examples drawn from $P(\mathbf{x})$ and labeled examples drawn from $P(\mathbf{x}, \mathbf{y})$ are used to learn the conditional distribution $P(\mathbf{y}|\mathbf{x})$. The advantage of using unlabeled examples is that they can learn a representation $\mathbf{h} = f(\mathbf{x}, \mathbf{W})$ which can help group similar classes together. This, in turn, can improve the predictions made by $P(\mathbf{y}|\mathbf{x})$.

The MTL framework has the advantage that it unifies both the generative $P(\mathbf{x})$ and the discriminative $P(\mathbf{y}|\mathbf{x})$ models together. To see how this is possible, assume $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \Sigma)$. Thus,

$$P(\mathbf{x}) \propto \exp\Big\{ -\frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \Big\}, \tag{4.15}$$

where

$$\hat{\mathbf{x}} = g(\mathbf{x}, \mathbf{W}),$$
$$\Sigma = \mathrm{I} \quad \text{(identity covariance matrix)}.$$

Maximing $\log P(\mathbf{x})$ is equivalent to minimizing the mean square error (MSE) $||\mathbf{x} - \hat{\mathbf{x}}||^2$. Thus, if the MTL is designed to predict $\hat{\mathbf{x}}$ at the output of one of its sub-tasks, it will try to match the output $\hat{\mathbf{x}}$ with the input $\mathbf{x}$. In this sense, the MTL acts as a deep auto-encoder (DAE). The other sub-tasks of the MTL can still model $P(\mathbf{y}|\mathbf{x})$ as a multinomial distribution when trained using the CE criterion. An illustration of this framework is shown in Fig. 4.5.

The first task is trained using PTs, the second task is trained using DTs, and the third task is trained using raw features. Since the backpropagated errors generated from training PTs are noisy, the DAE can help fix these errors and thus discover more useful hidden layer representations. The MTL is trained to minimize the

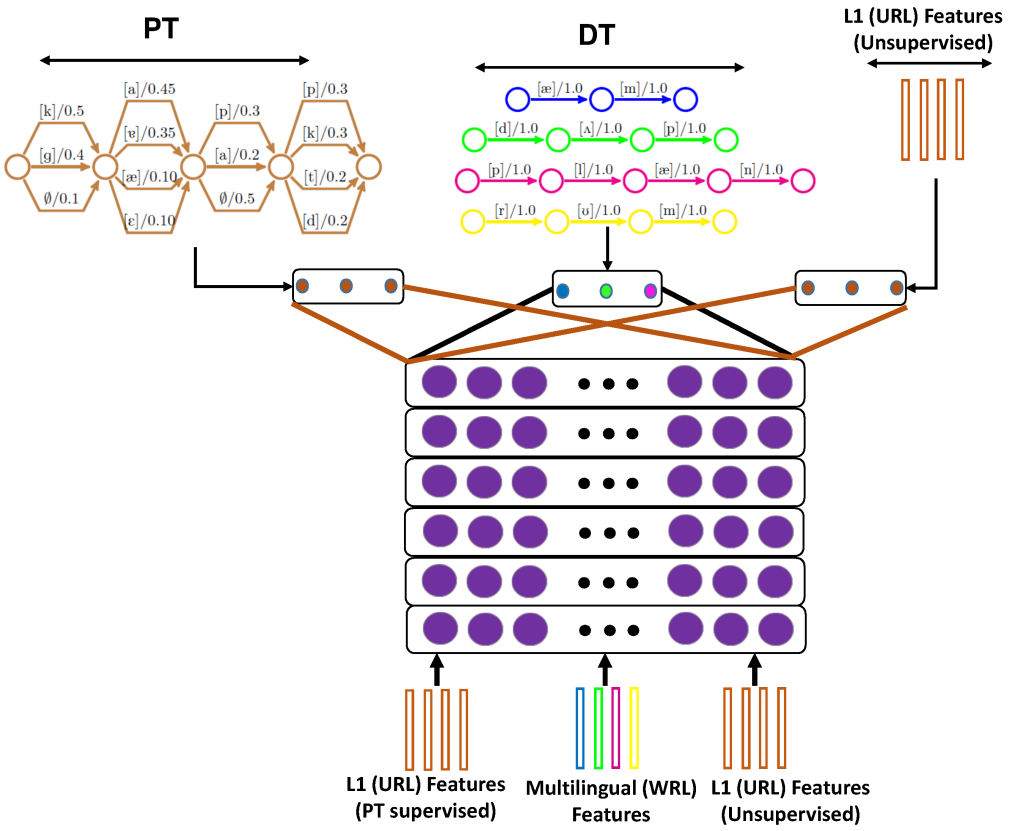

Figure 4.5: MTL using deep auto-encoder.

following loss:

$$\mathcal{L}(\mathbf{W}) = \mathcal{L}_{\text{CE-PT}}(\mathbf{W}) + \lambda_{\text{DT}}\mathcal{L}_{\text{CE-DT}}(\mathbf{W}) + \lambda_{\text{DAE}}\mathcal{L}_{\text{DAE}}(\mathbf{W}), \tag{4.16}$$

where $\mathcal{L}_{\text{CE-PT}}(\mathbf{W})$ and $\mathcal{L}_{\text{CE-DT}}(\mathbf{W})$ are the CE losses at the first and second subtasks of the MTL respectively. They are represented by Eq. (4.2) and Eq. (4.3). The final loss $\mathcal{L}_{\text{DAE}}(\mathbf{W})$ is the MSE loss and is given by

$$\mathcal{L}_{\text{DAE}}(\mathbf{W}) = \sum_{t} ||\mathbf{x}_t - \hat{\mathbf{x}}_t||^2 . \tag{4.17}$$

Auto-encoders have been previously used for noise reduction using single-layer networks in [75] and deeper recurrent layers in [76]. The denoising auto-encoder is basically a single-layer neural network which attempts to reconstruct a clean version of its own noisy input. In [76], this idea was further extend to deeper layers as deep denoising auto-encoder. By incorporating the RNN structure, it becomes a deep recurrent denoising auto-encoder.

More recently, deep denoising auto-encoders in the MTL framework have been used in the problem of far-field speech recognition [77]. In the far-field scenario, speech signals captured by distant microphones located far away from speakers are susceptible to dereveberation and additive noise. In contrast, time synchronized speech signals captured by close-talk microphones are relatively clean. An auto-encoder was used to learn the mapping between the noisy signals of distant microphones and the relatively clean signals of close-talk microphones. Since the primary objective is to improve the classification performance of an ASR system, the authors in [77] integrate the auto-encoder into an MTL framework. Thus, the unified network optimizes two tasks simultaneously - the denoising task and the recognition task.

Auto-encoders have also been used to generate bottleneck features in under-resourced scenarios when little training data are available. For example, in [78], the authors train a stack of deep auto-encoders (DAEs) in a layer-wise unsupervised manner to predict clean speech from artificially corrupted noisy speech. Then a bottleneck layer, an additional hidden layer, and a final softmax layer are added to the stack of DAEs before fine-tuning the entire network using backpropagation.

More relevant to our work is the study in [79]. The authors train a neural network to recognize digits from inaccurately labeled images in the MNIST dataset.

To incorporate a notion of perceptual consistency in the training, they train an auto-encoder in parallel to promote top-down consistency of model predictions with the observations. This allowed the model to discover the noise pattern in the data.

4.5 Results

In this section, we present the results for the methods described in Section 4.3 and Section 4.4.

4.5.1 Data

- **Corpus:** Multilingual audio files were obtained from the Special Broadcasting Service (SBS) network which publishes multilingual radio podcasts in Australia. These data include over 1000 hours of speech in 68 languages. The following languages were used in our experiments: Swahili (swh), Amharic (amh), Dinka (din), Mandarin (cmn), Arabic (arb), Cantonese (yue), and Hungarian (hun). The corpus is summarized in Table 4.1. The podcasts were not entirely homogeneous in the target language and contain utterances interspersed with segments of music and English words. An HMM-based language identification system was used to isolate regions that correspond mostly to the target language. These long segments were then split into smaller 5-second utterances.

Table 4.1: SBS multilingual corpus.

Language	Utterances		Phones
	Train	Test	
Swahili (swh)	462	123	48
Amharic (amh)	516	127	37
Dinka (din)	248	53	27
Mandarin (cmn)	467	113	52
Arabic (arb)	468	112	46
Cantonese (yue)	544	148	32
Hungarian (hun)	459	117	65
All	-	-	82

- **Turkers:** More than 2500 Turkers participated in transcribing, with roughly 30% of them claiming to know only English. The remaining Turkers claimed knowing other languages such as Spanish, French, German, Japanese, and Mandarin. The utterances were limited to a length of 5 seconds. This is because the Turkers did not understand the utterance language and it was easier for them to annotate short utterances than long. Since English was the most common language among the Turkers, they were asked to annotate the sounds using English letters. The sequence of letters was not meant to be meaningful English words or sentences since this would be detrimental to the final performance. The important criterion was that the annotated letters represent sounds they heard from the utterances as if they were listening to a sequence of nonsense syllables in some exotic language. Since no Turker is likely to generate the perfect transcript, each utterance was transcribed by 10 Turkers creating 10 different transcripts per utterance. These transcripts were converted to phones and merged into a PT using [24]. Turkers were typically paid $500 per 10 Turkers for transcribing an hour of audio.

- **PTs and DTs:** PTs, worth about 1 hour of audio in the target language, were collected from Turkers. The same audio files were presented to native transcribers to generate DTs. However, the DTs in the target language were never used for training ASR systems. These were used only for benchmarking oracle error rates. The oracle scenario is the ideal scenario where one would have access to DTs in the target language. PTs in the target language (URL) and DTs in the source languages (WRL) were used for training ASR systems. We are now ready to outline the amounts of training data that were used for training an ASR in the target language (URL) L.

  1. PTs in L: PTs, from about 40 minutes of audio in L, were collected from Turkers who did not speak L.
  2. Zero DTs in L: No DTs in L were used for training an ASR in L.
  3. DTs only in source languages: DTs from six other source languages ($\neq L$) were used for training an ASR in L. About 40 minutes of DTs were used per source language. Hence, the total amount of DTs available for training was $\approx$ 4 hours (40 minutes/language $\times$ 6 languages).
  4. Unsupervised data: There were about 5.5 hours of unsupervised data in L.

  The development and test sets were worth 10 minutes each. The test utterances

were sufficiently shuffled so as to avoid biasing to a subset of speakers or to a specific gender. As an example, consider Table 4.2 where *swh* is the target language to be recognized. Then the training set consists of 40 minutes of PTs in *swh* and 40 minutes of DTs in each of *amh*, *din*, *cmn*, *arb*, *yue*, and *hun* combined.

Table 4.2: Training set when Swahili (*swh*) is target language.

Language	Transcript Type	Size
Swahili (swh)	PT	40 min
Amharic (amh)	DT	40 min
Dinka (din)	DT	40 min
Mandarin (cmn)	DT	40 min
Arabic (arb)	DT	40 min
Cantonese (yue)	DT	40 min
Hungarian (hun)	DT	40 min
Swahili (swh)	-	5.5 hrs

- **Universal Phone Set:** The orthographic transcriptions for the PTs and DTs were converted to IPA based phone transcriptions. The canonical pronunciation was derived from a lexicon. If a lexicon was not available, a language-specific G2P model was used. To form a set of multilingual phone symbols, diphthongs/triphthongs were split into two/three individual phone symbols unless they were the same as English diphthongs. Diacritics such as tones and stress markers tend to make the phone symbols unique to a particular language. Therefore, to enable phone merging across languages, such language specific diacritics were removed from the canonical phone transcriptions.

  This was followed by merging the phones to a reduced phone set. If an IPA phone symbol was unique in the sense that it appeared in the phone transcriptions of only one language, then that symbol was merged with another symbol which differs in only one distinctive feature. Repeating this process several times guarantees that each phone is represented in at least two languages. This enables sharing data across languages. The merged phone set is the multilingual or universal phone set. The total number of phones in the multilingual set (i.e., all languages) was 82 which excludes the silence phone. An individual breakdown of phones per language is outlined in Table 4.1.

- **Language Models:** Finally, phone based language models (LMs) were built from the text in the target language mined from Wikipedia. To measure the

performance of the ASR systems, phone error rate (PER) was used as the evaluation metric. All experiments were conducted using the Kaldi toolkit [48]. Kaldi source code in C++ and toy examples of the proposed models are available in our github repository.[1]

4.5.2 Features

Thirteen dimensional MFCCs, spliced with +/- 3 neighboring frames, were extracted from speech utterances. These were then transformed using a LDA transform followed by feature-space maximum likelihood linear regression (fMLLR) transform resulting in 40-dimensional fMLLR features. These features were kept low dimensional to avoid the curse-of-dimensionality problem which is more likely to occur in under-resourced scenarios. These features were then mean normalized using cepstral mean normalization (CMN) before using them for DNN training.

4.5.3 Baselines and Proposed Models

The following baselines were used in our evaluation:

- Monolingual GMM-HMM and DNN (Section 4.5.4): These models were trained using DTs in the target language. This is the oracle scenario if we assume DTs were to be available in the target language. The oracle scenario was used only for benchmarking performance against other models. For DNN training, the DNN was first initialized using RBM pre-training[2] [80] using unlabeled data from source languages. Following this, a softmax layer was added on top of the SHLs and the DNN was fine-tuned using CE training with DTs in the target language.

