# MULTI-TASK AND TRANSFER LEARNING IN LOW-RESOURCE SPEECH RECOGNITION

by

Josh Meyer

---

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF LINGUISTICS

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2019

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Joshua Meyer, titled Multi-Task and Transfer Learning in Low-Resource Speech Recognition and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

[illegible] Date: July 26, 2019
Thomas Bever

[illegible] Date: 7/26/2019
Mihai Surdeanu

[illegible] Date: 7/26/19
Michael Hammond

[illegible] Date: 7/26/19
Clayton Morrison

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

We hereby certify that we have read this dissertation prepared under our direction and recommend that it be accepted as fulfilling the dissertation requirement.

[illegible] Date: July 26, 2019
Thomas Bever
Dissertation Committee Co-Chair
Department of Linguistics

[illegible] Date: 7/26/2019
Mihai Surdeanu
Dissertation Committee Co-Chair
Department of Computer Science

# Acknowledgments

Thank you to my family for always being there for me: Mom, Dad, Jon, Joe.

Спасибо тебе, Оксана, за то, что всегда верила в меня.

Thank you to my committee for keeping me on track: Tom Bever, Mihai Surdeanu, Mike Hammond, and Clay Morrison. Thanks to all the folks in Tucson who taught me how to find research funding: Georgia Ehlers, Dorian Voorhees, and Shelley Hawthorne-Smith. Thanks to my benevolent employers in the Linguistics Department: Andrew Carnie, Diana Archangeli, Diane Ohala, and Amy Fountain.

Thank you to all my friends in Tucson who kept me happy: Nick Kloehn, Mary-Caitlyn Valentinsson, Dra. Megan Figuerora, Nick Granum, Shiloh Drake, Rolando Coto, Kevin Schluter, Bill Cotter, Becky Sharp, Gus Hahn-Powell, Dane Bell, Emily Bell, Ryan Smith, Jaime Parchment, Yuan-Lu Chen, David Ellison, Genêt Klasek, Joe Dupris, Shannon Grippando, Sam Johnston, Megan Kittleson, and Michael Capizzi.

Thank you to the amazing folks at the American University of Central Asia, for welcoming me over the years: especially Angie Popova and Svetlana Jacquesson. Thanks to Gulnara Shabdanova, for both being a friend and for playing a huge hand in my research in Kyrgyzstan.

Thank you to the Mozilla Machine Learning Research Group for giving me the opportunity to work with all kinds of great languages and share my code Openly with the world: Kelly Davis, Reuben Morais, Alexandre Lissy, Tilman Kamp, Eren Gölge. A special thanks to Kelly for making it all possible.

Thank you to those gave thoughtful feedback on drafts of the disseratation: Guangpu Huang, Joan Bajorek, Sebastian Ruder, Michael Capizzi, and Sam Johnston.

Thank you to Fran Tyers for inspiring me to do more, to make Open data and code, and for good times in Moscow with хачапури. Thanks to Nick Howell for good conversations on neural nets and privacy.

Thank you to LIMSI and the CNRS for hosting me as a visiting researcher: Lori Lamel and Jean-Luc Gauvain.

Lastly, thank you to the organizations that have funded me over my graduate research career: the National Science Foundation*, the American University of Central Asia, the French Embassy to the United States, the Internationl Speech and Communication Association, the U.S. Department of State Bureau of Intelligence and Research, and the University of Arizona Social and Behavioral Sciences Research Institute.

*This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (DGE-1746060). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

# Contents

[†]This chapter is a modified version of a paper co-authored by Josh Meyer, Francis Tyers, Reuben Morais, and Kelly Davis.

## Abstract

This thesis investigates methods for Acoustic Modeling in Automatic Speech Recognition, assuming limited access to training data in the target domain. The Acoustic Models of interest are Deep Neural Network Acoustic Models (in both the Hybrid and End-to-End approaches), and the *target domains* in question are either different languages or different speakers. Inductive bias is transfered from a source domain during training, via *Multi-Task Learning* or *Transfer Learning*.

With regards to Multi-Task Learning, Chapter (5) presents experiments which explicitly incorporate linguistic knowledge (i.e. phonetics and phonology) into an auxiliary task during neural Acoustic Model training. In Chapter (6), I investigate Multi-Task methods which do not rely on expert knowledge (linguistic or otherwise), by re-using existing parts of the Hybrid training pipeline. In Chapter (7), new tasks are discovered using unsupervised learning. In Chapter (8), using the "copy-paste" Transfer Learning approach, I demonstrate that with an appropriate early-stopping criteria, cross-lingual transfer is possible to both large and small target datasets.

The methods and intuitions which rely on linguistic knowledge are of interest to the Speech Recognition practitioner working in low-resource domains. These same sections may be of interest to the theoretical linguist, as a study of the relative import of phonetic categories in classification. To the Machine Learning practitioner, I hope to offer approaches which can be easily ported over to other classification tasks. To the Machine Learning researcher, I hope to inspire new ideas on addressing the small data problem.

# Chapter 1

# Introduction

## 1.1 Objectives of Dissertation

The main goal of this dissertation is to answer the following question: *How can we best exploit small target datasets (and larger, source domains) to train Acoustic Models for Automatic Speech Recognition via Multi-Task Learning and Transfer Learning?*

There are many interesting approaches to working with small datasets in Automatic Speech Recognition (ASR). In the following studies, I have chosen to work specifically with the Multi-Task Learning (MTL) and Transfer Learning (TL) approaches because in their general forms they can be applied to data beyond just speech or audio, and as such the findings of this work may find applications in other areas of machine learning.

Firstly, I investigate methods of label-generation which build off linguistic knowledge (i.e. phonetics and phonology). These methods can be used to train an Acoustic Model for any language. Secondly, I investigate methods of label-generation which do not rely on expert knowledge (linguistic or otherwise). I demonstrate that it is possible to automatically generate new label-spaces which are useful for Multi-Task acoustic modeling by re-using existing parts of the traditional ASR pipeline or by discovering new labels with unsupervised learning. Lastly, using the well-known "copy-paste" Transfer Learning approach, I demonstrate that with an appropriate early-stopping criteria cross-lingual transfer is possible to both large and small target datasets. The chapter on Transfer Learning ends with experiments on model interpretation which get at the linguistic relevance of the embeddings learned at various layers of the model.

The methods and intuitions which rely on linguistic knowledge are of interest to the ASR practitioner who wants to train Acoustic Models for low-resource domains. These same sections are of interest to the theoretical linguist, as a study of relative import of phonetic categories in classification. To the machine learning practitioner, I hope to offer approaches here which can be directly (or with some minor adjustments) ported over to other classification tasks. To the machine learning researcher, I hope to inspire new ideas on addressing the small data problem.

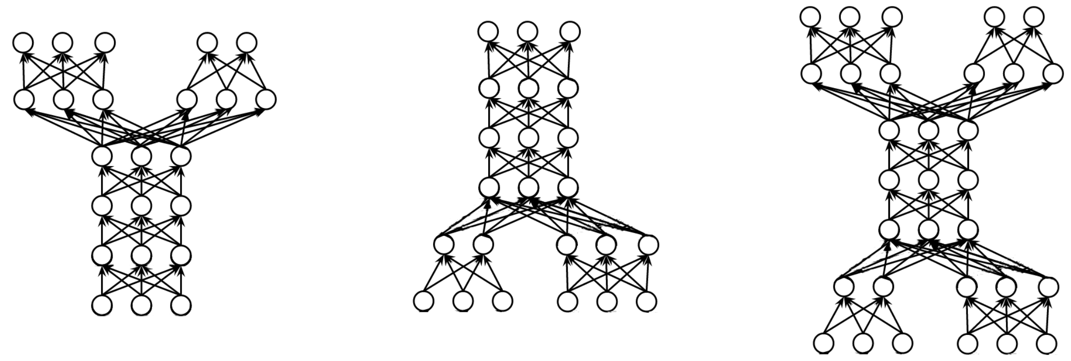
### 1.1.1 Multi-Task Learning

The term "Multi-Task Learning" is used to describe both a model training procedure and a model architecture.[20] The architecture definition is straightforward: a Multi-Task model is a model which can perform multiple tasks. A task is defined here as a mapping of data to labels, between a certain data domain and label domain. With regards to the training procedure definition, a model is trained with MTL when a subset of parameters are updated in parallel, and another subset of parameters are updated individually for each task. In fact, it is possible that the trained MTL model may only perform one task. In this scenario, the auxiliary tasks exist solely to exert inductive bias during training. These auxiliary tasks serve as "training wheels" which are removed once the main task is ready to perform on its own. The current research takes this approach to Multi-Task Learning: multiple tasks are trained in parallel, but at test time only the main task is used for classification.

MTL models come in all shapes and sizes, but they can be boiled down to one of three main categories: (1) multi-label, (2) multi-data, or (3) multi-label and multi-data. These groups are shown for neural networks in Figure (1-1).[*] For neural nets, we have a set of *shared hidden layers*, and some *task-specific* input or output layers.

The Multi-Task studies here deal with multi-label models: *one input layer* and *multiple output layers*. In this way, MTL flips the small data problem on its head –

[*]These figures are adapted from Heigold et al. (2013), and reproduced with permission from the first author. Variations on these neural nets will appear throughout the dissertation.



(a) Multi-Label (b) Multi-Data (c) Multi-Label + Multi-Data

Figure 1-1: Possible Multi-Task Architectures

creating more labels instead of finding more data. Given multiple sets of labels for some dataset, MTL trains feature extractors which are biased towards task-independent representations of the data. An intuitive example of this kind of multi-label scenario is the classification of a picture of a German Shepard as a `German Shepard`, a `canine`, and a `mammal`. These three labels all belong to the same training data example: a particular photograph of a German Shepard. This scenario assumes the researcher has access to multiple sets of labels for the data, however this is usually not the case.

I investigate methods of generating labels for existing datasets which can be then used in a Multi-Task framework. The architecture of the MTL Deep Neural Network (DNN) is not manipulated, and as such any improvements in model performance are attributable to the choices of labels alone. The techniques developed here for label generation are all language-independent. The first experiments are specific to working with speech data, whereas last experiments are applicable to any classification task, regardless of data-type.[†]

### 1.1.2 Transfer Learning

The term "Transfer Learning" refers to a general model training procedure. A model is trained on some task in some domain, and then a subset of the parameters are used

[†]This assumes the data can be represented numerically as N-dimensional feature vectors.

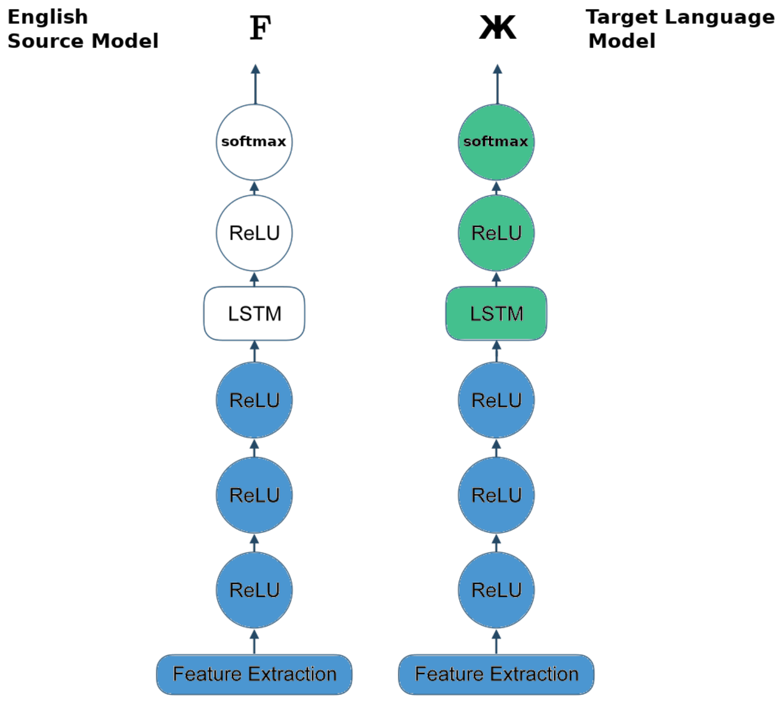


Figure 1-2: "Copy-Paste" Transfer Learning. In this example, a subset of model parameters trained on a language using the Latin alphabet is used to create a new model for a new language, written with the Cyrillic alphabet. Blue model parameters have been copied from the source model, and the green model parameters have been re-initialized from scratch.

to initialize a new model for a new domain (or a new task). The subset of parameters may happen to be the exact set of all model parameters. This is possible in the case that the input features and output labels for the target task exist in the same dimensionality as the source task. For the work investigated in this dissertation, all the input features have the same dimensionality, but the output labels are different. As such, the TL approach here requires initializing (from scratch) the output weight matrix and biases for the target task. Given that the TL approach involves copying some pre-trained weights and transfering them to a new model, I will refer to this approach as "copy-paste" Transfer Learning. A figure representing this "copy-paste" transfer may be found in Figure 1-2.