- Multilingual GMM-HMM and DNN (Section 4.5.5): These models never used any DTs or PTs in the target language. They were trained after pooling the DTs from all the source languages. Hence, these are multilingual ASRs. After RBM pre-training using source languages, the DNN was fine-tuned using CE training with DTs from the source languages as targets. Since

[1] `git clone -b teacher-student https://github.com/irrawaddy28/SBS-kaldi-2015`

[2] All DNNs considered in this study were initialized using RBM pre-training using 6 shared hidden layers (SHLs) with 1024 nodes per layer.

target language data were not used during training, these models are not adapted to the target language.

- Self-training DNN (Section 4.5.6): This is the model that was described in Section 4.2. This is an adapted model since a multilingual DNN was used to decode the utterances in the target language and then the decoded labels were used as the new targets for another round of training.

- MAP GMM-HMM (Section 4.5.7): This is the GMM-HMM model MAP adapted to the target language using PTs in the target language.

- Vanilla DNN (Section 4.5.8): This is the DNN model adapted to the target language using PTs in the target language. After RBM pre-training, a single softmax layer was added and fine-tuned using PTs in the target language.

The following are the proposed methods that were used in our evaluation:

- MTL-CE (Sections 4.5.9): This is the MTL model, from Section 4.3.1, adapted to the target language. It has two softmax layers. The first softmax layer was fine-tuned using PTs in the target language. The second softmax layer was fine-tuned using DTs in source languages. Both the tasks were trained to minimize the CE loss. The MTL model never used any DTs in the target language.

- MTL-KD (Section 4.5.10): This is the MTL model, from Section 4.3.2, adapted to the target language. The first task of the MTL model was trained to minimize the loss E_{KD} in Eq. (4.5) with $0 < \rho < 1$ and $T \geq 1$. Specifically, values of $\rho \in \{0.2, 0.4, 0.6, 0.8\}$ were used. For the special case of $T = 1$, Eq. (4.5) becomes KLD regularization [74].

- MTL-TI (Section 4.5.11): This is the MTL model, from Section 4.3.3, adapted to the target language. The first task of the MTL model was trained to minimize the loss E_{soft} in Eq. (4.11) and E_{hard} in Eq. (4.12) while using $\rho \in \{0.2, 0.4, 0.6, 0.8\}$.

- MTL-DAE (Sections 4.5.12): This is the semi-supervised MTL model, from Section 4.4.1, adapted to the target language. It is trained using PTs and unlabeled data in the target language along with DTs in source languages.

4.5.4 Monolingual GMM-HMM and DNN

In the first baseline, monolingual GMM-HMM and DNN models were trained and tested using DTs in the target language. This is the oracle scenario if we assume DTs were to be available in the target language.

The monolingual PERs over a total of about 7K-8K phones are given in Table 4.3. This gives us an estimate about the approximate best case (lower bound) PERs.

Table 4.3: PERs of monolingual GMM-HMM and DNN models. Dev set in parentheses.

Lang	**PER** (%)	
	GMM-HMM	**DNN**
swh	35.13 (45.78)	34.25 (39.64)
amh	51.90 (48.68)	46.69 (44.07)
din	51.56 (47.03)	48.37 (48.00)
cmn	31.80 (26.14)	28.26 (25.16)

4.5.5 Multilingual GMM-HMM and DNN

In this experiment, we assume DTs in the target language are not available during training. However, DTs of the source languages are still available. Thus, DTs from the 6 source languages were pooled together to train multilingual GMM-HMMs and multilingual DNNs. Decision tree clustering of the multilingual data resulted in about 1000 senones. The multilingual DNNs were trained using 6 hidden layers with 1024 nodes per layer and a final softmax layer with about 1000 output nodes representing the senones.

Table 4.4: PERs of multilingual GMM-HMM and DNN models. Dev set in parentheses.

Lang	PER (%)		
	GMM-HMM	**DNN**	# **Senones**
swh	63.02 (66.00)	60.40 (61.62)	950
amh	68.65 (68.47)	65.56 (64.82)	1008
din	67.93 (66.79)	63.81 (65.44)	1012
cmn	69.55 (67.08)	59.50 (59.50)	985

The PERs are given in Table 4.4. Expectedly, due to lack of DTs in the target language, the PERs are much higher than the oracle case in Table 4.3. Hence,

the PERs in Table 4.4 establish the worst case (upper bound) PERs. In all subsequent experiments, we start from the worst case PERs in Table 4.4 and attempt to approach the best case PERs in Table 4.3 by including PTs during training.

4.5.6 Self-Training DNN

In this experiment, we use a self-training algorithm [51] in which a multilingual DNN decodes the training utterances in the target language and then uses the confidence selected decoded labels to retrain itself in the target language [52]. Self-training is an unsupervised adaptation method.

The objective of this experiment is to evaluate the efficacy of ASR generated labels which in this case are self-training transcripts (STs). Since the multilingual DNN is not trained using the target language DTs, the decoded labels are very likely to be unreliable. Hence, we use only a subset of frames, selected by first evaluating the frame level confidences [51]. The frame confidences are simply the values of the posteriors of the best path in the decoding lattice generated as the output of the multilingual DNN. In the second step, an empirically determined threshold is chosen and compared with the frame confidences. Any frame whose confidence is above the threshold is selected for training. Otherwise, it is discarded.

Table 4.5: PERs of self-trained DNN models. Dev set in parentheses.

Lang	**PER** % **DNN**
swh	58.12 (60.36)
amh	64.16 (63.95)
din	63.13 (62.26)
cmn	57.37 (57.73)

The results are given in Table 4.5. Compared to the multilingual DNN in Table 4.4, the improvement due to self-training is in the range 0.68%-2.28%. We determined frame confidence thresholds as 0.5 or 0.6 from the development set.

4.5.7 Maximum A Posteriori GMM-HMM (MAP GMM-HMM)

In this experiment, the multilingual GMM-HMM model in Section 4.5.5 is adapted using the PTs of the target language. The multilingual GMM-HMM acoustic

model is adapted using MAP adaptation described in more detail in [15, 50]. The main component in this step is that the ASR search graph, represented as a WFST mapping from the acoustic signal to a sentence, is defined by the composition $H \circ C \circ L \circ G \circ PT$ instead of the usual $H \circ C \circ L \circ G$. Here, PT is the confusion network of phones obtained from crowd workers as was described in Section 2.1.3.

The PER results for the MAP adapted GMM are under the column MAP GMM-HMM in Table 4.6. The PER results for the multilingual DNN (column MULTI-DNN in Table 4.6) are replicated from Table 4.4 for purposes of comparison. The absolute improvement in PERs as a result of adapting using PTs is in the range 3.12%-15.08%. This is much better than the improvement obtained using ASR labels (STs) in the previous section. Since the MAP GMM-HMM models significantly outperform the self-trained DNNs, we use MAP GMM-HMM as the starting baseline within the class of PT adapted models.

4.5.8 Vanilla DNN

In this experiment, we follow the conventional procedure of adapting a multilingual DNN to the target language. In the first step, the softmax layer of the multilingual DNN (Section 4.5.5) is replaced by a randomly initialized softmax layer while retaining the SHLs [38]. The resulting DNN is then fine-tuned using the forced alignments generated by the MAP adapted model (Section 4.5.7). This is the conventional way to adapt a DNN using DTs [37]. However, this approach does not work very well for PTs largely due to the presence of incorrect labels in PTs [16]. The results are shown under the column Vanilla DNNin Table 4.6. Clearly, the performance of Vanilla DNNis worse than MAP GMM-HMM for Swahili and Dinka and only marginally better for Amharic. On an average, the Vanilla DNN marginally outperforms the MAP GMM-HMM by only about 0.22% absolute.

4.5.9 Multi-Task Learning With Cross Entropy (MTL-CE)

In this experiment, instead of using a single softmax layer, we use two separate softmax layers (one per task) as illustrated in Fig. 4.4. The first task is trained with PTs in the target language, whereas the second task is trained with DTs in the source languages. In addition, we found data augmentation beneficial for training.

That is, we introduced two additional copies of the input data to the first task. The results are shown under the column MTL-CE in Table 4.6. The absolute decrease in PER compared to the Vanilla DNN is consistent across all languages and is in the range 1.00%-1.5%. Comparing the PT adapted MTL-CE with the unadapted MULTI-DNN, the absolute decrease in PER is in the range 4.77%-15.51%.

Table 4.6: PERs of multilingual DNN (MULTI-DNN), MAP GMM-HMM, Vanilla DNN, MTL-CE models. The number in the parentheses is the absolute improvement in PER over MULTI-DNN. Best PER for each is language highlighted in bold.

Lang	**PER (%)**			
	Unadapted	**PT Adapted**		
	MULTI-DNN	**MAP GMM-HMM**	**Vanilla DNN**	**MTL-CE**
swh	60.40 (0.0)	45.32 (15.08)	45.89 (14.51)	**44.89 (15.51)**
amh	65.56 (0.0)	61.98 (3.58)	61.72 (3.84)	**60.79 (4.77)**
din	63.81 (0.0)	59.48 (4.33)	59.64 (4.17)	**58.65 (5.16)**
cmn	59.50 (0.0)	56.38 (3.12)	55.03 (4.47)	**53.53 (5.97)**

4.5.10 Multi-Task Learning With Knowledge Distillation (MTL-KD)

In this experiment, we train the MTL model using KD. The PERs comparing the MTL models trained with CE, KLD, and KD losses are outlined in Table 4.7. We highlight only the most interesting cases with ρ in the range $0.6 - 0.2$ and $T = 2$. From Table 4.7, it is clear that the KD models outperform the baseline CE and KLD models.

Now, we analyze the effect of T and ρ on recognition rates. Keeping ρ fixed and varying T is equivalent to comparing KLD with KD models. Thus, as T increases (keeping ρ constant), KD models outperform their KLD counterparts most of the time. Increasing T makes the class correlations more pronounced. This indicates that the temperature parameter improves the generalization capacity of the DNNs by avoiding tuning to the noisy PTs. Next, keeping $T > 1$ fixed and varying ρ is equivalent to comparing within the family of KD models. As ρ decreases, the PERs tend to decrease first and then increase. Desirable values of ρ are $\rho < 0.5$. This implies that the performance improves when the model relies increasingly on the teacher labels rather than the PT labels. However, this trend reverses for very low values of ρ. For example, in the extreme case when $\rho = 0$ (completely

Table 4.7: PERs of different MTL models trained with CE, KLD, and KD losses. The parameters ρ and T are the weighting and temperature parameters in Eq. (4.5). Best PER for each is language highlighted in bold.

Model	Parameters		Language			
	ρ	T	swh	amh	din	cmn
MTL-CE	1	-	44.89	60.79	58.65	53.53
MTL-KLD	0.6	1	44.11	59.97	58.19	51.00
MTL-KLD	0.4	1	44.21	59.36	58.33	50.29
MTL-KLD	0.2	1	44.63	59.55	58.65	50.93
MTL-KD	0.6	2	44.12	59.82	58.15	50.93
MTL-KD	0.4	2	**43.66**	59.40	**57.97**	49.85
MTL-KD	0.2	2	44.40	**59.08**	58.26	**49.38**

ignoring PT labels), we noticed exceedingly high PERs above 85%. This proves that PT labels are still useful.

4.5.11 Multi-Task Learning With Target Interpolation (MTL-TI)

In this experiment, we train the MTL model using TI. The PERs comparing the CE and TI models are outlined in Table 4.8. Again, we highlight only the most interesting cases of ρ $(0.6 - 0.2)$. Clearly, both variants of TI models outperform the CE model. Among the TI models, TI (Soft) outperforms TI (Hard) for the African languages (Swahili, Amharic, and Dinka). For Mandarin, TI (Hard) outperforms TI (Soft) by a small margin. Surprisingly, for both TI (Hard) and TI (Soft), $\rho = 0.4$ is the most desirable value. Moreover, quite conveniently, this value of ρ does not change across languages explored in this study. Similar to the KD model, values of $\rho < 0.5$ imply that the performance improves when the model relies increasingly on the DNN labels rather than the PT labels. This means interpolation is useful and that the new interpolated targets are effective in alleviating the noise in PT labels. However, similar to the KD model, setting $\rho = 0$ results in very high PERs.

Finally, a summary of the best KD and TI models for each language, along with their parameters, is highlighted in Table 4.9. The average improvement is about 2.12% absolute. This is quite useful for us considering that this is a zero-resource scenario with no access to reliable ground truth DTs in the target URL.

Table 4.8: PERs of different MTL models trained with CE and TI losses. The parameter ρ is the weighting parameter in Eq. (4.11) and Eq. (4.12).

Model	**Parameter**	**Language**			
	ρ	**swh**	**amh**	**din**	**cmn**
MTL-CE	1.0	44.89	60.79	58.65	53.53
MTL-TI (Hard)	0.6	43.96	60.44	58.69	51.14
MTL-TI (Hard)	0.4	44.08	59.98	57.94	**49.81**
MTL-TI (Hard)	0.2	44.24	60.58	59.19	51.20
MTL-TI (Soft)	0.6	43.49	60.19	58.62	51.09
MTL-TI (Soft)	0.4	**43.29**	**59.65**	**57.65**	50.02
MTL-TI (Soft)	0.2	44.16	61.14	59.26	50.79

Table 4.9: Summary of the best MTL-KD and MTL-TI models. Absolute improvements over the MTL-CE model inside parentheses.

Lang	**Baseline (CE)**	**Best**		**Parameters**	
	PER	**PER**	**Model**	ρ	T
swh	44.89	43.29 (1.60)	TI (Soft)	0.4	-
amh	60.79	59.08 (1.71)	KD	0.2	2
din	58.65	57.65 (1.00)	TI (Soft)	0.4	-
cmn	53.53	49.38 (4.15)	KD	0.2	2

4.5.12 Multi-Task Learning With Deep Auto-Encoder (MTL-DAE)

In this experiment, we train the MTL model in a semi-supervised fashion as illustrated in Fig. 4.5. Although the MTL models in the preceding sections improve PERs over the Vanilla DNN, they do not make use of large amounts of untranscribed audio data that are available in the target language. Thus, we use the DAE as an additional sub-task in the MTL framework. The structure of the DAE is simple. It uses the same SHLs as those in the MTL framework. In addition, it has a distinct output layer which is simply an affine transform layer added on top of the final SHL of the MTL. Thus, the SHL acts as the encoder and the affine transform layer acts as the decoder. The DAE is trained to minimize the MSE loss between the input features and output of the decoder.

We used about 4000 untranscribed utterances from the target language for training the DAE. First, fMLLR features were generated for the untranscribed utterances through a two-pass estimation of the fMLLR transforms. The PT adapted MAP GMM-HMM model was treated as the alignment model. Following this, the fMLLR features were normalized to zero mean and unit variance. In an iden-

tical fashion, the input features for all tasks in the MTL were normalized. This helps avoid the possibility of generating large MSE errors at the DAE output. In addition, we keep the weighting term λ_{DAE} in Eq. (4.16) to low values between 0.001-0.005.

The frames used to train the DAE far outnumbered the frames for other tasks. This results in minibatches getting biased toward the DAE task. In order to maintain a balance of frames across all tasks in the minibatch, we create duplicate copies of frames for both the PT and DT tasks. We used 4-6 copies for the PT tasks and 1-2 copies for the DT tasks. The number of copies and acceptable values of λ_{DT} in (4.16) were found from the development set.

Comparisons of the baseline models (GMM-HMM, Vanilla DNN) and the proposed MTL models (MTL+CE, MTL+KD, MTL+TI, MTL+DAE) are given in Figs. 4.6-4.9.

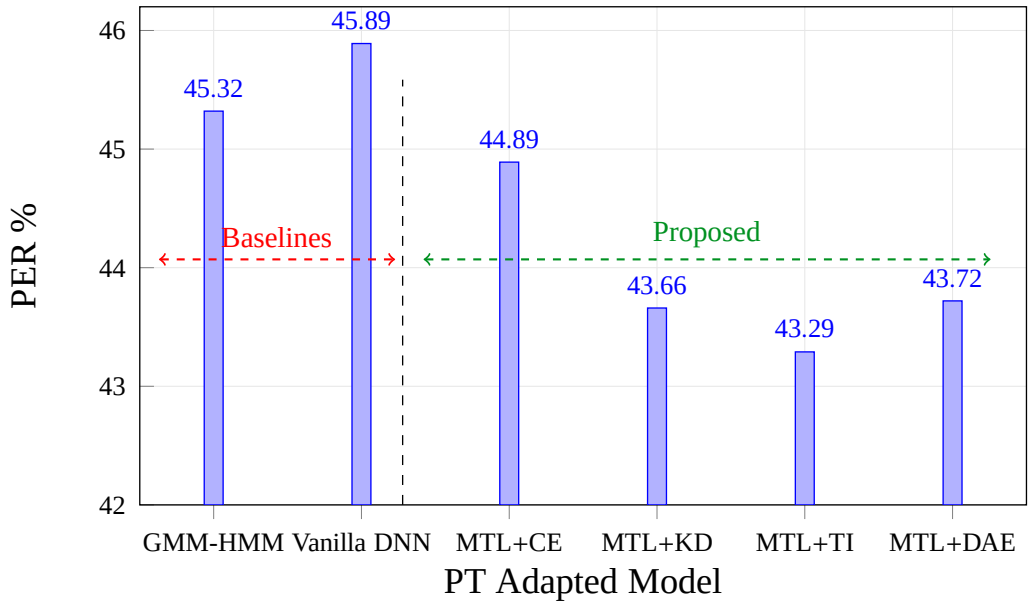

Figure 4.6: Comparison of PERs PT adapted baseline vs. proposed models in Swahili.

4.6 Summary

In this chapter, we summarized the improvements in PERs obtained using the PT adapted proposed MTL models over the unadapted multilingual DNN and

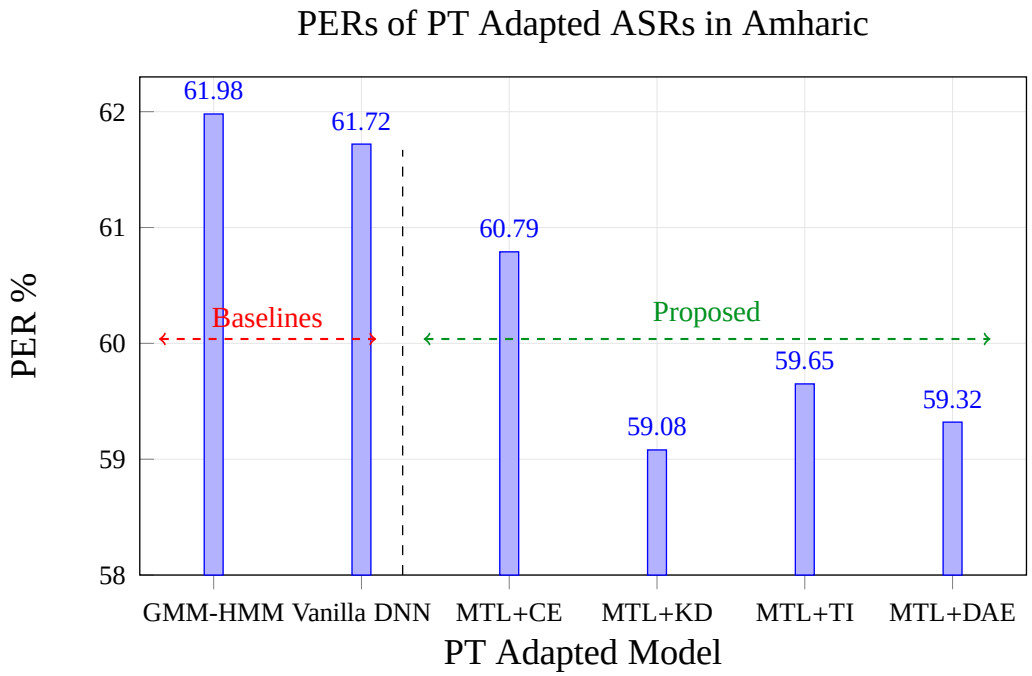

Figure 4.7: Comparison of PERs PT adapted baseline vs. proposed models in Amharic.

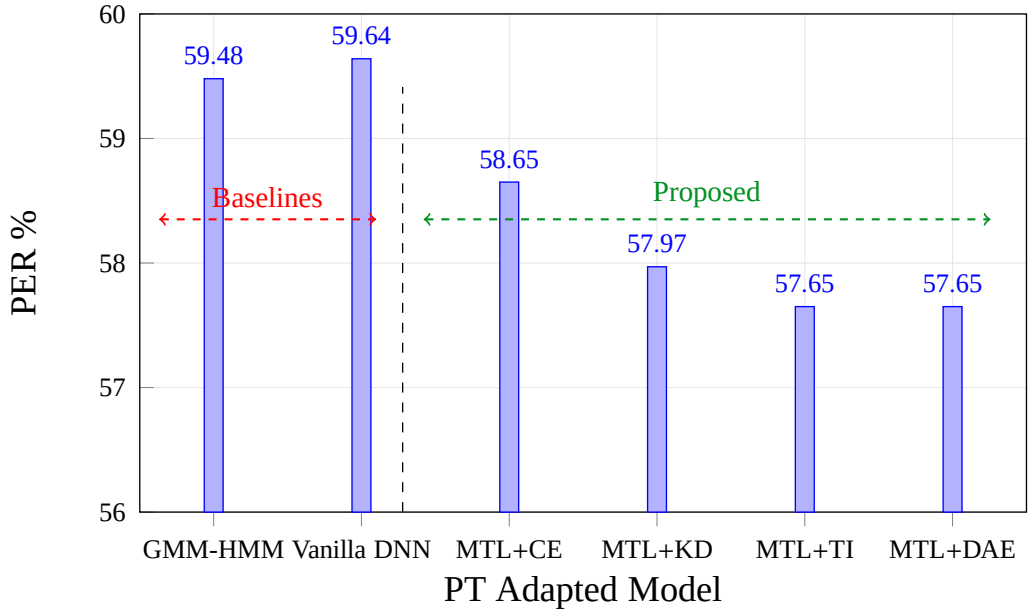

Figure 4.8: Comparison of PERs PT adapted baseline vs. proposed models in Dinka.

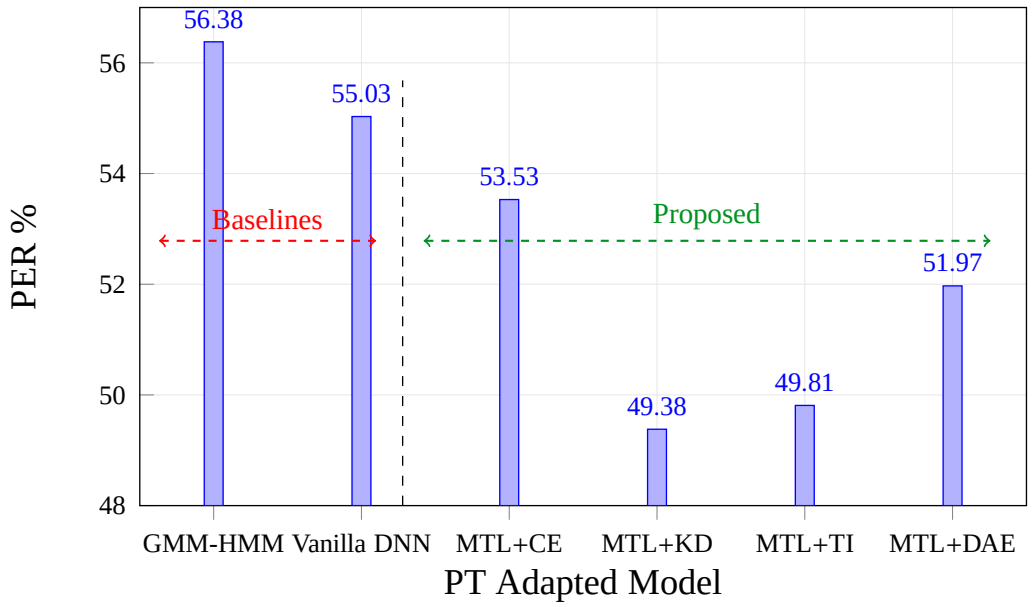

Figure 4.9: Comparison of PERs of PT adapted baseline vs. proposed models in Mandarin.

the PT adapted Vanilla DNN. This is highlighted in the fourth column of Table 4.10. The best MTL models for Swahili, Amharic, Dinka, Mandarin are MTL+TI, MTL+KD, MTL+TI/MTL+DAE, MTL+KD respectively. The average relative improvements of the best MTL over the Multilingual and Vanilla DNNs are 16.22% and 5.89% respectively.

In addition to this, we evaluate the utility factor of PTs in the last column of Table 4.10. We define the utility factor of PTs as the fraction of phones recovered using a PT adapted model (for e.g., best MTL model) compared to a DT adapted model (for e.g., monolingual/oracle DNN). The average utility factor is 43.02%.

Table 4.10: Summary of PERs for the unadapted baseline DNN (MULTI-DNN), PT adapted baseline DNN (Vanilla DNN), PT adapted proposed MTL (best MTL), DT adapted monolingual DNN (MONO-DNN). Relative improvements in PER of the best MTL over MULTI-DNN and Vanilla DNN are in the fourth column. Utility factor of PTs for different languages are in the last column.

Lang	**A. MULTI-DNN** (Unadapted)	**B. Vanilla DNN** (PT Adapted)	**C. Best MTL** (Rel. PER) (PT Adapted)	**D. MONO-DNN** (DT Adapted)	**Utility =** $\frac{A-C}{A-D}$
swh	60.40	45.89	43.29 (28.33, 5.67)	34.25	65.43%
amh	65.56	61.72	59.08 (9.89, 4.28)	46.69	34.34%
din	63.81	59.64	57.65 (9.65, 3.34)	48.37	39.90%
cmn	59.50	55.03	49.38 (17.00,10.27)	28.26	32.40%

Despite these improvements, there are about 35-68% (100 - Utility) phones

that could be recovered. Future work includes compensating label noise by interpolating PT labels with neural network predictions and estimating noisy channel (misperception) models of the non-native Turkers using DNNs.

Chapter 5

End-to-End Large Vocabulary Automatic Speech Recognition

5.1 Introduction

In the last few years, an emerging trend in automatic speech recognition (ASR) research has been the study of E2E systems [4, 81–89]. An E2E ASR system directly transduces an input sequence of acoustic features **x** to an output sequence of probabilities of tokens **y** (phonemes, characters, words etc.). This reconciles well with the notion that ASR is inherently a sequence-to-sequence task mapping input waveforms to output token sequences. Some widely used contemporary E2E approaches for sequence-to-sequence transduction are: (a) Connectionist Temporal Classification (CTC) [1, 11], (b) Recurrent Neural Network Encoder-Decoder (RNN-ED) [12, 13, 26, 27], and (c) RNN Transducer (RNN-T) [90]. These approaches have been successfully applied to large scale ASR [81–85, 88, 91–93]. In this study, we confine our focus primarily on CTC models.

The remainder of the chapter is organized as follows. In Section 5.2, we provide a summary of past work. In Sections 5.3-5.5, we explain the proposed Attention CTC, Hybrid CTC, and Mixed-unit CTC respectively. In Section 5.6, we provide experimental evaluations of our proposed algorithms. Finally, we summarize our study in Section 5.7.