## 1.2 Statement of the Problem

The problem addressed in this dissertation is the following: *Current methods for training speech recognition systems require massive collections of labeled data, but most use-cases have little — if any — available data.*

A large collection of recorded and transcribed speech (i.e. a spoken corpus) is required to create a speech recognizer for a new application. It is more than likely however, that such a corpus does not exist for the application of interest. The application could be for a new language (e.g. the Kyrgyz language), for a specific group of speakers (e.g. English speakers from Southern Wales), or for a new recording setting (e.g. far-field speech from lab meetings in some conference room). In all three of these cases, a speech recognizer trained on speech from any other language, speakers, or recording conditions will perform poorly. This is because the actual data of interest, the data on which our recognizer should be modeled, is not represented in the training set. Typically researchers and practitioners work around the problem by collecting as much data as possible from similar conditions. However, this approach does not solve the problem. A model trained on a massive corpus of American English will still perform worse for a single speaker compared to a model trained on a corpus of only that individual.

Data from the target domain is best for our models but, as noted earlier, it is not easy to find. The typical approach to building a new application is to record and transcribe a new dataset from scratch. The problem is: *recording a large corpus of new audio for every new use-case is not a scalable approach.* This is especially relevant to work involving artificial neural networks, which are particularly data-hungry. Given standard training approaches for these massive models, data collection becomes the major bottleneck. Collecting and creating data resources is a time-consuming and expensive endeavor. Collecting acoustic data is especially difficult because the audio must be acquired and then transcribed. Unlike text data which can be easily scraped from the web, scraping the web for audio is not a viable option. Transcription of casual, spontaneous speech requires upwards of five hours of work per hour of speech.

To create a state-of-the-art ASR system, the model requires approximately 2,000 hours of transcribed audio. A corpus of that size requires 10,000 hours of person hours of transcription. At a salary of merely \$10 dollars an hour, it will cost \$100,000 dollars to transcribe an adequately-sized speech corpus.[‡]

Given this problem of insufficient data for current training methods, we can conceptualize two main approaches to tackle the problem:

1. Change the Data
   (a) create large, labeled speech datasets for all the world's languages, accents, noise environments, etc.
2. Change the Technique
   (a) develop techniques which can make use of *much* smaller *labeled* corpora
   (b) develop techniques which can make use of purely *unlabeled* speech corpora

Of these two approaches, (1) is not feasible. There currently exist nearly 7,000 languages in the world, and of those, the top 397 languages have more than 1 million native speakers each [91]. To create large, labeled corpora for even the most widely spoken languages of the world would take far too much time and money.

Beyond the practical challenges concerning dataset creation, the problem of missing data will not disappear even if we cover all the existing languages, dialects, and accents in the world. Languages are continually changing and accents are constantly merging and diverging. As such, a speech recognition system trained from a given dataset will become out-dated within just one generation of new speakers. Furthermore, even if we chose to ignore the language change issue, speech corpora are recorded in a finite set of environments (e.g. in the office, in a park, over the phone), and the recognizer will perform significantly worse on speech recorded in any new environment. Current approaches would suggest that we train a new model for each environment, as is current practice with broadcast news speech versus telephone conversations [140, 55].

[‡]An excellent counter example to this problem is the Common Voice project from Mozilla, which crowd-sources both the recordings and the transcription verification.

Given that the number of new environments is infinite, the generalization problem is here to stay.

## 1.3 Context of the Problem

The broader context of the stated problem is the following: *The small data problem is found everywhere in machine learning, not just speech recognition.*

The issues of modeling low-resource languages, unknown speakers, and new environmental noise may seem very specific to ASR, but in fact, these are core issues of machine learning faced in every application of classification, regression, and pattern matching (cf. survey in [107]). In the well-established field of computer vision, much work in object recognition has been devoted to this problem, under the name *domain adaptation* (cf. a recent survey in [112]). Likewise, the problem surfaces in NLP under the names *domain adaptation* and *transfer learning* [35, 107, 16]. Taking a step back to a higher level of abstraction, all of these issues (ASR of different accents, classification of photos taken with different cameras, POS tagging for different writing styles) fall within one major issue in machine learning.

> *A classifier will perform best on data from its training set, and worse on all other data.*

This is because the classifier overfits to the training data and learns representations which ignore useful information about the task at hand and encode misleading information about dataset noise. Using this framing of the problem (mismatch between testing and training data), we are able to attack the specific issues found in Automatic Speech Recognition for low-resourced languages by drawing inspiration from research in other machine learning fields such as NLP and computer vision.

The main ideas which inspired the approach in this thesis come from *Multi-Task learning* [20], *domain adaptation*, *transfer learning* [120], *learning to learn* [145], *curriculum learning* [14], *domain knowledge integration* [62], and *representation learning* [13].

## 1.4 Research Questions

The main research questions investigated in this thesis are the following:

1. *How can linguistic theory be used to create new labels for speech data (for training via Multi-Task Learning)?*

2. *How can new labels be automatically discovered for speech data without expert domain knowledge (for training via Multi-Task Learning)?*

3. *How can Transfer Learning in ASR be truly general (for cross-lingual transfer scenarios)?*

The first question is relevant to ASR practitioners as well as theoretical linguists. The second and third questions are of interest to ASR practitioners as well as the general machine learning community, because the techniques can be extended to any other classification or regression problem.

## 1.5 Methodology

The two main methodologies of the dissertation may be summarized as follows: *(1) Building on the traditional Hybrid speech recognition pipeline, Multi-Task Acoustic Models are trained where auxiliary tasks are either crafted from expert linguistic knowledge or automatically generated. (2) Using a trained End-to-End ASR for a source language (English), "copy-paste" Transfer Learning is investigated on an open, massively multilingual speech corpus.*

### 1.5.1 Multi-Task Learning

Traditional Automatic Speech Recognition is composed of two main phases: first, the audio signal is transcribed at the phonetic level; second, the resulting phonetic transcription is transduced into a string of words. These two steps are modulated by two independently trained systems: the Acoustic Model and the decoding graph, respectively. The Multi-Task experiments here relate to the first system: the acoustic model. The Acoustic Model in Hybrid speech recognition is a Deep Neural Network (DNN) which takes as input some audio signal features and returns as output a probability over phonetic symbols. Like all DNNs, these Acoustic Models require labeled data to be trained via backprop.

I examine DNN Acoustic Models which are trained with multiple output layers in order to update a set of shared hidden layers. These extra output layers are referred to as **auxiliary tasks**, **new tasks**, or **new labels**. Figure (1-3) displays a diagram of the DNN architecture used in the Multi-Task studies.[§]

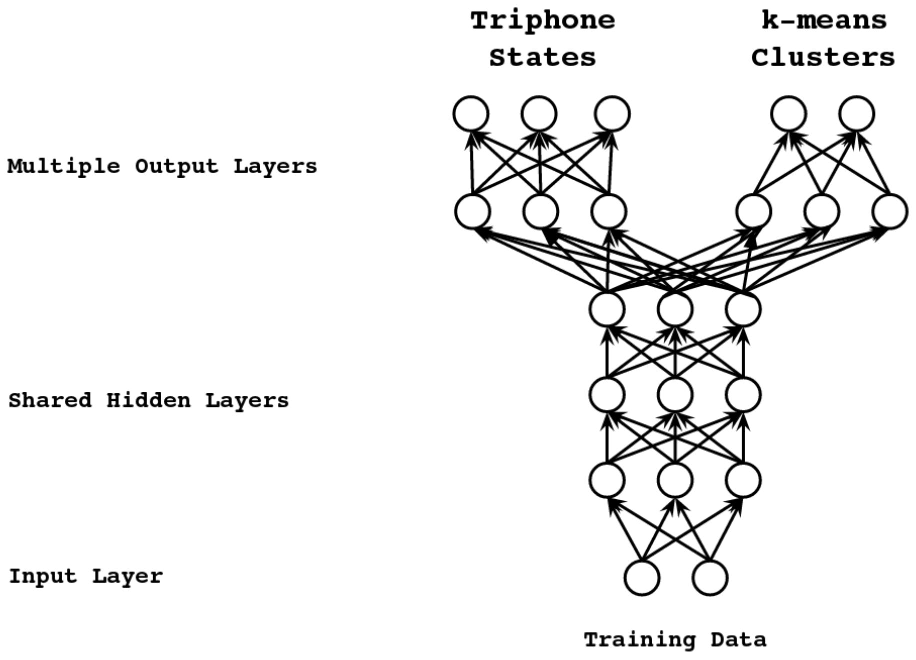


Figure 1-3: Multi-Task DNN Acoustic Model

The number of input nodes on these DNNs correspond to the dimensionality of audio features, and the number of nodes on each output layer corresponds to the number of targets for that particular label set. In the traditional case (i.e. Single-Task Learning) the dimensionality of the output layer merely reflects the number of context-dependent phonemes (i.e. triphones), whereas the dimensionality of the input layer reflects standard audio features (e.g. MFCCs). During Multi-Task learning, all

[§]Figure reproduced from Heigold et al. (2013).

tasks (i.e. separate output layers) are trained in parallel, but during testing only one, main task is used.

### 1.5.2 Transfer Learning

The final experiments in Transfer Learning investigate an End-to-End Transfer Learning approach for Automatic Speech Recognition which bypasses the need for linguistic resources or domain expertise. Certain layers are copied from a trained source model, new layers are initialized for a target language, the old and new layers are stitched together, and the new layers are trained via gradient descent. The effects of fine-tuning the original, copied layers are also investigated. Additionally, the quality of the embedding spaces learned by each layer of the original model are investigated. Specifically, results are given from linguistically motivated logistic regression tasks which are trained on top of feature-spaces at different model depths. Results are presented for transfer-learning from English to the following twelve languages: German, French, Italian, Turkish, Catalan, Slovenian, Welsh, Irish, Breton, Tatar, Chuvash, and Kabyle.

## 1.6 Significance of Dissertation

The significance of this dissertation is twofold: to the ASR community and to the machine learning community. *This dissertation is significant to the ASR community because it demonstrates multiple new avenues for working with low-resource languages. This dissertation is significant to the general machine learning community because it works toward generalizable techniques for training statistical classifiers for any dataset.*

With regards to Multi-Task Learning, while advances have been made in numerous machine learning domains, its use in ASR is currently limited. One of the major blocks for MTL in ASR is that the auxiliary tasks need to be defined by an expert linguist. I demonstrate methods to automatically generate tasks for a given dataset, leading to improvement on the target domain.

Past work on MTL in ASR can be divided into two main categories: monolingual and multilingual. Multilingual MTL acoustic modeling involves training a single DNN with multiple output layers, where each output layer represents targets for a separate language.[75, 71, 59] Monolingual MTL acoustic modeling involves designing multiple tasks for a single language, where each task is linguistically relevant (e.g. vocal cord vibration, place of articulation, manner of articulation). [12, 76, 135, 25, 24] This dissertation integrates advances from both these veins of research (i.e. monolingual and multilingual) in addition to automatizing the generation of the auxiliary task labels. With regards to multilingual training, I take a new approach to representation of the source language. There have been previous efforts to represent source languages via linguistic categories, but these attempts have almost always created a larger label-space, projecting the source language labels onto some form of multilingual alphabet. These attempts have require linguistic mappings from source to target language, or from source language to multilingual representation. The techniques here do not require any mapping between languages. Rather, I use linguistic concepts to create a more abstract label set from the source language. With regards to monolingual, Multi-Task speech recognition, I project a language's labels into a lower dimensional space, but with more dimensions than previous work. My approach removes one linguistic dimension, instead of mapping every phoneme onto a single dimension. The tasks generated automatically (i.e. without any linguistic knowledge) are perhaps the most important contribution of this dissertation. Continuing work in this direction holds much promise — because truly unsupervised auxiliary task generation can be ported to any new classification task.