5.2 Background

As one of the most popular E2E methods, the CTC approach [1, 11] was introduced to map input speech frames into output label sequences [4, 73, 81, 82, 92, 94–100]. To deal with the issue that the number of output labels is shorter than the number of input speech frames in speech recognition tasks, CTC introduces a special *blank* label and allows for repetition of labels to force the output and input

sequences to have the same length.

CTC outputs are usually dominated by blank labels. The outputs corresponding to the non-blank labels usually occur with spikes in their posteriors. Thus, an easy way to generate ASR outputs using CTC is to concatenate the non-blank labels corresponding to the posterior spikes and collapse those labels into word outputs if needed. This is known as greedy decoding. It is a very attractive feature for E2E modeling as there is neither any LM nor any complex decoding involved. The E2E models used in this study use greedy decoding.

As the goal of ASR is to generate a word sequence from speech acoustics, the word is the most natural output unit for E2E models. A big challenge in the word-based CTC model, a.k.a. acoustic-to-word CTC or word CTC, is the OOV issue [101–104]. In [92, 95, 99], only the most frequent words in the training set were used as output targets whereas the remaining words were just tagged as OOVs (or UNKs). These OOVs can be neither modeled nor recognized as valid words during evaluation. For example, if the transcription of an utterance is "have you been to newyorkabc" in which newyorkabc is an OOV (infrequent) word, the training token or recognition output sequence for this utterance will be "have you been to OOV". The presence of OOV tag in the ASR output degrades the end-user experience. In [95], a CTC with up to 25 thousand (k) word targets was explored. However, the ASR accuracy of the word CTC was far below the accuracy of a context dependent (CD) phoneme CTC model, partially due to the high OOV rate when using only around 3k hours of training data.

Thus, the accuracy gap between a word CTC and CD phoneme CTC can be attributed to multiple reasons. First, training a word CTC requires orders of magnitude of more training data than a CD phoneme CTC because words which can be modeled well require sufficient number of training examples. Words which do not meet the sufficiency requirement are simply tagged as OOVs. Hence, these words cannot be modeled as valid words during training and recognized during evaluation. Second, even in the presence of large training data, it is difficult to capture the entire vocabulary of a language. For example, a word CTC cannot handle emerging hot-words which become popular after a network has been built.

Several studies in the past have attempted to address these issues. In [92], it was shown that by using 100k words as output targets and by training the model with 125k hours of data, a word CTC was able to outperform a CD phoneme CTC. However, easy accessibility to such large databases is rare. Usually, at most a few thousand hours of data are easily accessible. In [105], the authors were able to

train a word CTC model with only 2k hours of data with ASR accuracy comparable to a CD phoneme CTC. Their proposed training regime included initializing the word CTC with a well-trained phoneme CTC, curriculum learning [106], Nesterov momentum-based stochastic gradient descent, dropout, and low rank matrix factorization [107]. To address the hot-words issue, [105] also proposed a spell and recognize (SAR) model which has a combination of words and characters as output targets. The SAR model is used to learn to first spell a word as a sequence of characters and then recognize it as a whole word. Whenever an OOV is detected, the decoder consults the letter sequence from the speller. Thus, the displayed hypothesis is more meaningful to the users than OOV. However, the overall recognition accuracy of the SAR model improved only marginally over a word-only CTC.

In this study, we propose a three-stage solution to improve the recognition accuracy of the all-neural word CTC using only 3.4k hours of data while also alleviating the OOV issue. Furthermore, our proposed word CTC also outperforms a conventional CD phoneme CTC using strong LM and complex graph based decoding.

- First, we propose *Attention CTC* [20] to address the inherent hard alignment problem in CTC. Since CTC relies on the hidden feature vector at the current time to make predictions, it does not directly attend to feature vectors of the neighboring frames. This is the hard alignment problem. The basic idea for improving CTC is to blend some concepts from RNN-ED into CTC modeling.

- Second, we propose *Hybrid CTC* [100] which is a single CTC consisting of a word CTC and a letter CTC trained jointly using multi-task learning [55, 56]. We train the word CTC first and then add a letter CTC as an auxiliary task by sharing the hidden layers of the word CTC. During recognition, the word CTC generates a word sequence, and the character CTC is only consulted at the OOV segments. This makes the Hybrid CTC capable of recognizing OOVs and thereby reducing errors introduced by OOVs.

- Finally, we further improve the word CTC and reduce OOV errors by introducing *Mixed-unit CTC* [21]. Here, the OOV word is decomposed into a mixed-unit sequence of frequent words and letters at the training stage. During testing, we do greedy decoding for the whole E2E system in a single step without the need of using the two-stage (OOV-detection and then letter-sequence-

consulting) process as in Hybrid CTC.

With all these components, the final word CTC improves the baseline word CTC by relative 12.09% WER reduction and also outperforms the traditional CD phoneme CTC with strong LM and decoder by relative 6.79%.

5.3 Attention CTC

In this section, we outline various steps required to model attention directly within CTC. An example of the proposed Attention CTC network is shown in Fig. 5.1. We propose the following key ideas to blend attention into CTC. (a) First, we derive context vectors using *time convolution features* (Sec 5.3.1) and apply attention weights on these context vectors (Sec 5.3.2). This makes it possible for CTC to be trained using soft alignments instead of hard. (b) Second, to improve attention modeling, we incorporate an *implicit language model* (Sec 5.3.3) during CTC training. (c) Finally, we extend our attention modeling further by introducing *component attention* (Sec 5.3.4) where context vectors are produced as a result of applying attention on hidden features across both time and their individual components. Since our network is basically a CTC network, the input and output sequences are of the same length (i.e., $T = U$). However, we will use the indices t and u to denote the time step for input and output sequences respectively. This is only to maintain notational consistency with the equations in RNN-ED. It is understood that every input frame $\mathbf{x}_t$ generates output $\mathbf{y}_t = \mathbf{y}_u$.

5.3.1 Time Convolution (TC) Features

Consider a rank-3 tensor $\mathbf{W}' \in \mathbb{R}^{n_1 \times n_2 \times C}$. For simplicity, assume $n_1 = n_2 = n$ where n is the dimension of the hidden feature $\mathbf{h}_t$. Our attention model considers a small subsequence of $\mathbf{h}$ rather than the entire sequence. This subsequence, $(\mathbf{h}_{u-\tau}, \cdots, \mathbf{h}_u, \cdots, \mathbf{h}_{u+\tau})$, will be referred to as the *attention window*. Its length is C and it is centered around the current time u. Let τ represent the length of the

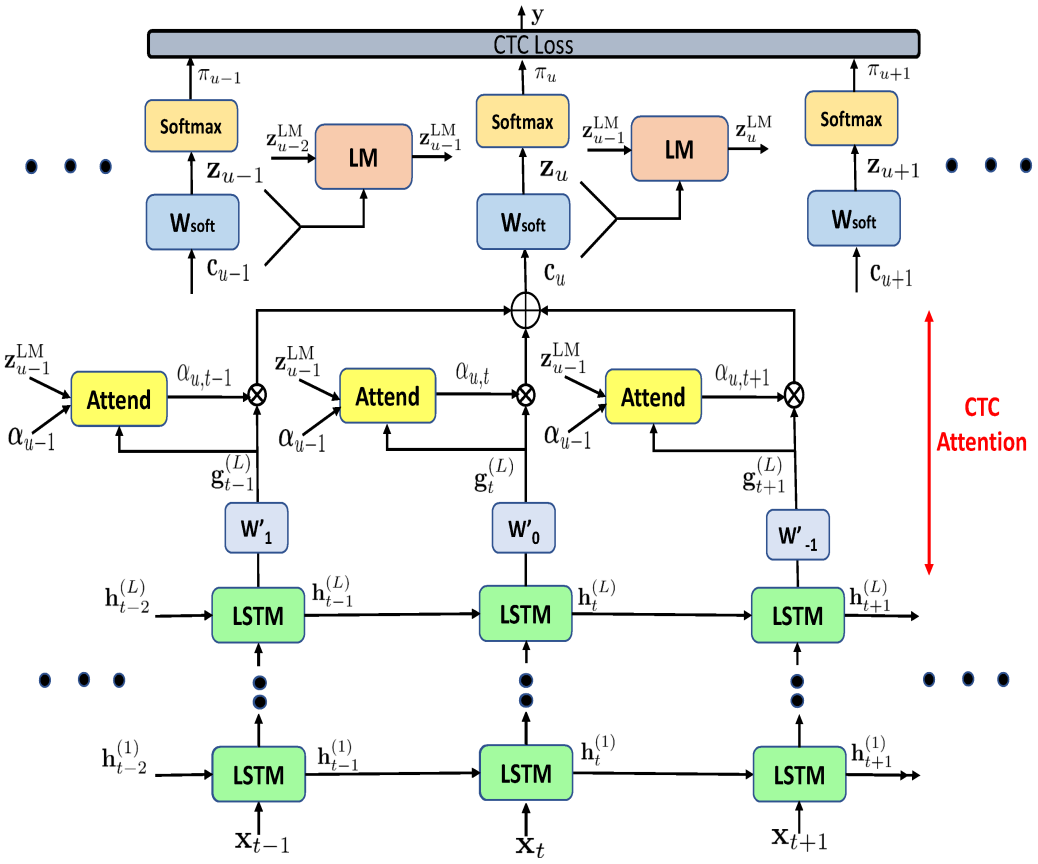

Figure 5.1: An example of an Attention CTC network with an attention window of size $C = 3$ (i.e., $\tau = 1$).

window on either side of u. Thus, $C = 2\tau + 1$. Then $\mathbf{c}_u$ can be computed using

$$\begin{aligned}\mathbf{c}_u &= \mathbf{W}' * \mathbf{h} \\ &= \sum_{t=u-\tau}^{u+\tau} \mathbf{W}'_{u-t}\mathbf{h}_t \\ &\stackrel{\Delta}{=} \sum_{t=u-\tau}^{u+\tau} \mathbf{g}_t \\ &= \gamma \sum_{t=u-\tau}^{u+\tau} \alpha_{u,t}\mathbf{g}_t. \end{aligned} \tag{5.1}$$

Here, $\mathbf{g}_t \in \mathbb{R}^n$ represents the *filtered* signal at time t. The last step Eq. (5.1) holds when $\alpha_{u,t} = \frac{1}{C}$ and $\gamma = C$. Since Eq. (5.1) is similar to Eq. (2.11) in structure, $\mathbf{c}_u$ represents a special case context vector with uniform attention weights $\alpha_{u,t} = \frac{1}{C}$, $t \in [u - \tau,\ u + \tau]$. Also, $\mathbf{c}_u$ is a result of convolving features $\mathbf{h}$ with $\mathbf{W}'$ in time. Thus, $\mathbf{c}_u$ represents a *time convolution feature* and $\mathbf{W}'$ a *time convolution kernel*. This is illustrated in Fig. 5.1 for the case of $\tau = 1$ (after ignoring the Attend block and letting $\alpha_{u,t} = \frac{1}{3}$).

5.3.2 Content Attention (CA) and Hybrid Attention (HA)

To incorporate non-uniform attention in Eq. (5.1), we need to compute a non-uniform $\alpha_{u,t}$ for each $t \in [u - \tau,\ u + \tau]$ using an attention network similar to Eq. (2.12). However, since there is no explicit decoder like Eq. (2.10) in CTC, there is no decoder state $\mathbf{s}_u$. Therefore, we use $\mathbf{z}_u$ instead of $\mathbf{s}_u$. The term $\mathbf{z}_u \in \mathbb{R}^K$ is the logit to the softmax and is given by

$$\begin{aligned}\mathbf{z}_u &= \mathbf{W}_{\text{soft}}\mathbf{c}_u + \mathbf{b}_{\text{soft}}, \\ \mathbf{y}_u &= \text{Softmax}(\mathbf{z}_u), \end{aligned} \tag{5.2}$$

where $\mathbf{W}_{\text{soft}} \in \mathbb{R}^{K\times n}, \mathbf{b}_{\text{soft}} \in \mathbb{R}^K$. Thus, Eq. (5.2) is similar to the Generate(.) function in Eq. (2.9) but lacks the dependency on $\mathbf{y}_{u-1}$ and $\mathbf{s}_u$. Consequently, the Attend(.) function in Eq. (2.12) becomes

$$\boldsymbol{\alpha}_u = \text{Attend}(\mathbf{z}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}), \tag{5.3}$$

where $\mathbf{h}$ in Eq. (2.12) is replaced with $\mathbf{g} = (\mathbf{g}_{u-\tau}, \cdots, \mathbf{g}_{u+\tau})$. Next, the scoring function Score(.) in Eq. (2.13) is modified by replacing the raw signal $\mathbf{h}_t$ with the filtered signal $\mathbf{g}_t$. Thus, the new Score(.) function becomes

$$e_{u,t} = \text{Score}(\mathbf{z}_{u-1}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}_t), \tag{5.4}$$

$$= \begin{cases} \mathbf{v}^T \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{b}), \text{ (content)} \\ \mathbf{v}^T \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{V}\mathbf{f}_{u,t} + \mathbf{b}) \text{ (hybrid)} \end{cases} \tag{5.5}$$

with $\mathbf{f}_{u,t}$ a function of $\boldsymbol{\alpha}_{u-1}$ through Eq. (2.16). The content and location information are encoded in $\mathbf{z}_{u-1}$ and $\boldsymbol{\alpha}_{u-1}$ respectively. The role of $\mathbf{W}$ in Eq. (5.5) is to project $\mathbf{g}_t$ for each $t \in [u-\tau,\ u+\tau]$ to a common subspace. Score normalization of Eq. (5.4) can be achieved using the softmax operation in Eq. (2.14) to generate non-uniform $\alpha_{u,t}$ for $t \in [u-\tau,\ u+\tau]$. Now, $\boldsymbol{\alpha}_u$ can be plugged into Eq. (5.1), along with $\mathbf{g}$ to generate the context vector $\mathbf{c}_u$. This completes the attention network. We found that excluding the scale factor γ in Eq. (5.1), even for non-uniform attention, was detrimental to the final performance. Therefore, we continue to use $\gamma = C$.

5.3.3 Implicit Language Model (LM)

The performance of the attention model can be improved further by providing more reliable content information from the previous time step. This is possible by introducing another recurrent network that can utilize content from several time steps in the past. This network, in essence, would learn an implicit LM. In particular, we feed $\mathbf{z}_{u-1}^{\text{LM}}$ (hidden state of the LM network) instead of $\mathbf{z}_{u-1}$ to the Attend(.) function in Eq. (5.3). To build the LM network, we follow an architecture similar to RNN-LM [108]. As illustrated in the LM block of Fig. 5.1, the input to the network is computed by stacking the previous output $\mathbf{z}_{u-1}$ with the context vector $\mathbf{c}_{u-1}$ and feeding it to a recurrent function $\mathcal{H}(.)$. This is represented as

$$\mathbf{z}_{u-1}^{\text{LM}} = \mathcal{H}(\mathbf{x}_{u-1}, \mathbf{z}_{u-2}^{\text{LM}}), \quad \mathbf{x}_{u-1} = \begin{bmatrix} \mathbf{z}_{u-1} \\ \mathbf{c}_{u-1} \end{bmatrix}, \tag{5.6}$$

$$\boldsymbol{\alpha}_u = \text{Attend}(\mathbf{z}_{u-1}^{\text{LM}}, \boldsymbol{\alpha}_{u-1}, \mathbf{g}). \tag{5.7}$$

We model $\mathcal{H}(.)$ using a long short-term memory (LSTM) unit [109] with n memory cells and input and output dimensions set to $K+n$ (since $\mathbf{x}_{u-1} \in \mathbb{R}^{K+n}$) and n

(since $\mathbf{z}_{u-1}^{\text{LM}} \in \mathbb{R}^n$) respectively. One problem with $\mathbf{z}_{u-1}^{\text{LM}}$ is that it encodes the content of a pseudo LM, rather than a true LM, since CTC outputs are interspersed with blank symbols by design. Another problem is that $\mathbf{z}_{u-1}^{\text{LM}}$ is a real-valued vector instead of a one-hot vector. Hence, this LM is an implicit LM rather than an explicit or a true LM.

5.3.4 Component Attention (COMA)

In the previous sections, $\alpha_{u,t} \in \mathbb{U}$ is a scalar term weighting the contribution of the entire vector $\mathbf{g}_t \in \mathbb{R}^n$ to generate the output $\mathbf{y}_u$ through Eq. (5.1) and Eq. (5.2). This means all n components of the vector $\mathbf{g}_t$ are weighted by the same scalar $\alpha_{u,t}$. In this section, we consider weighting each component of $\mathbf{g}_t$ distinctively. Therefore, we need a vector weight $\boldsymbol{\alpha}_{u,t} \in \mathbb{U}^n$ instead of the scalar weight $\alpha_{u,t} \in \mathbb{U}$ for each $t \in [u-\tau,\ u+\tau]$. The vector $\boldsymbol{\alpha}_{u,t}$ can be generated by first computing an n-dimensional score $\mathbf{e}_{u,t}$ for each t. This is easily achieved using the Score(.) function in Eq. (5.5) but without taking the inner product with $\mathbf{v}$. For example, in the case of hybrid, the scoring function becomes

$$\mathbf{e}_{u,t} = \tanh(\mathbf{U}\mathbf{z}_{u-1} + \mathbf{W}\mathbf{g}_t + \mathbf{V}\mathbf{f}_{u,t} + \mathbf{b}). \tag{5.8}$$

Now, we have C column vectors $[\mathbf{e}_{u,u-\tau}, \cdots, \mathbf{e}_{u,u+\tau}]$, one for each t, where each $\mathbf{e}_{u,t} \in (-1,1)^n$. Let $e_{u,t,j} \in (-1,1)$ be the j^{th} component of the vector $\mathbf{e}_{u,t}$. To compute $\alpha_{u,t,j}$ from $e_{u,t,j}$, we normalize $e_{u,t,j}$ across t keeping j fixed. Thus, $\alpha_{u,t,j}$ is computed as

$$\alpha_{u,t,j} = \frac{\exp(e_{u,t,j})}{\sum_{t'=u-\tau}^{u+\tau} \exp(e_{u,t',j})}, \quad j = 1, \cdots, n. \tag{5.9}$$

Since exp(.) and tanh(.) are both one-to-one functions, their composition is also one-to-one. Thus, there is a one-to-one relation between $\alpha_{u,t,j}$ and $g_t(j)$. Consequently, $\alpha_{u,t,j}$ can be interpreted as the amount of contribution of $g_t(j)$ in computing $c_u(j)$. Now, from Eq. (5.9), we know the values of the vectors $\boldsymbol{\alpha}_{u,t}$, $t \in [u-\tau,\ u+\tau]$. Under the COMA formulation, the context vector $\mathbf{c}_u$ can be computed using

$$\mathbf{c}_u = \text{Annotate}(\boldsymbol{\alpha}_u, \mathbf{g}, \gamma) = \gamma \sum_{t=u-\tau}^{u+\tau} \boldsymbol{\alpha}_{u,t} \odot \mathbf{g}_t, \tag{5.10}$$

where $\odot$ is the Hadamard product.

In the past, attempts have been made to apply attention but on RNN-ED models. For example, in [110], the authors explored the use of attention-based RNN-ED [26,27] for word outputs. In other studies, CTC was used to improve attention-based RNN-ED indirectly using an MTL framework consisting of both CTC and RNN-ED. CTC was used as either at the top layer [111, 112] or at the intermediate encoder layer [113] of the network. However, none of these approaches used attention directly within the CTC network. Note that as extensions of CTC, both RNN-T [90, 93] and RNN aligner [86] either change the objective function or the training process to relax the frame independence assumption of CTC. The proposed Attention CTC is different from all these approaches since we use attention mechanism to improve the hidden layer representations with more context information without changing the CTC objective function and training process. Our primary motivation in this work is to address the hard alignment problem of CTC, as outlined earlier, by modeling attention directly within the CTC framework.

5.4 Hybrid CTC

In this section, we describe the Hybrid CTC model. To solve the OOV issue in the acoustic-to-word model, the Hybrid CTC model uses a word CTC as the primary model and a letter CTC as the auxiliary model in an MTL framework. The word CTC model emits a word sequence, and the output of the letter CTC is only consulted at the segment where the word CTC emits an OOV token. This is illustrated in Fig. 5.2. The word CTC generates the sequence "*play artist OOV*". The word sequence from the letter CTC is "*play artist ratatat*". Since the segment containing "*ratatat*" from the letter CTC has the most time overlap with the segment containing "*OOV*" from the word CTC, the OOV token "*OOV*" is replaced with "*ratatat*". Thus, the final output of the Hybrid CTC is "*play artist ratatat*".

The detailed steps for building the Hybrid CTC model are described as follows:

- Build a multi-layer LSTM-CTC model with words as its output units. Map all the words occurring less than N times in the training data as the OOV token. The output units in this LSTM-CTC model are all the words occurring at least N times in the training data, together with OOV, blank, and silence tokens.

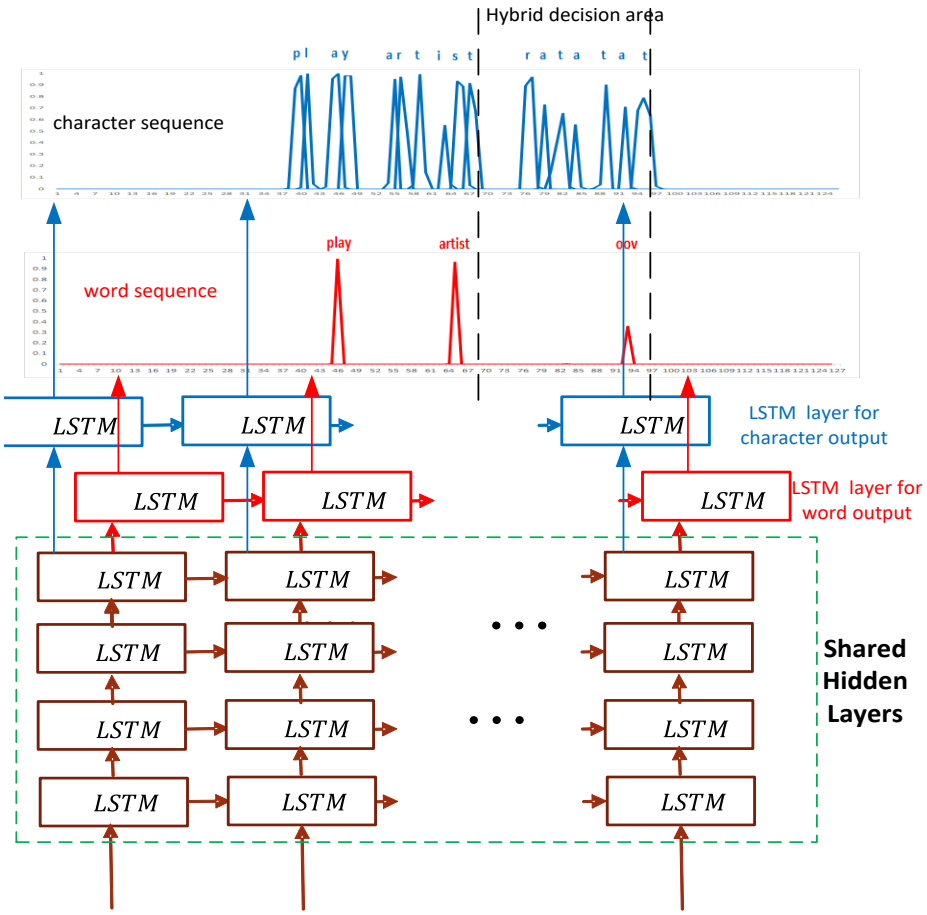

Figure 5.2: An example of how the Hybrid CTC solves the OOV issue of the acoustic-to-word CTC. The words “play, artist, OOV” are obtained from the word CTC. The words “play artist ratatat” are obtained from the letter CTC. Hence, the final output of Hybrid CTC is “play, artist, ratatat” with the first two words obtained from the word CTC and the last word obtained from letter CTC.

- Freeze the bottom $L - 1$ hidden layers of the word-CTC, add one LSTM hidden layer and one softmax layer to build a new LSTM-CTC model with letters as its output units.

- During testing, generate the word output sequence using greedy decoding. If the output word sequence contains an OOV token, replace the OOV token with the word generated from the character CTC that has the largest time overlap with the OOV token.

Table 5.1: Examples of how words are represented with different output units. "Newyork" is a frequent word while "newyorkabc" is an OOV (infrequent word). The word-based CTC treats "newyork" as a unique output node and "newyorkabc" as the OOV output node.

Decomposition Type	newyork	newyorkabc
All words: single-letter	n e w y o r k	n e w y o r k a b c
All words: double-letter	ne wy or k	ne wy or ka bc
All words: triple-letter	new yor k	new yor kab c
All words: word	newyork	OOV
OOVs only: single-letter	newyork	n e w y o r k a b c
OOVs only: word+single-letter	newyork	newyork a b c
OOVs only: word+triple-letter	newyork	newyork abc

5.5 Multi-letter and Mixed-unit CTC

In this section, we first describe the letter-based Multi-letter CTC and then the word-based Mixed-unit CTC.

5.5.1 Multi-letter CTC

Inspired by gram CTC [98] and multi-phone CTC [114], we extend the output units with double-letter and triple-letter units to benefit from long temporal units which are more stable. We hope to improve the Hybrid CTC system as the OOV token may be replaced by more precise words generated by the CTC with multi-letter units.

Gram CTC and multi-phone CTC are based on letters and phonemes respectively, but allow to output variable number of letters (i.e., gram) and phonemes at each time step. The units in gram CTC and multi-phone CTC are learned automatically with the modified forward-backward algorithm to take care of all the decompositions. Both of them need much more complicated decoding than greedy decoding when generating outputs. In contrast, we simply decompose every word (which includes both frequent and OOV words) into a sequence of one or more letter units, with examples shown in the first three rows of Table 5.1. This decomposition is much simpler, without changing the CTC forward-backward process and can use the same greedy decoding procedure as the CTC with single-letter units.