With regards to Transfer Learning in ASR, the final chapter is important because it extends previous results to multiple languages, using open data and an open, pre-trained source model. To my knowledge, this is the first paper reporting End-to-End ASR results from a free and open model, which is important for replicability.

## 1.7 Limitations of Dissertation

The main limitations of this work are the following: *The only measure of interest is the accuracy of the final, trained model. The training time and space requirements were largely ignored.*

Some of the methods developed here, if run on larger datasets, may require long training times. For the current study this is not an issue because the focus is small data sets, but it does face an issue for the generalization of these methods to all classification problems. In the approach taken here, every additional auxiliary task requires an additional iteration over the entire dataset.

## 1.8 Road Map

This dissertation is written for the reader who has at least some familiarity with Deep Neural Networks. Familiarity of Automatic Speech Recognition, Multi-Task Learning, and Transfer Learning are beneficial, but not required.

Chapter (2) contains an overview of Automatic Speech Recognition. Chapter (3) contains an overview of past traditional approaches in ASR to deal with small or unseen data. Chapter (4) delivers a brief overview of Multi-Task Learning as well as past work in ASR using MTL. Chapter (5) presents experiments and results from linguistically defined auxiliary tasks. Chapter (6) presents results results from experiments where auxiliary tasks are derived as a by-product in the traditional ASR pipeline. Chapter (7) presents results from MTL experiments where the auxiliary task labels are discovered via unsupervised clustering. Chapter (8) explores a Transfer Learning approach with End-to-End ASR on a massively multilingual corpus. Chapter (9) concludes the dissertation with a discussion and directions for future work.

# Chapter 2

# A Brief Overview of ASR

## 2.1 Introduction

This chapter outlines the training and testing procedures in standard Automatic Speech Recognition (ASR) pipelines. The overview will provide the reader with a technical grounding in ASR.

All research exists in some historical context, answering pressing questions of the timee while making use of and reacting to existing technologies. Speech Recognition research in particular is a field which has clearly grown in directions defined by military, academic, and commercial influence. Early work on Speech Recognition reflected the needs of telecommunications companies. Then came a wave of interest from the US Department of Defense, and most recently the four tech giants – Google, Amazon, Facebook, and Apple. While all these initiators pushed researchers in different directions, they all share one common goal: to make ASR more human-like.

## 2.2 1952: Isolated Word Systems: Bell Labs

Most of the early work on ASR in the 50's and 60's focused on isolated word, speaker-dependent speech recognition. This research was lead by R&D labs at Bell Labs and the NEC Corporation, but also academic labs such as MIT Lincoln Labs. [36, 42, 46, 103] At this time there were also attempts at phoneme-level transcription, but these were less successful (the task is inherently harder). [105]

The typical use case was a single adult male carefully reading off single digits into a microphone. One of the very first demonstrations reported accuracy rates of up to 99% on isolated digit recognition [36]. This system relied on approximations of the formant frequencies to recognize entire words. There was no model of syllables or consonants or vowels or any kind of sub-word unit in these systems. The word was treated as a single unit, and during classification all words were compared to each other to find the best match. An example set of templates (from the original Bell Labs paper) is shown in Figure (2-1).

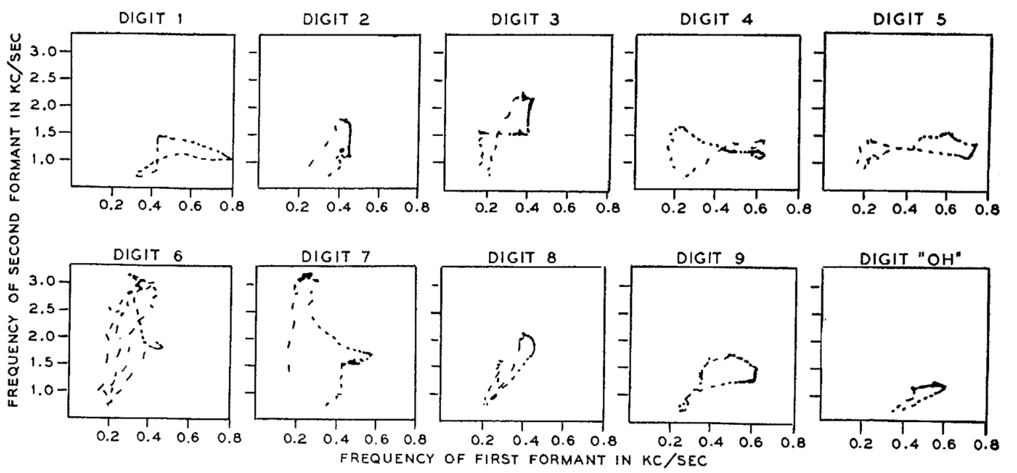


FIG. 2. Photographs of formant 1 *vs* formant 2 presentation of the digits. Trace interruption period = 10 ms. Recognition criteria depend upon significant differences in these shapes and upon their relative duration in the frequency space.

Figure 2-1: Template-matching of Spoken Digits: Davis et al. 1952

This work, along with most others of this time period, relied on a template-matching framework. An exemplar of each word was saved to disk (for each speaker ideally), and when a user spoke a new, unknown word into the microphone, the computer compared that audio with all the exemplars it had on file for that speaker. The closest match was returned back, and when recognizing a set of ten digits, this worked surprisingly well. The Bell Labs system worked in the following way:

1. FEATURE EXTRACTION

    (a) Two frequency ranges of the audio are extracted, which roughly correlate to the first two formants of speech.

    (b) These two formants are plotted on an `x` vs. `y` axis in time.

2. TEMPLATE MATCHING

    (a) The 2-D plot from new audio is compared to each of 10 exemplars on file, and closest match is returned.

It may sound like Speech Recognition was off to a great start with accuracy rates near 99%, but this early template-matching approach could not be extended for a few reasons. These approaches relied on acoustic properties of vowels in a whole-word template-matching scheme which requires a representation of each word saved on disk. If extended to larger vocabularies, this paradigm would run into major issues in time and space complexity. If the system needed to recognize $1,000$ words instead of just 10, the time needed to compare new input to each of the $1,000$ exemplars would be prohibitive. Additionally, the space on disk would increase with every additional word. Therefore, both time and space complexity are proportional to the number of words in the lexicon. This does not bode well for scaling to all words in a language. Aside from implementation issues with this approach, a more fundamental limitation is the use of whole words as templates, and extracting only formant frequencies as features. Two words with similar consonants and identical vowels (e.g. ‘dog’ vs ‘dock’) would be nearly indistinguishable for this system.

The major issues with these digit recognizers stem from the unit of comparison (i.e. what does the template represent?) and the extraction of features (i.e. how is the template stored?). In the following, the unit of comparison changes from the word to the phoneme, and the features extracted change from formants to fine-grained frequency bins, such as Mel-frequency cepstral coefficients (MFCCs).

## 2.3 1971: Constrained Sentence Recognition: ARPA

Speech research soon was boosted into full gear in 1971, when the Advanced Research Program Agency (ARPA) of the US Department of Defense launched the 5-year Spoken Understanding Research (SUR) program. The goal of the program was to “obtain a breakthrough in speech understanding capability that could then be used toward the development of practical man-machine communication systems”.[83] ARPA wanted something that Airforce pilots could control with their voice while their hands were busy steering. The SUR program sponsored four main research groups: two groups from Carnegie-Mellon University, one from Bolt Beranek and Newman Inc., and one from Systems Development Corporation. Probably the most significant result of the ARPA project was James Baker’s 1975 dissertation at CMU, which firmly established the Hidden Markov Model in ASR.[7]

The contestants were given the task of creating a system which could recognize simple sentences from a vocabulary of $1,000$ words with a 10% WER in reasonable time. In order to make a recognizer that could handle sentences instead of isolated words, where the length of that string was unknown to the system, major overhauls of the Isolated Word system were needed. First of all, it was clear that storing an exemplar of each word on disk is not an option with continuous speech systems. In addition to the nearly impossible task of discovering word boundaries from raw audio, the decoding speed would be horrendous (even if word boundaries were found). The machine would have to compare each word to each of the $1,000$ candidate words on disk. The space used on disk would be prohibitive, not to mention the time needed by the user to record every word during speaker enrollment. Modeling whole words became an obvious bottleneck to recognition of continuous speech. As such, the phoneme became the unit of modeling. Figure (2-2) displays a possible decoding graph from CMU’s submission: the Harpy system.[83]

The phoneme is the smallest meaningful speech sound. Every language has a finite set of phonemes, and with this finite set of phonemes all words are created. Typically languages don’t have more than 50 phonemes, and that number will not increase with vocabulary or grammar complexity. Where simple systems had hard limits of 100 or $1,000$ words, with only 50 discrete phonemes there is no upper limit to the number of words a phoneme-based system can recognize. All of the teams in the ARPA project used the phoneme as the unit for modeling speech. At the end of the project, the team at Carnegie Mellon showed best performance with Harpy.[94] Like in Isolated

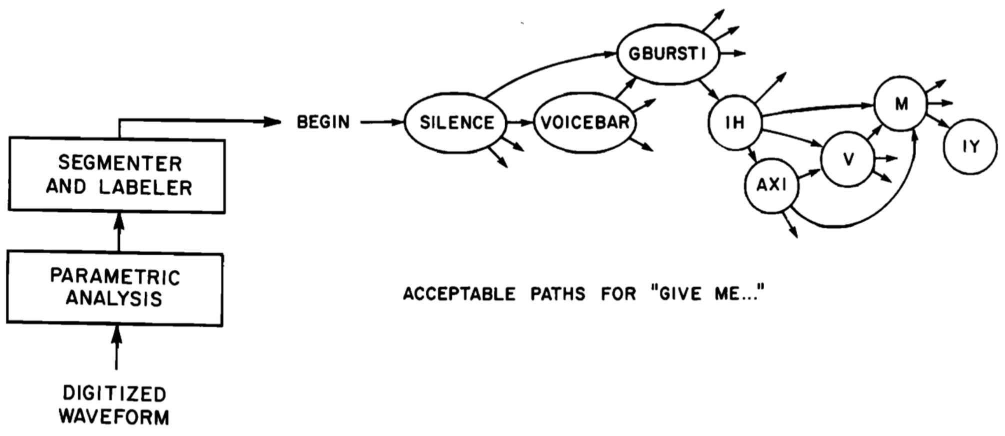


Figure 2-2: Example Decoding Graph from the Harpy system. (Klatt 1977)

Word Recognition, all teams used some kind of template matching, but with phoneme templates instead of word templates. The Harpy system decoded a new utterance in the following way:

1. FEATURE EXTRACTION
    (a) process audio with 5kHz low-pass filter and sample to 10k samples per second
    (b) extract linear prediction coefficients in 10-ms frames with a 10-ms shift
    (c) group together adjacent acoustic segments if similar
2. GRAPH CONSTRUCTION
    (a) a set of 98 phonemes (and diphones) defined by experts
    (b) pronunciations for all words defined
    (c) pronunciations of all accepted sentences in the grammar compiled into one graph ($15,000$ states with self-loops)
3. DECODING
    (a) incoming audio segments compared against 98 templates
    (b) best path (with beam search) returned

Harpy is a speaker-specific system, and the 98 phoneme templates need to be tuned to each speaker. For a new speaker to be enrolled into the system, she must spend about 30 minutes recording example sentences, which are then force-aligned to the graph. This forced-alignment, however, assumes that at least one speaker has already been enrolled, and their 98 phoneme templates are used to align the next speaker's audio. Given that command and control was the target application, a limited vocabulary and limited grammar was reasonable. The user said one short sentence (from a constrained, finite set), and the decoder compared that sentence to a graph of all possible sentences, and returned the closest match. This assumes the user actually said a sentence in the machine's vocabulary. Each user was trained to work with the machine (learn its grammar), and the machine was trained to work with each user (via enrollment).