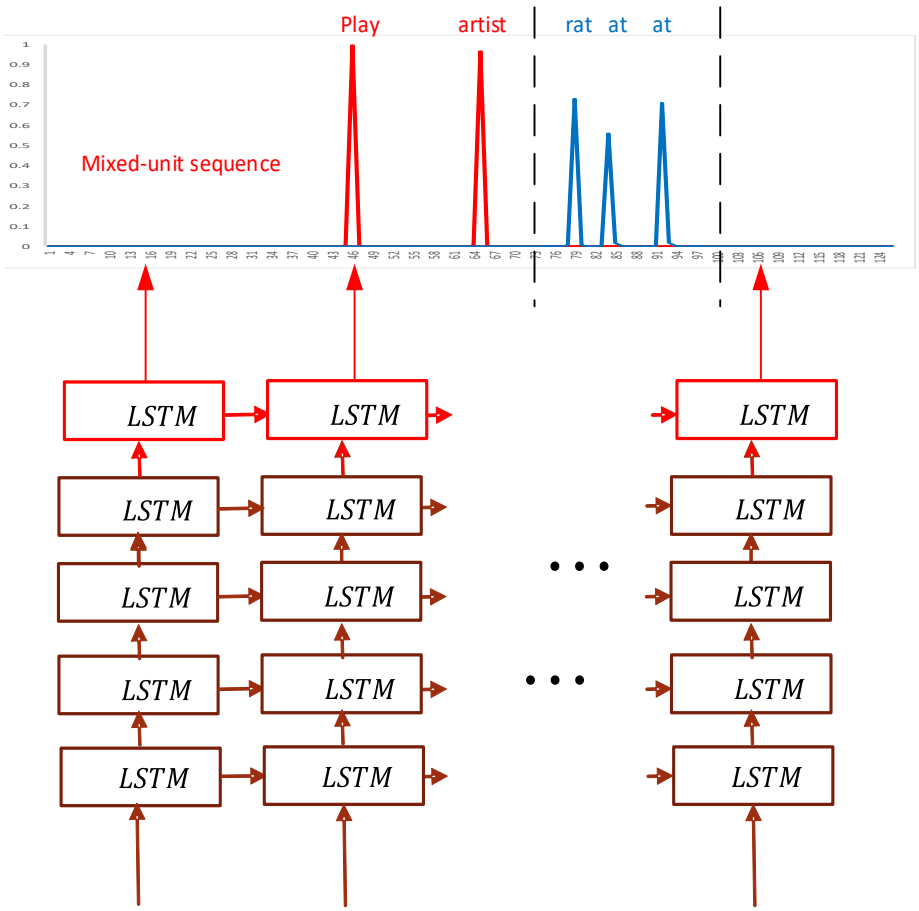

Figure 5.3: An example of how the Mixed-unit CTC solves the OOV issue of the acoustic-to-word CTC. The final output of Mixed-unit CTC is “play, artist, rat at at”.

5.5.2 Mixed-unit CTC

In Hybrid CTC, the shared-hidden-layer constraint is used to help the time synchronization of word outputs between the word and letter CTC models. However, the blank symbol dominates most of the frames, and therefore the time synchronization may not be very reliable. The ideal case should be when the spoken word is in the frequent word list, the system emits a word output. And when the spoken word is an OOV (infrequent) word, the system emits a letter sequence from which a word is generated by collapsing the letter sequence. This cannot be done with the Hybrid CTC because the two CTCs are running in parallel without a perfect time synchronization. A direct solution is to train a single CTC model with mixed units (combination of words and multi-letter units). This is illustrated in Fig. 5.3. If the word is a frequent word, then we just keep it in the output token list. If the

word is an OOV (infrequent word), then we decompose it into a letter sequence. As shown in the fifth row of Table 5.1, the OOV "newyorkabc" is decomposed into "n e w y o r k a b c" for single letter decompositions. However, the word "newyork" is not decomposed because it is a frequent word. Therefore, the output units of the CTC are mixed units, with both words (for frequent words) and letters (for OOV words).

However, we note that artificially decomposing OOVs only into single-letter sequences may confuse CTC training because the network output modeling units are frequent words and letters. To solve such a potential issue, we decompose the OOV words into a combination of frequent words and letters. For example, in the last two rows of Table 5.1, the OOV "newyorkabc" is decomposed into "newyork a b c" if we use words and single-letter units or "newyork abc" if we use words and triple-letter units. In the CTC with mixed units, we use "$" to separate each word in the sentence. For example, the sentence "have you been to newyorkabc" is decomposed into "$ have $ you $ been $ to $ newyork abc $". If $ is not used to separate words, we do not know how to collapse the mixed units (words+letters) into output word sequences. During training, since the OOV words are decomposed into mixed units from words and letters, there is no OOV output node in the Mixed-unit CTC model. Consequently, during testing, the model is very likely to emit OOV words as a sequence of frequent words and letters while still emitting frequent words when frequent words are spoken.

In the past, other studies [93, 115] have explored using sub-word units such as wordpieces [116]. Using wordpieces, a word is decomposed into smaller lexical units which can be a mixture of valid words and non-linguistic multi-letter units based on their frequency of occurrences. Our approach is different from building wordpieces since we decompose *only* OOVs while still retaining the high frequency words as whole word units.

5.6 Results

In this section, we compare the performance of the proposed CTCs with the baseline CTCs. The proposed methods were evaluated using the Microsoft's Cortana voice assistant task. The training and test sets contain approximately 3.3 million utterances (~ 3400 hours) and 5600 utterances (~ 6 hours) respectively in US-English. First, we evaluate the performance of our proposed Attention CTC

(Section 5.3) and Multi-letter CTC (Section 5.5.1) using letter CTCs. Then, we evaluate the performance of our proposed Hybrid CTC (Section 5.4) and Mixed Unit CTC (Section 5.5.2) using both words and letters as output targets.

All CTC models were trained on top of either unidirectional or bidirectional LSTMs (BLSTMs). The unidirectional LSTM has 1024 memory cells while the BLSTM has 512 memory cells in each direction (therefore still 1024 output dimensions when combining outputs from both directions). Then they are linearly projected to 512 dimensions. The base feature vector computed every 10 ms frame is an 80-dimensional vector containing log filterbank energies. Eight frames of base features were stacked together ($m = 80 \times 8 = 640$) as the input to the unidirectional CTC, while three frames were stacked together ($m = 80 \times 3 = 240$) as the input to the bidirectional CTC. The skip size for both unidirectional and bidirectional CTCs was three frames as in [95]. The dimension n of vectors $\mathbf{h}_t, \mathbf{g}_t, \mathbf{c}_u$ was set to 512. For decoding, the greedy decoding procedure (no complex decoder or external LM) was used. Thus, our systems are pure all-neural systems.

5.6.1 Experiments with Letter-Based CTCs

In this section, we evaluate the performance of our proposed Attention CTC (Section 5.3) and Multi-letter CTC (Section 5.5.1) using letters as output targets. The motivation behind improving letter-based CTC is the following. As the outputs from the letter CTC are used to replace the OOV token from the word CTC during testing, the letter CTC should be as accurate as possible.

5.6.1.1 Unidirectional CTC with 28-letter set (Section 5.3)

In the first set of experiments, Vanilla CTC [1] and Attention CTC models were evaluated with a unidirectional 5-layer LSTM. The output layer has 28 output nodes (hence, K = 28) corresponding to a 28-letter set (26 letters 'a'-'z' + space + blank). τ was empirically set to 4, which means the context window size (C) for attention was 9. The results are tabulated in Table 5.2. The top row summarizes the WER for Vanilla CTC. All subsequent rows under "Attention CTC" summarize the WER for the proposed CTC models when attention modeling capabilities were gradually added in a stage-wise fashion. The best Attention CTC model is in the last row and it outperforms the Vanilla CTC model by 18.75% relative.

Table 5.2: WERs of letter-based Vanilla CTC [1] and Attention CTC for $\tau = 4$ ($C = 9$) trained with a 5-layer unidirectional LSTM and 28-letter set. Relative WER improvements are in parentheses.

E2E Model (letter-based)	WER (%)
Vanilla CTC	29.60 (0.00)
Attention CTC	
TC (Sec 5.3.1)	27.36 (07.56)
+CA (Sec 5.3.2)	25.41 (14.16)
+HA (Sec 5.3.2)	25.62 (13.45)
+LM (Sec 5.3.3)	24.74 (16.42)
+COMA (Sec 5.3.4)	**24.05 (18.75)**

Table 5.3: WERs of letter-based Vanilla CTC [1] and Attention CTC for $\tau = 4$ ($C = 9$) trained with a 5-layer BLSTM and 28-letter set. Relative WER improvements are in parentheses.

E2E Model (letter-based)	WER (%)
Vanilla CTC	26.36 (0.00)
Attention CTC	
TC (Sec 5.3.1)	25.21 (04.36)
+CA (Sec 5.3.2)	22.73 (13.77)
+HA (Sec 5.3.2)	22.52 (14.57)
+LM (Sec 5.3.3)	21.69 (17.72)
+COMA (Sec 5.3.4)	**20.81 (21.06)**

There is a slight increase in WER when adding HA on top of CA. In general, for the other experiments, we find that adding HA is beneficial although the gains are marginal compared to all the other enhancements (CA, LM, COMA). Benefits of location based attention could become more pronounced when attention spans over very large contexts [26].

5.6.1.2 Bidirectional CTC with 28-letter set (Section 5.3)

In this set of experiments, the Vanilla and Attention CTC models were evaluated with a BLSTM of 5-layers and $\tau = 4$ using the 28-letter set. Otherwise, we followed the same training regime as in the previous Section 5.6.1.1. The results are tabulated in Table 5.3. Similar to the unidirectional case, the best Attention CTC model outperforms Vanilla CTC by about 21.06% relative. This shows that even a strong baseline like bidirectional CTC does not undermine the efficacy of the proposed Attention CTC models.

Table 5.4: WERs of letter-based Vanilla CTC [1] and Attention CTC for $\tau = 4$ ($C = 9$) trained with a 5-layer BLSTM and 83-letter set. Relative WER improvements are in parentheses.

E2E Model (letter-based)	WER (%)
Vanilla CTC	23.29 (0.00)
Attention CTC	
TC (Sec 5.3.1)	22.30 (04.25)
+CA (Sec 5.3.2)	21.34 (08.37)
+HA (Sec 5.3.2)	20.81 (10.65)
+LM (Sec 5.3.3)	19.98 (14.21)
+COMA (Sec 5.3.4)	**18.49 (20.61)**

5.6.1.3 Bidirectional CTC with 83-letter set (Section 5.3)

In this set of experiments, in addition to the BLSTM, we construct a new letter set by adding new characters on top of the 28-letter set [97]. These additional letters include capital letters used in the word-initial position, double-letter units representing repeated characters like *ll*, apostrophes followed by letters such as *'de*, *'r* etc. Readers may refer to [97] for more details. Altogether such a large unit inventory has 83 letters, and we refer to it as the 83-letter set. The results for this experiment are tabulated in Table 5.4. Again, Attention CTC models consistently outperform Vanilla CTC with the best relative improvement close to 20.61%. This shows that the proposed Attention CTC network can achieve similar improvements, no matter whether the Vanilla CTC is built with advanced modeling capabilities (from unidirectional to bidirectional) or different sets of letter units (28 vs. 83 units).

5.6.1.4 Multi-letter CTC (Section 5.5.1)

In the preceding experiment, we were able to improve the WER by expanding the number of letters. Motivated by these observations, we evaluate the impact of using different sizes of letter units. The single-letter set has 30 symbols, including 26 English characters [a-z], ', *, $, and blank. The double-letter and triple-letter sets have 763 and 8939 symbols respectively, covering all the double-letter and triple-letter occurrence in the training set. All the CTC models in this section are 6-layer BLSTMs. As shown in the second column of Table 5.5, the WER reduces significantly when the output units become larger, i.e., more stable. The letter CTC using triple-letter as output units achieves 13.28% WER, reducing 24.29%

Table 5.5: WERs of letter-based CTC models, trained with 6-layer BLSTMs, having single, double, and triple-letter output units (Section 5.5.1). Three structures are evaluated: Vanilla CTC [1], Attention CTC ($\tau = 4$), and Attention CTC ($\tau = 4$) sharing 5 hidden layers with the word CTC.

E2E Model (letter-based)	WER (%)		
	Vanilla	Attention	Attention 5 layers sharing
single-letter	17.54	14.30	16.74
double-letter	15.37	12.16	14.00
triple-letter	13.28	11.36	12.81

relative WER from the letter CTC using single-letter as output units.

The Attention CTC is then trained with $\tau = 4$. As shown in the third column of Table 5.5, Attention CTC improves over the Vanilla CTC hugely, obtaining 18.47%, 20.88%, and 14.46% relative WER reduction for single-letter, double-letter, and triple-letter CTC models respectively. The best letter CTC is the one with triple-letter outputs and attention modeling, which can obtain 11.36% WER.

The Hybrid CTC model described in Section 5.4 has both word and letter CTCs, which share 5 hidden LSTM layers. On top of the shared hidden layers, we add a new LSTM hidden layer and a softmax layer to model letter outputs (single, double, or triple-letters). Attention modeling is applied to boost the performance. As shown in the fourth column of Table 5.5, the WER of letter CTC with such shared-hidden-layer constraint performs worse than its counterpart (Attention CTC in third column). This indicates one shortcoming of the Hybrid CTC - it sacrifices the accuracy of the letter CTC because of the shared-hidden-layer constraint used to synchronize the word outputs between the word and letter CTCs.

5.6.2 Experiments With Word-Based CTCs

Having improved the letter CTC in the previous section, we now evaluate the performance of our proposed Hybrid CTC (Section 5.4) and Mixed-unit CTC (Section 5.5.2) using both words and letters as output targets. However, we refer to these CTCs as word CTCs for simplicity in terminology. We are primarily interested in boosting the accuracy of recognizing non-OOV words while also recognizing the OOV words as close as possible to the ground truth words.

For the baseline word CTC (Vanilla CTC [1]) model, we built a 6-layer BLSTM. This model has around 27k output targets which is the same as the number of fre-

Table 5.6: WERs of word-based Vanilla CTC [1] and Hybrid CTC (Section 5.4) models. All Hybrid CTC models have a word-based CTC and a letter-based Attention CTC ($\tau = 4$), sharing 5 hidden layers. All CTC models were trained with 6-layer BLSTMs.

E2E Model	WER (%)
Vanilla CTC (word only)	9.84
Hybrid CTC: word + double-letter Attention CTC	9.66
Hybrid CTC: word + triple-letter Attention CTC	9.66

quent words in the training data. These frequent words occurred at least 10 times in the training data. All the other words (infrequent words) were mapped to an OOV output token. We have also tried other word CTCs with varying number of output units. However, the model using 27k word outputs performs the best. This word CTC model yields 9.84% WER, among which the OOV tokens contribute 1.87% WER. It significantly improves the WER of unidirectional word CTC reported in [100] which indicates that bidirectional modeling is critical to the E2E system. Unless otherwise stated, all CTC models in this section except Attention CTC use the same structure as the Vanilla CTC model.

5.6.2.1 Hybrid CTC (Section 5.4)

As the CTC models with double-letter and triple-letter output units worked very well in Table 5.5, we use them to build the Hybrid CTC models with the OOV lookup process described in Section 5.4. The Hybrid CTC uses a 6-layer BLSTM, i.e., 5 shared-hidden-layers and an additional layer for each task (word and letter CTC). Thus, the underlying structure of Hybrid CTC is similar to that of the Vanilla CTC. As shown in Table 5.6, both hybrid models achieved 9.66% WER. Several factors contribute to such small improvement (from 9.84% WER of the Vanilla CTC) of the Hybrid CTC. First, the shared-hidden-layer constraint degrades the performance of the letter CTC, potentially affecting the final hybrid system performance. Second, although the shared-hidden-layer constraint helps to synchronize the word outputs from the word and letter CTC, we still observed that the time synchronization can fail sometimes. In such cases, the OOV token is replaced with its neighboring frequently occurring word because of word segments misalignment. Because of these factors, although the triple-letter CTC is better than double-letter CTC in Table 5.5, there is no difference in the WERs when they are integrated into the Hybrid CTC setup in which they handle only a

Table 5.7: WERs of word-based Vanilla CTC [1], Mixed-unit CTC (Section 5.5.2), and Mixed-unit CTC + Attention. All CTC models were trained with 6-layer BLSTMs.

E2E Model	WER (%)
Vanilla CTC (word only)	9.84
Mixed (OOV: single-letter) CTC	20.10
Mixed (OOV: word + single-letter) CTC	10.17
Mixed (OOV: word + double-letter) CTC	9.58
Mixed (OOV: word + triple-letter) CTC	9.32
Mixed (OOV: word + triple-letter) Attention CTC	8.65

Table 5.8: Summary of WERs of conventional CD phoneme CTC, word-based Vanilla CTC [1], and word-based Mixed-unit CTC + Attention. All CTC models were trained with 6-layer BLSTMs.

Model	LM	WER(%)
1. Conventional: CD phoneme CTC	✓	9.28
2. E2E: Vanilla CTC (word only)	✗	9.84
3. E2E: Mixed-unit + Attention CTC	✗	8.65
Improvement	#3 vs #1	#3 vs #2
Relative	6.79%	12.09%

small portion of OOV words.

5.6.2.2 Mixed-unit CTC (Section 5.5.2)

We evaluate the CTC with mixed units in Table 5.7. As before, the word-based Vanilla CTC achieves a WER of 9.84%. In the first experiment, the mixed units contain single-letters and 27k frequent words. During training, we decompose OOV words into single-letter sequences. As analyzed in Section 5.5.2, artificially decomposing OOV words into letter sequences, while still retaining the frequent words, confuses CTC training. Therefore, the trained CTC model achieved 20.10% WER. When looking at the posterior spikes of this model, we observed that the word spikes and letter spikes are scattered into each other which proves our hypothesis.

Next, we decompose OOV words into a combination of frequent word and single-letter sequences, and train the CTC network with the mixed units (around 27k output targets). Immediately, the WER improved to 10.17%, but still a little worse than the Vanilla CTC. This is because the single-letter sequence brings instability to the modeling. When we decompose the OOV words into a combina-

tion of frequent words and double-letters (slightly higher than 27k output targets), the situation becomes better, and the resulting WER is 9.58%. When the triple-letters and frequent words are used (totally 33k output targets), the WER reaches 9.32%, beating the Vanilla CTC by 5.28% relative WER reduction.

Next, we improve the final Mixed-unit CTC model by applying attention. To save computational costs, because of large number of output units, we did not integrate the implicit LM in Eq. (5.6). The WER becomes 8.65%, which is about relative 12.09% WER reduction from the 9.84% WER of Vanilla CTC.

Finally, we compare the Mixed-unit + Attention CTC model with a traditional CD phoneme CTC. We trained a CD phoneme 6-layer BLSTM with the CTC criterion, modeling around 9000 tied CD phonemes. This CD phoneme CTC model achieves 9.28% WER when decoding with a 5-gram LM with totally around 100 million (M) N-grams. Despite a strong CD phoneme CTC model (with LM), the Mixed-unit + Attention CTC model (without any LM or complex decoder) is able to outperform the CD phoneme CTC model by about 6.79% relative. We summarize these WERs in Table 5.8.

Note that the proposed method not only reduces the WER of the word CTC, but also improves the user experience. The proposed method provides more meaningful output without outputting any OOV token to distract users. Most of the time, even if the proposed method cannot get the OOV word right, it comes out with a very close output. For example, the proposed method recognizes “text fabine” as “text fabian” and “call zubiate” as “call zubiat”, while the Vanilla CTC can only output “text OOV” and “call OOV”.

5.7 Summary

We advance acoustic-to-word CTC model by proposing Attention CTC, Hybrid CTC, and Mixed-unit CTC. In Attention CTC, we blend attention-based modeling capability directly within the CTC framework. To solve the OOV issue in word CTC, we presented Hybrid CTC which uses a word and letter CTC as the primary and auxiliary tasks in an MTL framework. Finally, to boost the performance of Hybrid CTC, we introduced Mixed-unit CTC whose output units are frequent words combined with sequences of multi-letters. For the frequent word, we just model it with a unique output node. For the OOV word, we decompose it into a sequence of frequent words and multi-letters. We evaluate all these methods

on a 3400 hours Microsoft Cortana voice assistant task. The proposed acoustic-to-word Mixed-unit CTC when combined with attention reduces relative 12.09% WER from the word-based Vanilla CTC. Such an acoustic-to-word CTC is a pure end-to-end model without any LM and complex decoder. It also outperforms the traditional CD phoneme CTC with strong LM and decoder by relative 6.79% WER reduction.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we proposed ASR models for two contrasting tasks characterized by the amounts of labeled data available for training. We discussed various drawbacks associated with each task, proposed solutions, and successfully demonstrated the efficacy of our proposition using experiments conducted with real-world data.

In the first part of the thesis, we dealt with under-resourced scenarios which are typically prevalent in under-resourced languages. In the first case, we assumed the availability of very limited amounts of transcribed data in the target language (URL) while simultaneously having access to large amounts of transcribed data in several source languages (WRLs). Training ASR systems with the limited data in the target language often leads to poor generalization primarily due to the overfitting problem. To alleviate this, we proposed transfer learning techniques which transfer the acoustical knowledge from the source languages to the target language. In particular, we defined a new objective function which minimizes the cross-entropy of the target language along with a regularizer which minimizes the cross-entropy of the source languages. It is well known that regularization helps prevent overfitting the model by constraining the model parameters to lie in a more reliable space.

In the second case, we assumed a more adverse scenario when there are absolutely no transcribed data in the target language. This can be attributed to the difficulty of finding native transcribers in the target language. To partially overcome this difficulty, we resorted to collecting transcriptions from online non-native crowd workers, or Turkers, who neither speak the target language nor have any familiarity with it. Because of their non-nativity, the labels they provide are usually inaccurate and noisy. We experimentally proved that DNNs trained using

noisy labels do not necessarily improve error rates over GMMs. To mitigate this problem, we proposed four DNN models trained with MTL style training. The first model was trained using a mixture of noisy PTs and clean DTs in two separate sub-tasks of the MTL network. In the second model, we added a DAE as a third sub-task. The DAE tries to reconstruct the inputs (raw features) at its output by minimizing the MSE between the inputs and the outputs. This is a case of semi-supervised learning since the DAE can be trained without labels. In the third model, we proposed training MTL using Knowledge Distillation. Here, we were able to transfer knowledge from a well-trained multilingual DNN (teacher model) to the target language DNN (student model) using a generalized softmax. In the fourth model, we proposed training MTL using Target Interpolation. In this method, the confidences of the labels provided by noisy transcriptions are modified using the confidences of the target language DNN.

In the second part of the thesis, we proposed advancing all-neural speech recognition by directly incorporating attention modeling within the CTC framework. One drawback of with CTC is the hard alignment problem as it relies only on the current input to generate the current output. In reality, the output at the current time is influenced not only by the current input but also by inputs in the past and future. We address this issue by incorporating attention into the CTC framework. The key idea behind attention is that it is able to apply weights to each of current, past, and future inputs depending on the degree of influence they exert on the output. To this end, we derive new context vectors using time convolution features to model attention as part of the CTC network. To further improve attention modeling, we utilize content information extracted from a network representing an implicit language model. Finally, we introduce vector based attention weights that are applied on context vectors across both time and their individual components.

6.2 Future Work

This thesis lays the foundation for some interesting future research directions.

- Despite the initial success in training ASR systems using noisy labels provided by the Turkers (non-native speakers), a number of problems still need to be addressed. Finding the kinds of errors the Turkers make while transcribing a

foreign language could further lower the error rates. For example, an American Turker will usually perceive the three allophones of the voiceless stop consonant - $[\mathrm{t}]$ (voiceless alveolar stop), $[\underset{\sqcap}{\mathrm{t}}]$ (voiceless dental stop), and $[\underset{\sqcap}{\mathrm{t}}^{\mathrm{h}}]$ (voiceless aspirated dental stop) - as only $[\mathrm{t}]$. Thus, the Turker makes two errors by misperceiving the allophones $[\underset{\sqcap}{\mathrm{t}}]$ and $[\underset{\sqcap}{\mathrm{t}}^{\mathrm{h}}]$ as $[\mathrm{t}]$. The question to address is: How do we incorporate these misperception errors into a DNN? One possible solution is the following. We know we have DTs for the WRLs. On top of this, we could collect corresponding PTs by letting the non-native Turkers transcribe the same set of utterances. Following this, we could train a DNN using DTs first. A softmax layer could then be added on top of this trained DNN. Then the DNN could be retrained using PTs, updating the parameters of the softmax layer while keeping the lower layers fixed. The net effect of this two-stage training is that the resulting DNN models the misperception errors caused by the Turkers and has the ability to auto-correct such errors.

- One drawback of the attention-based CTC model is that it does not make use of large amounts of text-only data that are easily available online from news broadcasts, articles, books etc. It is possible to use a recurrent neural network transducer (RNN-T) [90] training paradigm that can train on both text and acoustic data. This will have the ability to learn a language model and an acoustic model in the same network.

- Attention modeling could be further explored for acoustic model adaptation in different environments. Acoustic models tend to be domain dependent and do not perform well if there is a mismatch between training and test conditions. As an alternative, the Mixture of Experts (MoE) model [117–119] was introduced for multi-domain modeling. It combines the outputs of several domain specific models (or experts) using a gating network. The role of the gating network is to derive weights, one for each expert. The final output of the MoE model is a linear combination of the outputs of the experts weighted by the weights obtained from the gating network. However, one drawback is that the gating network directly uses raw inputs and is unaware of the state of the experts. Moreover, the gating network does not take into consideration the output at the previous time step. The MoE model could benefit by using our attention model instead of a simple gating network. First, the outputs of the experts could be used as the inputs to the attention model. Next, the attention model could use the outputs and expert weights from the previous time step to generate

the weights at the current time. From our initial experiments in [120], we have demonstrated that a MoE model equipped with attention is able to outperform a baseline model using an LSTM based gating network and lowers the WER by 20.48% relative.

References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. Int. Conf. in Learning Representations*, 2006, pp. 369–376.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Sig. Process. Magazine.*, vol. 29, no. 6, pp. 82–97, 2012.

[3] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams et al., "Recent advances in deep learning for speech research at Microsoft," in *Proc. ICASSP*. IEEE, 2013, pp. 8604–8608.

[4] D. Yu and J. Li, "Recent progresses in deep learning based acoustic models," *IEEE/CAA J. of Autom. Sinica.*, vol. 4, no. 3, pp. 399–412, July 2017. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7974889

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 30–42, Jan 2012.

[6] O. Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[8] O. Vinyals, S. V. Ravuri, and D. Povey, "Revisiting recurrent neural networks for robust ASR," in *ICASSP*, 2012.

[9] L. Besacier, E. Barnard, A. Karpov, and T. Schultz, "Automatic speech recognition for under-resourced languages: A survey," *Speech Communication*, vol. 56, pp. 85–100, Jan 2014.

[10] D. Crystal, *Language Death*, First ed. New York: Cambridge University Press, 2002, ISBN 978-0521012713.

[11] A. Graves and N. Jaitley, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. of Machine Learning Research*, 2014, pp. 1764–1772.

[12] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Empirical Methods in Natural Language Processing*, 2014.

[13] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[14] A. Das and M. Hasegawa-Johnson, "Cross-lingual transfer learning during supervised training in low-resource scenarios," in *Interspeech*, 2015, pp. 3531–3535.

[15] M. Hasegawa-Johnson, P. Jyothi, D. McCloy, M. Mirbagheri, G. Liberto, A. Das, B. Ekin, C. Liu, V. Manohar, H. Tang, E. Lalor, N. Chen, P. Hager, T. Kekona, R. Sloan, and A. K. C. Lee, "ASR for under-resourced languages from probabilistic transcription," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 1, pp. 46 – 59, 2017.