What these recognizers lacked in flexibility, they gained in accuracy. The machine didn't have to consider more than $1,000$ words and a simple grammar. Furthermore, there was no real issue of noise conditions, because recording and testing would be both in quiet conditions. There was no worry about microphone robustness or sampling-rate issues, because the creators knew exactly beforehand what hardware the recognizer ran on. This approach worked just fine until users wanted something more human-like. First of all, training the recognizer to work for every new user was a hassle. We humans don't need to relearn all sounds in our language when we meet someone new, but these machines did. We humans can understand our friends when we're in an echoey factory or in a small room, but these machines couldn't.

Regardless of the successes of the SUR program, ARPA was disappointed. The best system, Harpy, decoded in about 80x real time. Harpy could not be used in practice, and speeding her up was not a simple task. In his review of ARPA's Speech Understanding Research Program, Dennis Klatt concluded that "all [teams] failed to meet the ARPA goals."[83]

In addition to problem of speed, grammar flexibility was a major concern for making systems that could recognize spontaneous speech. The kinds of sentences recognized by Harpy were determined by a Backus-Naur form (BNF) grammar. This consisted of a set of hand-crafted rules, and was not easily extensible. Even in the 1980's, researchers realized that such a grammar was not a viable option. A major shift was about to take place in the speech recognition world, moving away from template matching and strict grammars towards statistical Acoustic Models and statistical grammars. Instead of hard assignments, a better system would assign a kind of likelihood to the sentence, word, or sound in question.

## 2.4 1994: The GMM-HMM Approach: HTK

Hidden Markov Models (HMMs) were introduced in the 1970's [7], Gaussian Mixture Models (GMMs) [78] and statistical grammars [79] were introduced in the 1980's, but it wasn't until the 1990's that all three parts were combined into one Open Source toolkit: the Hidden Markov Model Toolkit [168].

The Hidden Markov Model Toolkit (HTK) was the first toolkit to incorporate all of the core components of Modern GMM-HMM speech recognition. CMU's Sphinx was a leader in the ASR toolkit game [88], but the first version of CMU's Sphinx used Vector Quantized codebooks to estimate emission probabilities on HMM states, while HTK began with GMMs in 1994. Regardless of HTK and Sphinx's (minor) differences in performance, HTK's extensive documentation (The HTK Book) became the reference of choice for most speech recognition researchers. The first results of the HTK system were reported as a benchmark on DARPA's Resource Management task in 1992 [162]. The standard GMM-HMM pipeline of HTK is outlined below:

1. FEATURE EXTRACTION
    (a) sliding window feature extraction
2. GMM-HMM MONOPHONE TRAINING
    (a) Flat-start alignment
    (b) Baum-Welch re-estimation
3. GMM-HMM TRIPHONE TRAINING

   (a) Phonetic decision tree

   (b) Baum-Welch re-estimation

4. DECODING

   (a) Network compilation

   (b) Viterbi decoding

## FEATURE EXTRACTION

A common goal of all feature extraction in speech recognition is the following: to make speech-information more available for statistical modeling. The raw audio signal is not conducive to modeling speech. Feature extraction enhances the signal-to-noise ratio, but it is more than just a "cleaning" of audio. Raw audio is simply air pressure measured in time. Speech is transmitted via change in pressure, and as such raw audio contains all the information relevant to human speech. However, speech sounds are differentiated by amplitudes in both the frequency and time domain. Figure (2-3) displays audio in raw, waveform (top) and with frequency information made explicit as a spectrogram (below). Feature extraction makes that frequency information available explicitly, instead of relying on the statistical model to infer on its own.

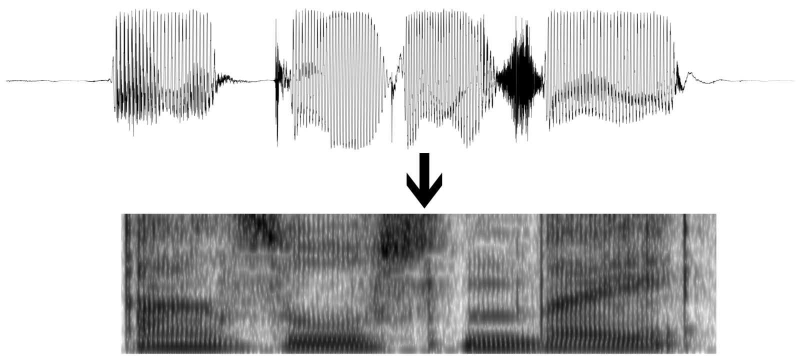


Figure 2-3: Waveform vs. Spectrogram.

The two standard audio feature extraction techniques in HTK (and in ASR in general) are Mel-frequency cepstral coefficients (MFCCs) or perceptual linear prediction (PLPs). Overlapping windows are shifted across the audio, from beginning to end, to extract feature vectors. For each timestep, the audio falling inside the window is subjected to a series of transformations which reduce the dimensionality of the data and highlight speech-specific information. Typically the windows are 25 milliseconds long, and the the shift is 10 milliseconds as shown in Figure (2-4). In this way, enough time is allowed to capture phonetic information inside one window.

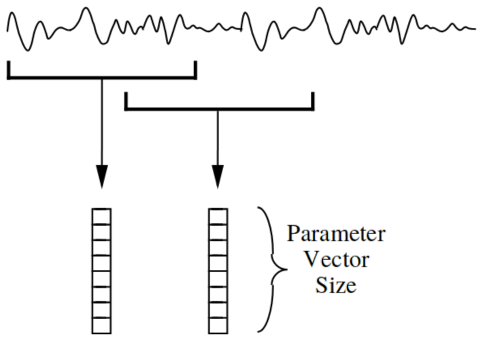


Figure 2-4: Sliding Window Feature Extraction via HTK. (Young 2002)

Ideally, every feature vector should contain enough information to identify the phoneme which was uttered at that timestep.

## MONOPHONE TRAINING

A monophone GMM-HMM Acoustic Model contains (as the name suggests) one model for one phoneme. For a language with 50 phonemes, a monophone system will have 50 separate GMM-HMMs. The number of states in each phoneme's HMM is not specified, and is a matter of researcher discretion. Given that human speech is a purely left-to-right signal, monophone HMMs are defined to be strictly left-to-right, with the addition of self-loops. Abstracting away from some of the implementation specifics, each monophone HMM will be designated as such:

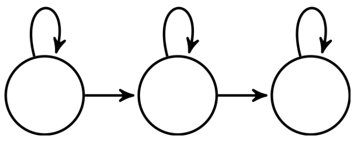
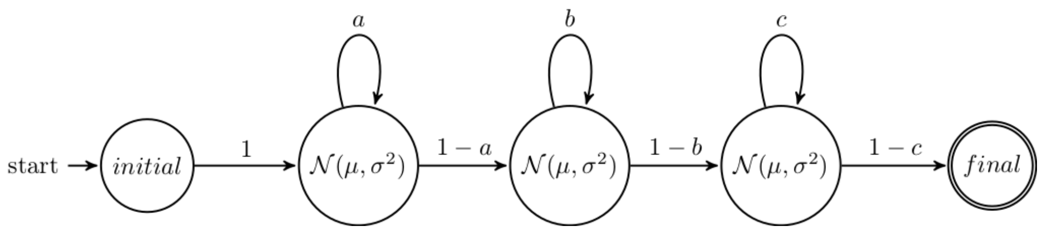


Figure 2-5: Three-state Hidden Markov Model.

The self-loops (as in the Harpy system) nicely model the differences in time which occur in speech. A long, drawn-out vowel will be modeled with the exact same HMM as a short, quick vowel. The longer vowel will just have to pass through more self-loops. Most GMM-HMMs in speech recognition use three states (c.f. Figure (2-5)) because neighboring phonemes blend together. Acoustic features at phoneme boundaries differ significantly from the center of the phoneme. All else being equal, the central state of a monophone should be more consistent than its edges. In addition to modeling co-articulation, three states will allow for the modeling of more complex sounds like affricates or stop consonants, which are composed of multiple acoustic events. For instance, a stop consonant like `[b]` can be broken into three parts (closed voicing, plosive release, aspiration). Each of these three parts is a separate acoustic event, and should ideally be modeled separately. Hidden Markov Models with multiple states allow for fine-grained modeling of a single phoneme into its component parts. However, merely defining the typology of an HMM is not sufficient to use it in speech recognition. HMMs are defined by:

1. a set of states
2. a set of transitions between states
3. the probability of traversing each transition
4. a set of possible observations
5. a probability of each state generating each possible observation

So far we have defined how many states we want (three states per phoneme), we have defined the *kinds* of transitions between states (left-to-right or self-loop), and we have defined the shape of the emission probability of each state (a multivariate Gaussian mixture model). However, what this gets us is merely a set of skeleton GMM-HMMs. For these HMMs to be useful, we need to have accurate transition probabilities (such that the probability of leaving any non-final state == 1) and emissions probabilities (the $\mu, \sigma^2$ for each Gaussian component). That is, we need to estimate the parameters of our HMM-GMMs. Incorporating all the requirements discussed to this point, we have defined the architecture in Figure (2-6). All the relevant parameters are untrained, however.



start
initial
1
$\mathcal{N}(\mu, \sigma^2)$
$a$
$1-a$
$\mathcal{N}(\mu, \sigma^2)$
$b$
$1-b$
$\mathcal{N}(\mu, \sigma^2)$
$c$
$1-c$
final

Figure 2-6: Untrained HMM-GMM

One approach to get these numbers would be to sit down and think really hard about what these numbers should be, filling them in ourselves. We could make self-loops for vowels longer than stop consonants, and we could define the parameters of Gaussians based on some knowledge of acoustic phonetics. However, this is a pretty horrible idea. So instead of trying to guess these values, we use some concepts from machine learning to *learn* them from data. We are going to *estimate our parameters.*

To learn about the acoustics of human speech, we need to collect a lot of it. However, just having access to a huge spoken corpus isn't enough; we need to know which segments of the audio correspond to which states in our GMM-HMMs so that we can update parameters given the appropriate data. While it is certainly possible to align segments of audio to HMM states by hand, it is a very tedious job (examples of this kind of corpus include TIMIT [50] and the Ohio Buckeye corpus [115]).

A much better approach is to automatically align audio segments (i.e. feature vectors) to HMM states. The algorithms for this alignment have been known since the 1960's, and are guaranteed to achieve the best HMM for the given data [10, 122]. This parameter estimation technique is called the Baum-Welch Algorithm, and it falls into the category of Expectation Maximization (EM) algorithms. The technique involves an iterative process in which the model generates a new set of labels for the data, and trains itself on that new dataset. Using the Baum-Welch algorithm, given (1) a speech corpus, (2) transcriptions of the speech, and (3) a pronunciation dictionary, we can estimate all the parameters of our Acoustic Models (i.e. our GMM-HMMs).

## Flat-Start Alignment

In order to successfully train our monophone GMM-HMMs from speech data, we eventually want to have the entire corpus split into segments which correspond to the states of the HMMs. Before we can begin iterations of Baum-Welch, we need a first guess as to what the parameters could be – that is, we need an initial alignment to the data. Flat-start training is a very simple solution to this first-alignment problem.

For Baum-Welch re-estimation to work, we use the parameters of our model to generate new alignments for the model itself. However, in the first step, we have no initialization for the model parameters. We could initialize all parameters randomly, and then start Baum-Welch. This in principle should work, but it would take a long time because the initialization would be nowhere near the data. A better first guess would come not from random parameters, but from the audio itself.

In flat-start training, we assume a few things: (1) utterances, (2) transcriptions for the utterances, and (3) a phonetic transcription for every word in the transcripts. Given these three things, we can construct a phonetic transcription for each utterance. Given a phonetic transcription of an utterance in training, starting from left to right, we assign an equal section of the audio to each phoneme in the transcription. This first left-to-right alignment is our initial guess (i.e. our flat-start). In practice, this works faster than randomly initialized parameters.

```
procedure Compile Utterance HMMs from Transcripts
    for each utterance in corpus do
        Initialize utterance HMM with start state
        for each word in utterance do
            Lookup pronunciation of word in lexicon
            for each phoneme in word do
                Lookup monophone HMM definition for phoneme
                Concatenate monophone HMM to utterance HMM
            end for
        end for
        Finalize utterance HMM with accepting state
    end for
end procedure

procedure Align Feature Vectors to Utterance HMMs
    for each utterance HMM in utterance HMMs do
        N= number of states (minus initial and final) in utterance HMM
        M= number of feature vectors from utterance audio
        nVecs = M/N = number of feature vectors assigned to each state
        Traversing feature vectors left-to-right, assign nVecs vectors to each state
    end for
end procedure
```

Figure 2-7: Flat Start Alignment: First HMMs are compiled for every utterance in the training set, and then the audio feature vectors are aligned to those HMM states from left to right. Flat-Start alignment allows us to make alignments for an initial parameter estimation.