[16] A. Das and M. Hasegawa-Johnson, "An investigation on training deep neural networks using probabilistic transcriptions," in *Interspeech*, 2016, pp. 3858–3862.

[17] A. Das, P. Jyothi, and M. Hasegawa-Johnson, "Automatic speech recognition using probabilistic transcriptions in Swahili, Amharic, and Dinka," in *Interspeech*, 2016.

[18] A. Das, M. Hasegawa-Johnson, and K. Veselý, "Deep autoencoder based multi-task learning using probabilistic transcriptions," in *Interspeech*, 2017, pp. 2073–2077.

[19] A. Das and M. Hasegawa-Johnson, "Improving DNNs trained with non-native transcriptions using knowledge distillation and target interpolation," in *Interspeech*, 2018, pp. 2434–2438.

[20] A. Das, J. Li, R. Zhao, and Y. Gong, "Advancing connectionist temporal classification with attention modeling," in *Proc. ICASSP*, 2018.

[21] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, "Advancing acoustic-to-word CTC model," in *Proc. ICASSP*, 2018.

[22] A. Das, J. Li, G. Ye, R. Zhao, and Y. Gong, "Advancing acoustic-to-word CTC model with attention and mixed-units," *IEEE Trans. Audio, Speech, Lang. Process.*, submitted for publication.

[23] P. Jyothi and M. Hasegawa-Johnson, "Acquiring speech transcriptions using mismatched crowdsourcing," in *AAAI*, 2015.

[24] P. Jyothi and M. Hasegawa-Johnson, "Transcribing continuous speech using mismatched crowdsourcing," in *Interspeech*, 2015, pp. 2774–2778.

[25] H. Sak, A. Senior, K. Rao, O. Irosy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Proc. ICASSP*, 2015, pp. 4280–4284.

[26] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brake, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *CoRR*, vol. abs/1508.04395, 2015. [Online]. Available: http://arxiv.org/abs/1508.04395

[27] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NIPS*, 2015.

[28] B. Wheatley, K. Kondo, W. Anderson, and Y. Muthusamy, "An evaluation of cross-language adaptation for rapid HMM development in a new language," in *ICASSP*, vol. 1, 1994, pp. I237–240.

[29] T. Schultz and A. Waibel, "Fast bootstrapping of LVCSR systems with multilingual phoneme sets," in *Eurospeech.*, 1997.

[30] J. Kohler, "Language adaptation of multilingual phone models for vocabulary independent speech recognition tasks," in *ICASSP*, vol. 1, 1998, pp. 417–420.

[31] A. Stolcke, F. Grezl, M.-Y. Hwang, X. Lei, N. Morgan, and D. Vergyri, "Cross-domain and cross-lingual portability of acoustic features estimated by multilayer perceptrons," in *ICASSP*, 2006, pp. 321–324.

[32] S. Thomas, S. Ganapathy, and H. Hermansky, "Cross-lingual and multi-stream posterior features for low resource LVCSR systems," in *Interspeech*, 2010.

[33] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký, "Probabilistic and bottleneck features for LVCSR of meetings," in *ICASSP*, 2007.

[34] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in *ICASSP*, 2012.

[35] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR," in *IEEE SLT Workshop*, 2012.

[36] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross language knowlege transfer using multilingual deep neural network with shared hidden layers," in *Proc. ICASSP*, 2013.

[37] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep neural networks," in *ICASSP*, 2013, pp. 7319–7323.

[38] N. Vu, W. Breiter, F. Metze, and T. Schultz, "An investigation on initialization schemes for multilayer perceptron training using multilingual data and their effect on ASR performance," in *Interspeech*, 2012, pp. 2586–2589.

[39] J.-T. Huang, "Semi-supervised learning for acoustic and prosodic modeling in speech applications," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2012. [Online]. Available: http://www.isle.illinois.edu/sst/pubs/2012/huang12thesis.pdf

[40] J.-T. Huang and M. Hasegawa-Johnson, "On semi-supervised learning of Gaussian mixture models for phonetic classification," in *NAACL HLT Workshop on Semi-Supervised Learning, 2013*.

[41] P. Huang and M. Hasegawa-Johnson, "Cross-dialectal data transferring for Gaussian mixture model training in Arabic speech recognition," *4th International Conference on Arabic Language Processing*, pp. 119–123, 2012.

[42] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[43] Özgül Salor and M. Demirekler, "On developing new text and audio corpora and speech recognition tools for the Turkish language," in *Interspeech*, 2002, pp. 349–352.

[44] J. L. Hieronymus, "ASCII phonetic symbols for the world's languages: WORLDBET," *J. Int. Phonetic Association*, 1993.

[45] R. Gopinath, "Maximum likelihood modeling with Gaussian distributions for classification," in *ICASSP*, 1998, pp. 661–664.

[46] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language.*, vol. 12, pp. 75–98, 1997.

[47] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 11, pp. 1641–1648, November 1989.

[48] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *IEEE ASRU Workshop.*, 2011.

[49] T. Schultz and A. Waibel, "Language independent and language adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, pp. 31–51, Aug 2001.

[50] C. Liu, P. Jyothi, H. Tang, V. Manohar, R. Sloan, T. Kekona, M. Hasegawa-Johnson, and S. Khudanpur, "Adapting ASR for under-resourced languages using mismatched transcriptions," in *ICASSP*, 2016, pp. 5840–5844.

[51] K. Vesely, M. Hannemann, and L. Burget, "Semi-supervised training of deep neural networks," in *IEEE ASRU Workshop.*, 2013, pp. 267–272.

[52] K. Knill, M. J. F. Gales, A. Ragni, and S. Rath, "Language independent and unsupervised acoustic models for speech recognition and keyword spotting," in *Interspeech*, 2014.

[53] K. Yu, M. Gales, and P. Woodland, "Unsupervised adaptation with discriminative mapping transforms," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 17, no. 4, pp. 714–723, May 2009.

[54] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks - Studies on speech recognition tasks," in *Int. Conf. Learn. Rep.*, 2013.

[55] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul 1997.

[56] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *ICASSP*, 2013, pp. 6965–6969.

[57] S. Scanzio, P. Laface, L. Fissore, R. Gemello, and F. Mana, "On the use of a multilingual neural network front-end," in *Proc. Interspeech*, 2008, pp. 2711–2714.

[58] K. Veselý, M. Karafiat, F. Grezl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Proc. IEEE SLT*, 2012, pp. 336–341.

[59] Z. Tuske, J. Pinto, D. Willett, and R. Schluter, "Investigation on cross- and multilingual MLP features under matched and mismatched acoustical conditions," in *Proc. ICASSP*, 2013, p. 7349–7353.

[60] H. Su and H.Xu, "Multi-softmax deep neural network for semi-supervised training," in *Interspeech*, 2015.

[61] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *arXiv:1503.02531*, 2015.

[62] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *NIPS*, 2014.

[63] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria," in *Interspeech*, 2014.

[64] W. Chan, N. R. Ke, and I. Lane, "Transfering knowledge from a RNN to a DNN," in *Proc. Interspeech*, 2015, pp. 3264–3268.

[65] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *Proc. ICASSP*, 2017, pp. 4820–4824.

[66] K. Markov and T. Matsui, "Robust speech recognition using generalized distillation framework," in *Interspeech*, 2016, pp. 2364–2368.

[67] S. Watanabe, T. Hori, J. L. Roux, and J. Hershey, "Student-teacher network learning with enhanced features," in *ICASSP*, 2017.

[68] T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono, "Domain adaptation of DNN acoustic models using knowledge distillation," in *Proc. ICASSP*, 2017.

[69] J. Li, M. Seltzer, X. Wang, R. Zhao, and Y. Gong, "Large scale domain adaptation via teacher-student learning," in *Interspeech*, 2017.

[70] J. Cui, B. Kingsbury, B. Ramabhadran, G. Saon, T. Sercu, K. Audhkhasi, A. Sethy, M. Nussbaum-Thom, and A. Rosenberg, "Knowledge distillation across ensembles of multilingual models for low-resource languages," in *Proc. ICASSP*, 2017, pp. 4825–4829.

[71] Y. Chebotar and A. Waters, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Proc. Interspeech*, 2016, p. 3439–3443.

[72] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, and S. Vitaladevuni, "Model compression applied to small-footprint keyword spotting," in *Proc. Interspeech*, 2016, p. 1878–1882.

[73] J. Li, R. Zhao et al., "Developing far-field speaker system via teacher-student learning," in *Proc. ICASSP*, 2018.

[74] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. ICASSP*, 2013, pp. 7893–7897.

[75] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008, pp. 1096–1103.

[76] A. L. Mass, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Interspeech*, 2012, pp. 22–25.

[77] Y. Qian, T. Tan, and D. Yu, "An investigation into using parallel data for far-field speech recognition," in *ICASSP*, 2016, pp. 5725–5729.

[78] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked autoencoders," in *ICASSP*, 2013, pp. 3377–3381.

[79] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *arXiv:1412.6596*, 2014.

[80] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layerwise training of deep networks," in *Adv. in Neural Information Processing Systems*, 2006, pp. 153–160.

[81] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Proc. ICASSP*. IEEE, 2015, pp. 4280–4284.

[82] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. ASRU*. IEEE, 2015, pp. 167–174.

[83] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *CoRR*, vol. abs/1508.01211, 2015. [Online]. Available: http://arxiv.org/abs/1508.01211

[84] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Proc. Interspeech*, 2017, pp. 939–943.

[85] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, D. Seetapun, A. Sriram et al., "Exploring neural transducers for end-to-end speech recognition," *arXiv preprint arXiv:1707.07413*, 2017.

[86] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *Proc. Interspeech*, 2017.

[87] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "Flat-start single-stage discriminatively trained HMM-based models for ASR," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 11, pp. 1949–1961, June 2018.

[88] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP*, 2018.

[89] T. N. Sainath, C.-C. Chiu, R. Prabhavalkar, A. Kannan, Y. Wu, P. Nguyen, and Z. Chen, "Improving the performance of online neural transducer models," *arXiv preprint arXiv:1712.01807*, 2017.

[90] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, vol. abs/1211.3711, 2012. [Online]. Available: http://arxiv.org/abs/1211.3711

[91] L. Lu, X. Zhang, K. H. Cho, and S. Renals, "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition," in *Proc. Interspeech*, 2015.

[92] H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," *arXiv preprint arXiv:1610.09975*, 2016.

[93] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-Transducer," in *Proc. ASRU*, 2017.

[94] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014. [Online]. Available: http://arxiv.org/abs/1412.5567

[95] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Proc. Interspeech*, 2015.

[96] N. Kanda, X. Lu, and H. Kawai, "Maximum a posteriori based decoding for CTC acoustic models," in *Proc. Interspeech*, 2016, pp. 1868–1872.

[97] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, "Advances in all-neural speech recognition," in *Proc. ICASSP*, 2017. [Online]. Available: https://www.microsoft.com/en-us/research/publication/advances-neural-speech-recognition/ pp. 4805–4809.

[98] H. Liu, Z. Zhu, X. Li, and S. Satheesh, "Gram-CTC: Automatic unit selection and target decomposition for sequence labelling," *arXiv preprint arXiv:1703.00096*, 2017.

[99] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Nahamoo, "Direct acoustics-to-word models for English conversational speech recognition," in *Proc. Interspeech*, 2017.

[100] J. Li, G. Ye, R. Zhao, J. Droppo, and Y. Gong, "Acoustic-to-word model without OOV," in *Proc. ASRU*. IEEE, 2017.

[101] I. Bazzi, "Modelling out-of-vocabulary words for robust speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[102] B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq, "Transcription of out-of-vocabulary words in large vocabulary speech recognition based on phoneme-to-grapheme conversion," in *Proc. ICASSP*, vol. 1, 2002, pp. I–861.

[103] A. Yazgan and M. Saraclar, "Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition," in *Proc. ICASSP*, vol. 1, 2004, pp. I–745.

[104] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," in *Proc. Interspeech*, 2005, pp. 725–728.

[105] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny, "Building competitive direct acoustics-to-word models for English conversational speech recognition," in *Proc. ICASSP*, 2018.

[106] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. International Conference on Machine Learning*, 2009.

[107] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, 2013.

[108] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural networks based language model," in *Proc. Interspeech*, 2010, pp. 1045–1048.

[109] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[110] L. Lu, X. Zhang, and S. Renais, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *Proc. ICASSP*, 2016, pp. 5060–5064.

[111] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[112] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," *arXiv preprint arXiv:1706.02737*, 2017.

[113] S. Toshniwal, H. Tang, L. Liu, and K. Livescu, "Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition," in *Proc. Interspeech*, 2017, pp. 3532–3536.

[114] O. Siohan, "CTC training of multi-phone acoustic models for speech recognition," in *Proc. Interspeech*, 2017, pp. 709–713.

[115] W. Chan, Y. Zhang, Q. Le, and N. Jaitly, "Latent sequence decompositions," *arXiv preprint arXiv:1610.03035*, 2016.

[116] Y. Wu, M. Schuster, Z. Chen, , Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[117] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixture of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[118] M. I. Jordan and R. A. Jacobs, "Hierarchical mixture of experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[119] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems," *Neural Networks*, vol. 12, no. 7, pp. 1131–1141, 1999.

[120] A. Das, J. Li, C. Liu, and Y. Gong, "Universal acoustic modeling using neural mixture models," in *Proc. ICASSP*, submitted for publication.