In this way, Flat Start training allows us to make an initial estimate for every parameter of every monophone GMM-HMM in our acoustic model. This is a very, very crude first estimate, and its assumptions are absurd. This alignment assumes that each phoneme in a word is exactly the same length as every other phoneme in that word, and actually in the whole utterance. Human speech doesn't work like this, and as such it seems like there's no way that flat start training could work, but it does. One assumption is the saving grace of Flat start training. The assumption is that the linear order of phonemes in an utterance is left-to-right. With just that one assumption built into the typology of our HMMs, we can use the Baum-Welch algorithm to make increasing better alignments to the speech data. It works because Baum-Welch training is guaranteed to improve model-data fit, no matter how bad the original fit is.

### Baum-Welch re-estimation

Now that we have an initial estimate of all the parameters in our monophone GMM-HMMs, we can use both the data and the model to make better and better alignments. The basic idea of Baum-Welch is that we can use the the current model parameters of any given utterance HMM to generate the most likely alignment of that HMM to its corresponding audio clip. Then we take that alignment as the ground-truth alignment, and update the model parameters accordingly.

By iterating over (1) parameter updating and then (2) alignment generation, we reach parameters that get closer and closer to true estimates. It is key to remember that all the utterances are updated in unison, and as such, the sounds and speech rates and other peculiarities of individual audio utterances become averaged out. Given that GMM-HMMs are generative models, we are maximizing the likelihood of the data given the model. Our implementation of the EM algorithm (i.e. Baum-Welch training) is a special case of Maximum Likelihood Estimation (MLE), where we don't have full information about the relationship between our model and the data. If we had the golden truth for alignments between feature vectors and HMM states, then we would just update the parameters once, and there would be no re-alignment afterwards. However, since we don't have the alignments, we need to guess at them, and our guesses get better over time.

## TRIPHONE TRAINING

Monophone systems work well, but they are too simple to model natural speech in its full complexity. In order to increase the complexity of monophone HMMs, there are a few routes we can take.

1. add more states to each HMM

2. add more GMMs to each state

3. create more HMMs

Monophones can become fairly powerful models if we keep adding Gaussian components to each state. We can also add more states to each HMM, and in this way capture more nuanced time information and co-articulation effects. However, no matter how many extra states we add and no matter how many extra Gaussians we throw into a state, at the end of the day, *monophone models will average together every instance of a phoneme*. This is bad because phonemes vary acoustically depending on their context, and we can predict those variations. If we can predict variation, we should model it.

Collapsing every realization of a phoneme into one HMM results in fuzziness in the model. For example, the vowel `[ah]` will display very different formant patterns if is pronounced between nasals `[n+ah+n]` ('non') versus when it is found between two stops `[t+ah+t]` ('tot'). These co-articulation effects are predictable given the context, and as such, we can reliably model them if we give our model enough freedom.

Ideally, we would like to model every phoneme in every phonetic context with its own HMM. In that way, we would be sure that all predictable variation would be built into the acoustic model. If we have $N$ number of phonemes in our language, instead of defining a mere $N$ monophones, we would ideally train $N * N * N$ models, one model for every possible context, to account for both left and right co-articulation effects. However, training $N^3$ models is much more difficult that training $N$ models, because the probability of finding $N^3$ contexts in any speech corpus is 0% . No language will allow for $N^3$ combinations of phonemes. Furthermore, given a finite data set, the more parameters we define the less data is available for each state.

## Phonetic Decision Tree

The phonetic decision tree allows us to model more context than a mere $N$ models, but fewer parameters than $N^3$. Introduced by Steve Young in 1994, the phonetic decision tree automatically defines model complexity by appropriately assigning and increasing model parameters as the dataset increases in size.[169] This efficiently avoids over-fitting on small datasets and increases model power to better exploit big datasets. Phonetic decision trees also allow for modeling of phonetic contexts which were never observed in the training data.

With monophone HMMs, the identity of the phoneme itself defines the model. To double the model complexity, we could split all training examples for each monophone into two categories: (a) the phoneme occurs in middle of word or (b) the phoneme occurs at a word edge. This division between word-internal phoneme and word-edge phonemes will account for a good deal of predictable variation. The `[l]` in 'lamp' and the `[l]` in 'complex' would be assigned to different monophones.

Now, we can stop here, (doubling complexity) or we can try to model more predictable information about each phoneme. Triphone decision trees work by extending the tree downwards from monophones, by asking questions about the left and right context. Out of all the possible predictors we could add to the model, left and right context accounts for a good deal of variation, regardless of speaker, noise channel, or anything else. The resulting models are called 'triphones' because they relate to three phonemes (left phone, right phone and central phone). [169]

The questions of the tree encode build on the linguistic knowledge that groups of phonemes may exert similar co-articulation effects. Instead of an unsupervised clustering algorithm that decreases the $N^3$ possible triphones to $M$ physical models, the tree makes splits in its branches based on some linguistically relevant questions. For example, a question in the tree could ask if the phoneme to the right is a nasal consonant (`R=Nasal?`), where the concept of nasal consonant would have to be defined before training. This way, triphones are grouped together by both the data and prior beliefs about natural classes of phonemes.

To train the phonetic decision tree (similar to Classification and Regression Trees (CART)), a set of monophones is trained. Next, every monophone HMM is copied and re-labeled as a triphone (given left and right context). Given these new, observed triphones, Baum-Welch re-estimation is performed over all the training data. The individual triphones will have a chance to update their parameters based on their own designated subset of the feature vectors. Next, with these updated triphone models we train the decision tree. All decision trees are trained at the *state* level. That is, each state can have its own root. Figure (2-8) shows a tree trained for the second

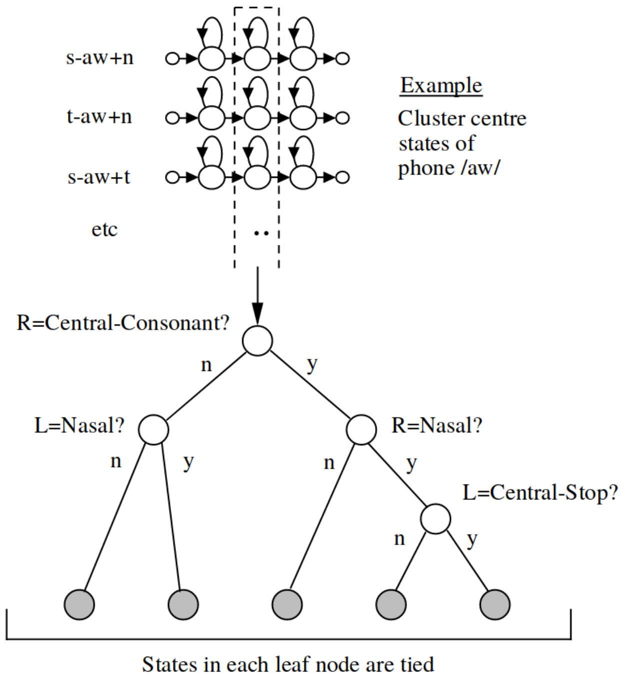


Figure 2-8: Phonetic decision tree for the second state of the phoneme [aw]. (Young 1994)

state of the phoneme [aw]. During the phonetic tree training process, the researcher assigns some hyperparameters:

1. what kinds of questions can split branches
2. how many leaves in the tree
3. how many Gaussians are allotted to the entire tree
4. how many Gaussians allotted to each leaf

## Baum-Welch re-estimation

After the phonetic decision tree has been defined, the new triphones are retrained based on the alignments from the possible triphones.

## DECODING

Decoding in HTK can be customized to a good extent, but there is a common thread in its approach that stems from Baker's work all the way back in the mid-1970's. A graph is compiled, and then the graph is searched. In HTK the graph is called a network.

### Network compilation

The compiled network contains hierarchical information about (1) grammatical word sequences, (2) word pronunciations, and (3) phoneme contexts. HTK is very flexible about which kinds of grammars can be integrated into the network, but the standard approach is to use an n-gram backoff language model. HTK also implements dynamic graph construction, so that the entire graph may be larger than available memory on the computer.

Taking an example from the HTK book itself, Figure (2-9) is a toy grammar for the two words 'bit' and 'but'. The grammar will accept any sequence of these two words, in any order. Looking at this HTK network from the level of words, we will find this kind of structure:

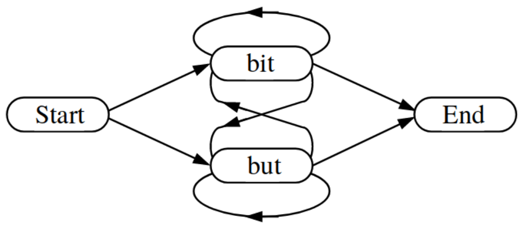


Figure 2-9: Word-Level Decoding Network in HTK. (Woodland 1993)

When we add information about pronunciations, we get the graph structure shown in Figure (2-10):

Finally, the lowest level of detail models to the contexts of the phonemes (triphones), shown in Figure (2-11). We can see that by modeling context-dependency, we've expanded our original monophone `[b]` into four possible `[b]`'s in context:

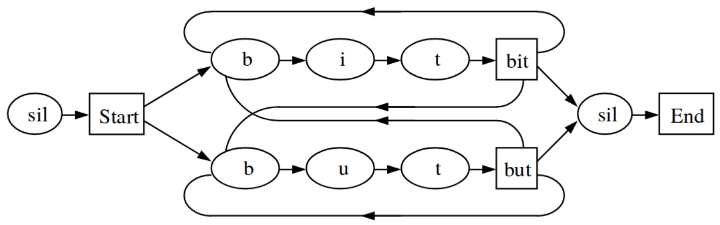


Figure 2-10: Monophone-Level Decoding Network in HTK. (Woodland 1993)

`[t+b+i],[sil+b+i],[t+b+u],[sil+b+u].`

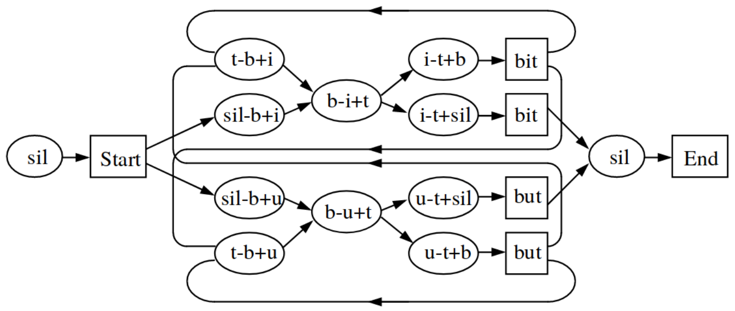


Figure 2-11: Triphone-Level Decoding Network in HTK. (Woodland 1993)

## Viterbi decoding

In HTK, the Viterbi algorithm is re-framed into what is called the 'Token Passing Model'. An example of Viterbi decoding is shown in Figure (2-12). This approach encodes information of visited paths via a Token which is passed from state to state. In Viterbi decoding, the single best possible path is returned. A 'Token' is a object which contains (a) the path sequence through which it travelled, and (b) the probability of that path given the input audio features. The HTK decoding graph represents words and phones as separate states, such that at the end of a string of phones, the Token must pass through a Word state. As a result, every Token collects a sequence of words as it travels through the graph.

Overall, HTK is a solid toolkit for speech recognition. More than anything, it became replaced because of its restrictive license.

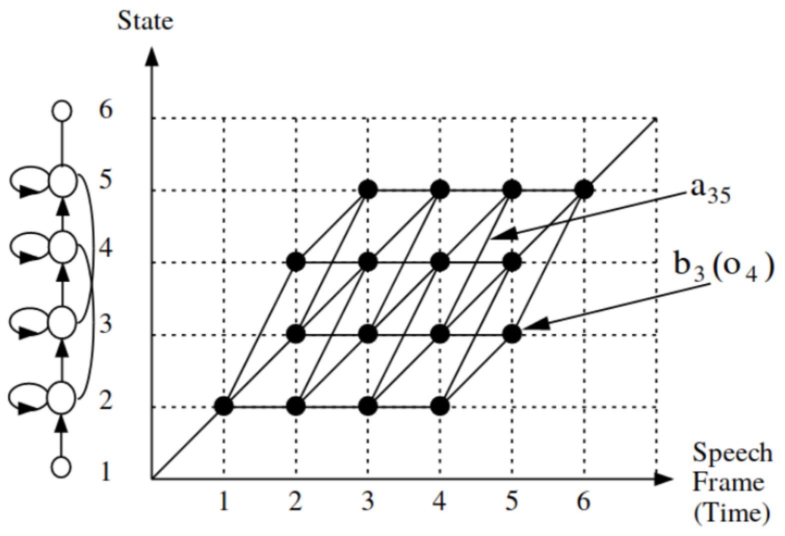


Figure 2-12: Viterbi Decoding in HTK. (Woodland 1993)

## 2.5 2011: The Modern DNN-HMM Approach: Kaldi

The modern HMM-based approach is built on Deep Neural Networks, and these models are refered to as DNN-HMMs. This is also called the 'hybrid' approach, because it is a merger of the HMMs from traditional GMM-HMM speech recognition and neural net acoustic modeling. The Hybrid approach started gaining traction in the 2000's, when neural nets were re-discovered and GPUs were used to perform backprop operations. For ASR in particular, the publishing of Hinton et al's 2012 paper "Deep Neural Networks for Acoustic Modeling in Speech Recognition", solidified the staying power of DNNs in speech recognition.

Kaldi is currently the most popular ASR toolkit for DNN-HMM research not only because it supported DNN acoustic modeling before other toolkits – Kaldi also takes a different implementation of the graph decoder, based on Weighted Finite State Transducer technology. [100, 101] Kaldi is written in C++ with a much more open license than HTK or Sphinx, making it appealing to both researchers and industry developers.

The DNN-HMM and GMM-HMM approaches are very similar in almost all respects, and the DNN is essentially just a drop-in replacement for the GMM acoustic model. Nevertheless, some adjustments need to be made for the Viterbi decoding math to work out. The standard decoding computations require the Acoustic Model to produce a *likelihood* of the audio given a phoneme, and while GMMs return *likelihoods* of the audio given the phoneme, neural nets return a *posterior* probability for each phoneme given the audio. Aside from some minor math adjustments, at decoding time GMMs and DNNs serve the same purpose. Both Acoustic Models return information about phoneme-audio relations. The rest of the decoding graph (phonetic dictionary, n-gram language model) remains the same.

Even though the DNN-HMM and GMM-HMM follow nearly identical procedures during decoding, their training is significantly different. In fact, the standard DNN training procedure relies on the GMM training to produce labeled data. Neural networks require labeled data for supervised training[*] and in our case the training example is a pair (X,Y), where X is a window of audio features and Y is a phoneme label. However, standard training data for speech recognition does not have labeled data of this kind. Typically we only have sentence-level utterance transcriptions, and as such, for DNN training to work, we need to generate the alignments of audio to phonemes.

GMM-HMM training uses Baum-Welch parameter estimation via flat-start monophone training to iterate steps of alignment and estimation in order to generate more and more accurate alignments (and more accurate GMMs). This procedure is mathematically sound, guaranteed to lead to better and better parameter estimates for the given data.

Piggy-backing off this procedure, DNN-HMM training depends on GMM-HMMs because Baum-Welch training produces alignments (i.e. labeled data). These alignments are then used as training data for DNN training with gradient descent via backprop.

Typically, to integrate time information into DNN Acoustic Model training, the input features are not merely passed in one frame at a time, frame-splicing occurs. The net still makes predictions one frame at a time, but left and right context is added on to some central frame. The net then makes a prediction as to the identity of the

[*]Notable exceptions are Sequence-to-sequence models.

phoneme which produced the central frame.

1. FEATURE EXTRACTION
   (a) sliding window feature extraction
2. GMM-HMM MONOPHONE TRAINING
   (a) Flat-start
   (b) Baum-Welch re-estimation
3. GMM-HMM TRIPHONE TRAINING
   (a) Phonetic decision tree
   (b) Baum-Welch re-estimation
4. DNN TRAINING
   (a) Frame-splicing
   (b) Gradient descent via backprop
5. DECODING
   (a) WFST Graph compilation
   (b) Viterbi decoding

## DNN TRAINING

Recall that the phonetic decision tree from the GMM-HMM model was used to define which triphone states would be merged together, and as such, the number of leaves in the tree equals the number of individual HMM models to be trained. The identity of each GMM is therefore the label for the training data, and when we train a DNN in Kaldi, we are predicting these same labels. We can view the DNN-Hybrid approach as a kind of model transfer, where we first create a generative classifier (GMMs defined by decision tree) and we train a DNN to perform the same task. This is shown in

Figure (2-13).[†] The number of nodes on the output layer of the DNN, therefore, equals the number of leaves in the previously trained decision tree.

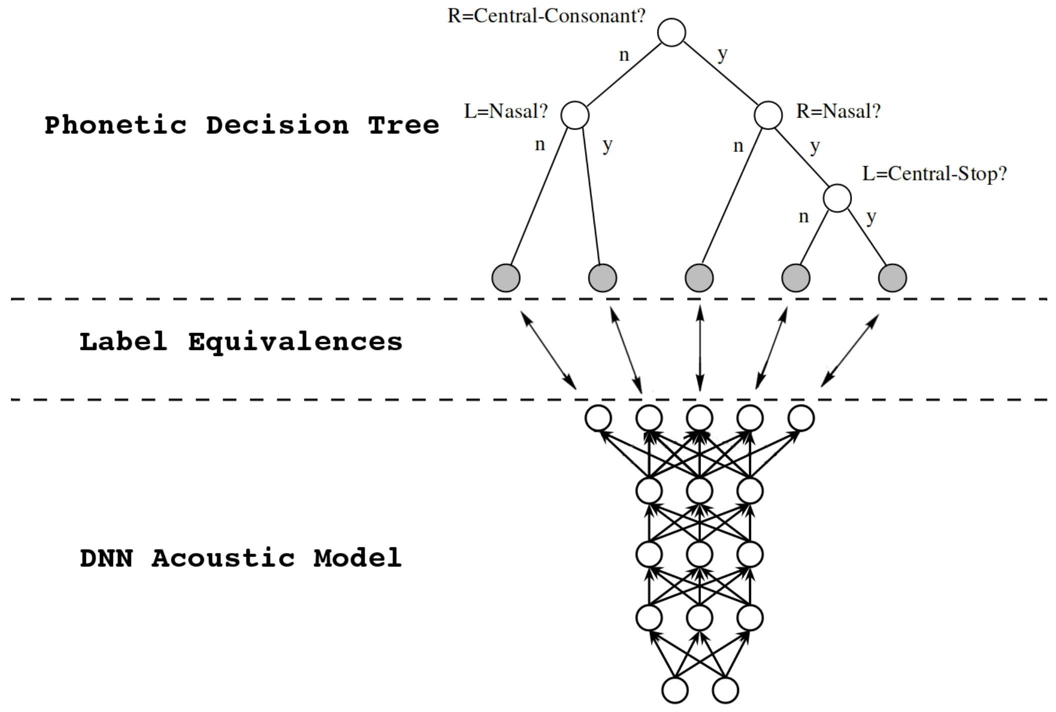


Figure 2-13: Label Equivalencies in GMMs vs. DNNs

Even though the labels come directly from the GMM model, for the DNN, each training data instance (audio frame) will contain additional contextual information. Given the alignments from the triphone GMM-HMM system, first we splice features in order to create training examples. This step should result in the same number of `(label,data)` pairs as were provided by the GMM system. It is common to splice a large window of frames (up to 10 frames on each side), and then reduce the dimensionality of the data with some transform (typically Linear Discriminant Analysis (LDA)).

## DECODING GRAPH

Kaldi's approach to graph compilation sets it apart from other toolkits. Building on work by Mohri [100, 101], Kaldi implements every knowledge source in the ASR

[†]The original tree figure comes from Young (1994) and the net comes from Heigold (2013).

pipeline (aside from the Acoustic Model) as a Weighted Finite State Transducer (WFST). WFSTs are an elegant choice for speech decoding, because they allow for not only hierarchical combination of knowledge (sentences, words, phonemes), but also the combination of weights from each level of the hierarchy. That is, we can just as easily assign probabilities to sequences of words as to sequences of phonemes. All these sources of probabilities are encoded as weights on a WFST, and during compilation, all that information and bias is saved and used.

Following Mohri, Kaldi creates a single master decoding graph from four (4) WFSTs. The final graph is called HCLG, and it is the composition of subgraphs H, C, L, and G.

H: pdfs $\rightarrow$ triphones

C: triphones $\rightarrow$ monophones

L: words $\rightarrow$ monophones

G: words $\rightarrow$ words

Starting from the highest level and working down, the WFST 'G' represents the grammar defined by the language model (c.f. Figure (2-14)) It accepts valid sequences of words, and returns both a weight (pseudo-probability) and that same sequence of words.

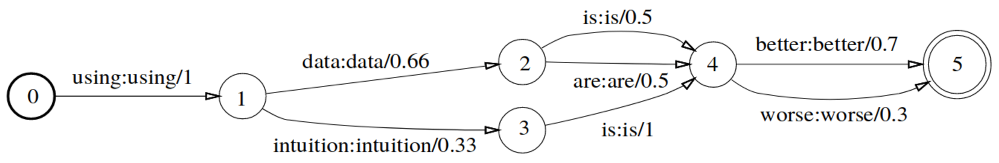


Figure 2-14: Grammar WFST. Mohri (2002)

If we use a backoff n-gram language model to define our grammar, then we will have different possibilities depending on how much of the sequence was observed in training. A decoding graph cannot recognize a word it never saw. For example, given a sequence of words (e.g. "She codes best in Python."), that sequence will be assigned a higher probability if it was seen contiguously, rather than if all the words were seen apart. If that sentence was never seen in training, but the two sentences: "She codes best in class." and "She prefers C++ to Python.", then the original sentence is grammatical, but the bigram "in Python" was never seen. As such, the backoff weights for "in" and "Python" would be combined. This is the idea behind a backoff language model, and WFSTs naturally encode this with a backoff state. In Figure (2-15), *b* is the backoff state, *w1* is "in" *w2* is "class" *w3* is "Python"

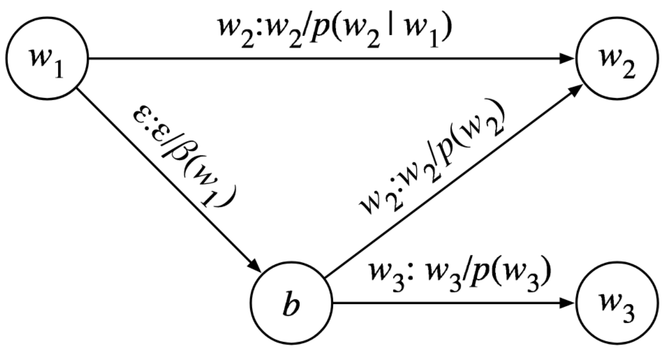


Figure 2-15: WFST Backoff Model. Mohri (2002)

To encode information about word pronunciation from the phonetic dictionary (also known as a Lexicon), Kaldi uses a WFST called L.fst (c.f. Figure (2-16)). This WFST accepts a sequence of phonemes and returns a word and a weight (along with a string of epsilons, which are ignored).

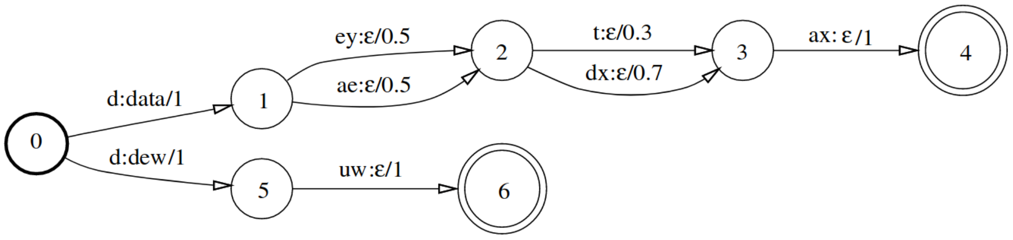


Figure 2-16: Phonetic Dictionary WFST. Mohri (2002)

The next WFST encodes information about the triphone-monophone relationship. This WFST is called `C.fst` because it contains phoneme context. It translates a sequence of phonemes into a sequence of triphones (Figure (2-17)).

This last WFST encodes all the information from the phonetic decision tree. It accepts triphones and returns state identities. You will notice that all arcs leaving a

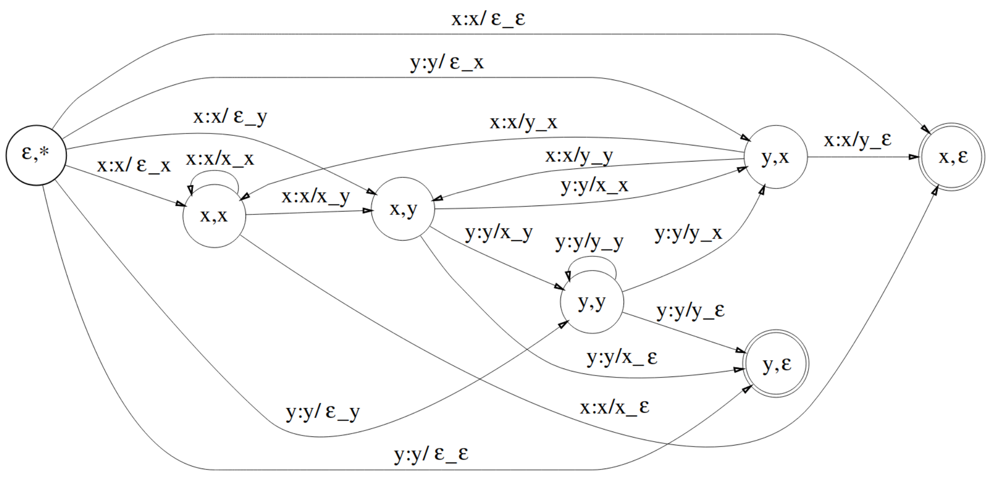


Figure 2-17: Monophone to Triphone WFST. Mohri (2002)

state have the same label. This is because Kaldi encodes all information about states via arcs. Technically, HCLG does not contain states. They are inferred from the arcs. As such, the H.fst (Figure (2-18)) accepts a triphone and returns the IDs of the states of which the triphone is comprised. If we have 3-state models, then C.fst will return 3 states for every triphone. These state IDs are linked to either individual Gaussian mixtures (for GMM-HMM) or output nodes of a neural network (for DNN Hybrid system).

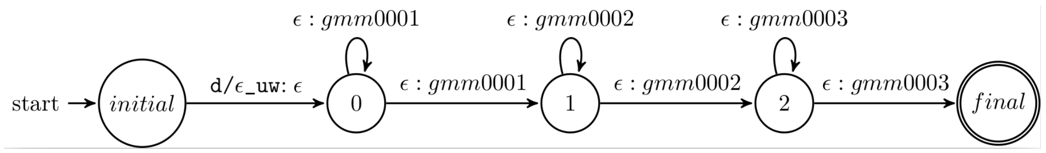


Figure 2-18: A Gaussian to Triphone State WFST.

## 2.6 2014: End-to-End ASR

The ASR world changed dramatically in 2014 with the first publications showing promising results from End-to-End (i.e. audio to text) speech recognition. [57, 64, 66, 58, 28] Previous approaches to ASR involved training separate models for acoustics, word sequences, and pronunciations. Mastering each of these separate models takes considerable time for an engineer, and is a burden to implementing and troubleshooting speech recognition in production. Furthermore, the traditional, Hybrid approach is disfavored *a priori* because each model requires its own objective function during parameter estimation, and as such joint estimation of all models is not possible. Traditional Hybrid approaches are also disfavored because they pose a strict limit on the set of recognizeable words. If a word is present in the language model, then the Hybrid model can decode it – however, words absent from the language model are Out-Of-Vocabulary (OOV) and will be impossible to recognize in speech.

End-to-end speech recognition is prefered because there is a single Loss function for parameter estimation. This Loss allows us to directly model the problem of interest: finding the best transcript for a given audio segment. However, by throwing out the HMMs from the Hybrid approach, End-to-End models do not have access to the alignment information typically used in training traditional models. As such, the End-to-End speech recognition model must find an alignment of the text and audio during training – this is not a simple task.

The first approaches to End-to-End speech recognition made use of the Connectionist Temporal Classification (CTC) objective function. [57, 64, 66, 58, 2] The Sequence-to-Sequence model with Attention approach was established soon after. [28, 6] A notable extension of the Sequence-to-Sequence model with Attention is the so-called "Listen, Attend, Spell" (LAS) model. [22] Breaking with recurrent models, Wav2Letter was introduced and built on purely convolutional layers and a modification to the CTC Loss function. [30] The RNN Transducer approach allows for on-line, streaming decoding as in CTC models, but also by-passes the CTC conditional independence assumption. [124, 9]

Over the last five years, End-to-End models have become more performant, but they typically are trained on massive in-house datasets only available to the tech giants. Nevertheless, there is current research on making these End-to-End models work for smaller datasets, and Chapter (8) of this dissertation lends itself to that literature.

## 2.7 Conclusion

The pipelines and algorithms used in Automatic Speech Recognition have changed significantly since the 1950's, and they continue to evolve. It is easy to see that these techniques build off of one another, discarding or adjusting modules in the pipeline. Older systems' approaches may be incorporated into newer pipelines, as we find in Hybrid neural net Acoustic Modeling. These neural networks rely on Gaussian-based models to generate alignments for supervised learning.

Speech recognition has evolved in order to produce more accurate systems, faster decoding, and simpler pipelines. The emergence of End-to-End systems is a prime example of research moving along lines not dictated by model accuracy alone. It has taken years for End-to-End models to reach the performance (in terms of Word Error Rate) of Hybrid DNN-HMM models. However, the simplicity of a single, unified neural network in addition its potential to make better use of larger datasets has driven research forward for End-to-End models.

# Chapter 3

# Previous Work on Model-Data Fit for GMM-HMMs

*The easiest and most straightforward way to improve noise robustness is to attempt to collect large amounts of data from a wide range of acoustic environments. Although this so-called multi-environment training works reasonably well up to a certain limit, it is obvious that the problem of noise robustness cannot be solved by simply collecting a huge training set. It is impossible to collect such a database that would cover all possible usage environments. (Viikki 1998, p. 134)*

## 3.1 Introduction

The problems of model generalization existed long before DNNs were introduced to speech recognition. The field's long history with GMM Acoustic Models contains a wealth of studies which tackle the same low-resource problems faced today. While many of the specifics in the implementations of these approaches cannot be directly ported over to DNN-based speech recognition, they present interesting conceptualizations of the problem.

This low-resource issue is a special case of *the generalization problem*, because we would like to generalize from training to test data, but it is rarely successful out-of-the-box.[47] For speech, the test data may have been recorded (a) in a different noise environment (e.g. outdoors, warehouse, in a car), (b) via a different channel (e.g. different microphones, cellphones), or (c) by different speakers (e.g. different ages, different genders). An extreme case of mis-match is when the training and test data come from (d) different languages. The terms *noise robustness* [56], *channel robustness* [125], *speaker adaptation* [161, 104, 90, 51], and *language adaptation* [134, 15] have been used to categorize these sub-literatures.

Overfitting to a domain occurs because statistical models encode not only information relevant to the task, but they also "learn" misleading noise specific to the training data. With regards to ASR, this means that speaker-dependent models will encode information about speakers, noise-dependent models will encode information about noise, and channel-dependent models encode information about channels. For a speech recognizer to do its job, it must ignore all irrelevant information about speakers, noise, and channels. A speech recognizer should only care about speech (i.e. phonemes), and be able to recognize that speech in any context.

The following GMM-based approaches attack the problem by identifying three important resources: (1) in-domain data, (2) in-domain model, and (3) out-of-domain data (either small amount or not available). Given these three resources, the following general approaches have been proposed:

1. make the two datasets more similar
2. adapt the old model to the new data
3. adapt the new data to the old model

Of the three approaches listed above, the first approach is called *feature normalization.*[154, 37] Feature normalization attempts to perform feature extraction so that features from different domains will look similar for phonetically similar units. For example, an `[a]` will look like an `[a]` whether is was spoken by a man or a woman. You can think of this approach as a kind of data cleaning, where we want to filter out all noise but save all linguistic information. These feature normalization techniques make no reference to acoustic models, and as such can be applied before any model is trained.

The second approach consists of so-called *model-based adaptation*.[129, 138, 144, 48, 90, 51, 151] Model-based adaptations aim to take old model parameters and update (or transform) them in a way that improves task performance on a new data set. This assumes we have access to some new data, and usually that the data set is small.[*] Whether we are estimating a transform of the model or we are estimating the model itself, we can think of model adaptation as falling into two major categories. Adaptation using (a) only adaptation data or (b) adaptation data *plus* prior knowledge. The use of these two sources is what differentiates the popular Maximum Likelihood Estimation (MLE) [90] from the Maximum A Posteriori (MAP) [51] estimation approaches. In the following, $\theta$ refers to model parameters, and $X$ refers to a training data set.

$$\theta_{MLE} = \arg\max_{\theta} p(X|\theta)$$
$$\theta_{MAP} = \arg\max_{\theta} p(X|\theta) \cdot p(\theta)$$

Regardless of the source of knowledge, we can think all kinds of model-based transforms as taking old model parameters $\theta_{old}$, and projecting them into a different space $\theta_{new}$.[138] Ideally, we want this projection to:

1. *lose* information specific to the original data set
2. *capture* new information about the new data set
3. *retain* general information common to both data sets

We assume the two data sets (original + adaptation) have information in common, and that the original model captured some of that common information. Otherwise there would be no point in using the original model at all.

[*]If the new data set is big enough, it would make more sense to train a new model from scratch.

A third approach to improving data-model fit is *feature adaptation.* Feature Adaptation is a special case where we use the model to find a good transformation of the data to the model. [63, 39]

## 3.2 Feature Normalization

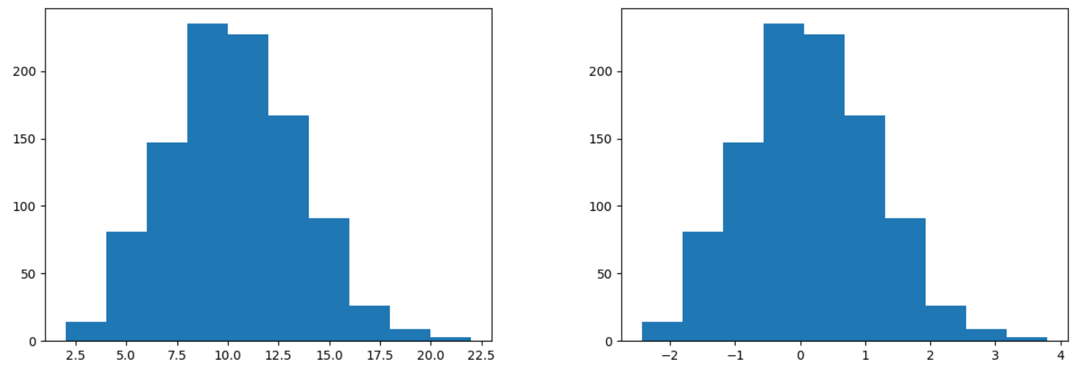
### 3.2.1 Cepstral Mean and Variance Normalization

Cepstral Mean and Variance Normalization (CMVN) is an attempt to extract features which are robust to environmental influences.[154] During CMVN an Acoustic Model is never referenced, only the features themselves. CMVN it is an attempt to perform speech recognition better in unseen environments and unseen speakers, and it is not trained from a dataset *per se*, but applied locally per utterance. CMVN is an application of a technique common in statistics: *z-scoring.* To build an intuition for what kind of transformation CMVN performs on the speech data, below is an example of CMVN (z-scoring) performed on a small random dataset (drawn from a Poisson probability density function). The first line of Python code below generates the data, and the second line performs the transform.

```
import numpy as np


org_data = np.random.poisson(lam=10.0,size=1000)
new_data = ( ( org_data - np.average(org_data) ) / np.std(org_data) )
```

Figure (3-1) displays the data as histograms before (3-1a) and after (3-1b) the transformation, we can visualize what CMVN accomplishes. Firstly, we can see that the spatial relations among all data points remains the same, this is a nice feature if we want to distinguish different classes of data (as in phoneme classification). While the shape of the data hasn't changed, if you look at the `x-axis`, you will observe that the data has shifted to the left. Once the mean was subtracted from each datapoint, the data becomes centered at `[0]`.



(a) Original Data (b) Normalized Data

Figure 3-1: Z-score Normalization

Now that we've seen the general case of z-scoring, it is easy to see how we can use CMVN to combat lower recognition performance on data from (1) different noise conditions and (2) different recording channels. These conditions typically carry some stable, additive noise which can be effectively separated from the signal by removing the mean and dividing by the standard deviation. In speech recognition, however, it has been found that instead of taking the global mean and standard deviation for some audio file, it is best to use immediate context (via a sliding window) to estimate the mean ($\mu$) and standard deviation ($\sigma$).

This is good intuitively, because we may have noise which is stable for some period of time, but not the entire time of the utterance. For example, someone turns on a fan for just a few minutes during recording. If we try to subtract the fan noise from the entire utterance, we will remove signal where the fan noise doesn't exist. For this reason, the sliding window works well.

More concretely, since we are working with audio data (typically PLPs or MFCCs), what we are doing here is removing the average amplitude (over a window) from each frequency bin in our feature vector. In the below equations $x_t$ represents the feature vector at time $t$; $D$ allows for a lag in processing time for on-line processing (i.e. $D == delay$); $i$ indexes the $ith$ dimesion of the feature vector. The original equations and explanations can be found in ([154], p. 136).

$$\hat{x}_{t-D}(i) = \frac{x_{t-D}(i) - \mu_t(i)}{\sigma_t(i)}$$

$$\mu_t(i) = \frac{1}{N}\sum_{j=1}^{N} x_j(i)$$

$$\sigma_t(i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\Big(x_j(i) - \mu_t(i)\Big)^2}$$

### 3.2.2 Vocal Tract Length Normalization

Vocal Tract Length Normalization (VTLN) is a feature normalization approach inspired by the anatomy of the human body. [29, 44, 80, 89] The basic idea is that human speech passes through an individual human's vocal tract, and the vocal tract imparts itself into the acoustics of the final signal. This is precisely why we humans can easily recognize our mother's voice over the phone without any introduction. This signal, however, is distracting for a statistical acoustic model. The human vocal tract adds noise to the acoustic signal, distorting the locations of formant frequencies in phonetic sounds. VTLN attempts to estimate the vocal tract of the speaker, and use it to warp the signal into a more speaker-independent version of itself.

## 3.3 Feature-based Adaptation

### 3.3.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) helps accomplish two concurrent tasks: (1) compress data, and (2) enhance data separability for classification.[63] It's a technique that helps us *discriminate* the data via a *linear* transform (hence the name LDA). Whether we're using GMMs or DNNs in speech recognition, at the end of the day we have labels for our data which represent classes in some feature space. It is probably the case that our classes are not easily separable in their original space (one reason why ASR is a hard task), but LDA will help us take those features and project them into a lower-dimensional space where we can distinguish phones better.

LDA is often compared to Principal Component Analysis (PCA), given that both techniques seek to find a linear projection of data into lower-dimensional space, where the interesting characteristics of the data are enhanced, and uninformative aspects of the data are lost.[18, 17] The main difference between these two techniques is that LDA makes use of labels whereas PCA does not. PCA merely extracts the features dimensions where variability is the highest. LDA, on the other hand, identifies dimensions where inter-class variability is high and intra-class variability is low.

LDA makes reference to two pieces of information: (1) feature vectors and (2) their labels. It may seem like we are only accessing data, and as such LDA should not be considered adaptation. However, we should remember that labels for the data are assigned during training, and they depend on the architecture of our model. Standard ASR training begins with only utterances and their text transcriptions. In order to acquire frame-level labels for our phonemes, we need to generate the labels ourselves. Monophone training gives us some intermediate labels, which are then used to bootstrap a triphone system. The decision tree of the triphone system is what determines the number and kind of labels we eventually end up with. As such, our labels are a direct consequence of our training procedure and model architecture, and therefore LDA can be classified as *feature adaptation* because we implicitly make reference to our original model.

Heteroscedastic Discriminant Analysis (HDA), addresses the one shortcoming of standard LDA.[86] Standard LDA assumes that all classes have equal variance, but in reality this is rarely the case. HDA makes provisions to account for unequal variance between classes (aka. heteroscedasticity).

## 3.4 Model-based Adaptation

### 3.4.1 Maximum Likelihood Linear Regression

Maximum Likelihood Linear Regression (MLLR) trains a transformation of model parameters such that the adapted model maximizes the likelihood of some new data.[90] Given a previously trained acoustic model, MLLR takes all the parameters from all the Gaussians and finds a transformation of them which is more appropriate for some new data. In the formula below, $\eta$ are the parameters being learned for the transformation (i.e. adaptation) matrix. Our existing Gaussian model parameters are $\theta$. We are not learning new model parameters, but instead the best transformation of those parameters for our new data $X$.

$$\eta_{MLLR} = \arg\max_{\eta} p(X|\theta, \eta)$$

MLLR is especially useful when little adaptation data is available, because it does not require data from each phoneme (i.e. triphone) to make an update. If a GMM Acoustic Model has been trained for some large dataset, and we want to use MLLR to adapt that model to some new, smaller data set, the chances are that the smaller dataset has fewer seen triphones than in the original dataset. More concretely, since triphones are modeled as a collection of GMM components, let's say that we have two Gaussian components $a$ and $b$ with two mean vectors $\mu_a$ and $\mu_b$. In MLLR, we don't need any new data $x_i$ to be assigned to component $a$ in order to make an update to $\mu_a$. Rather, using some assumption about how components are related to each other, MLLR will train an update for a group of components at once. As long as at least one component in that group has new data, the transformation for all components will be trained. If the components $a$ and $b$ belong to the same cluster, then $a$ will be updated according to the transformation trained on $b$. Intuitively, we can think of this as an update rule stating: *If you don't have new data to update a certain part of the model, look for a similar part of the model, and use that update information.* The most extreme case of MLLR is when *all* components share a single transformation matrix. While this may seem odd, for certain noise or channel conditions a single transformation may actually be effective for adaptation to a new environment. The noise inside a moving car will be fairly stable throughout a phone-call, so training one global transformation to a new environment is justified.

With more adaptation data, more informed *regression classes* for clusters of components can be learned via a decision tree. The intuition behind this decision tree is that similar sounds will be transformed similarly. For example, a single transformation may be trained for vowels, another transformation may be trained for consonants, and another for noise/silence. Another advantage of MLLR is that instead of directly adapting model parameters, we are merely estimating a transformation matrix. This means we have fewer parameters to estimate and store, and as such, we can save to disk multiple transformations (speaker, noise, channel) for a given model. A company wishing to customize an Acoustic Model to every user will have a much easier time using MLLR compared to completely retraining model parameters, because every user will only need a new adaptation matrix, as opposed to an entirely new model.

MLLR uses Maximum Likelihood Estimation to train a linear regression, where the linear regression is formulated as an affine transform. In the below equation, $A$ is a $d \times d$ matrix, and $d$ is the number of dimensions in a single Gaussian component of a GMM of a state, and $\mu$ is the mean vector of that component, such that $\mu \in \mathcal{R}^d$.

$$\hat{\mu} = A\mu + b$$

To make the math easier, we encode the weights and biases of the transformation into a single matrix:

$$\hat{\mu} = \mathbf{W}\xi$$

In this formulation, $\xi$ is an extended feature vector:

$$\xi = [1\mu^\top]^\top = [1, \mu_1, \mu_2 \ldots \mu_d]^\top$$

where $d$ is the dimensionality of the mean vector of a single Gaussian component in our model (and also of course, our original $d$-dimensional feature vectors). To return a $d$ dimensional mean vector for our Gaussians, $\mathbf{W}$ is a $d \times (d+1)$ matrix,

$$\mathbf{W} = [\mathbf{bA}]$$

Instead of just training one global transformation $\mathbf{W}$, you can train a set of transformations $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \ldots \mathbf{W}_n\}$ for $n$ subsets of the acoustic model. To accomplish this, typically a decision tree is used to find states which are likely to be transformed in a similar way. The following is the transformation matrix for Full-Covariance MLLR, with a padded Gaussian mean vector:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & \ldots & w_{1,(d+1)} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \ldots & w_{d,(d+1)} \end{bmatrix} \mu = \begin{bmatrix} 1 \\ \mu_1 \\ \vdots \\ \mu_d \end{bmatrix}$$

The parameters of the transforms are updated in order to maximize the likelihood of the new data given the transformed GMMs. If we use GMMs with diagonal covariance matrices there is a closed form solution to MLE, otherwise an iterative procedure is required.

### 3.4.2 Maximum A Posteriori

Maximum A Posteriori (MAP) parameter estimation allows us to naturally use prior knowledge in adapting existing model parameters to new data. [51] When we maximize the *a posteriori* distribution for a set of parameters $\theta$, given some new data $X$, using Bayes Theorem we can explicitly use prior knowledge in the form of a prior $p(\theta)$. This prior on $\theta$ can take many shapes and come from many different sources.

$$\begin{aligned}\theta_{MAP} &= \arg\max_{\theta} p(\theta|X) \\ &= \arg\max_{\theta} p(X|\theta) \cdot p(\theta)\end{aligned}$$

Typically, in GMM Acoustic Model adaptation, MAP adaptation refers to the case where $p(\theta)$ comes from some original model $\theta_{org}$ trained on some original data set $X_{org}$. In speaker adaptation, for example, we may have some previous, speaker-independent Acoustic Model $\theta_{org}$, and we wish to adapt that Acoustic Model to a new speaker, via some new data $X_{new}$. Typically, the amount of data in $X_{new}$ is small, and training a new model $\theta_{new}$ from scratch via standard MLE with $X_{new}$ will lead to bad performance. The main intuition behind MAP adaptation is that we use (a) our background knowledge $\theta_{org}$ in addition to (b) our new data $X_{new}$ to create a new model $\theta_{new}$ which exploits information from both sources via Bayes Theorem.

In standard Maximum Likelihood Estimation (MLE) of GMM model parameters, we estimate the means and (diagonal) covariances of our Gaussian probability density functions as such:

$$\begin{aligned}\mu_{\mathbf{MLE}} &= \frac{\sum_{n=1}^{N} \gamma_n \cdot \mathbf{x_n}}{\sum_{n=1}^{N} \gamma_n} \\ \sigma^{\mathbf{2}}_{\mathbf{MLE}} &= \frac{\sum_{n=1}^{N} \gamma_n \Big(\mathbf{x_n} - \mu\Big)^2}{\sum_{n=1}^{N} \gamma_n}\end{aligned}$$

Where $N$ is the number of training examples, $\mathbf{x_n}$ represents the feature vector $\mathbf{frame_n}$, and $\gamma_n$ is the probability of $\mathbf{frame_n}$ belonging to the current component of the current state. In MAP adaptation, we take a similar approach to standard MLE in estimating the means of a new, updated GMM. Crucially, when we have new adaptation data, we use the old GMM model parameters as a prior, and we use a hyperparameter $\tau$ to control the influence of the prior on the new model parameters.

A desireable trait of MAP adaptation is that as the amount of adaptation data increases, the influence of the prior fades away. This is accomplished by using a hyperparameter weight $\tau$.

$$\mu_{\mathbf{MAP}} = \frac{\tau}{\tau + N_{new}} \cdot \mu_{\mathbf{org}} + \frac{N_{new}}{\tau + N_{new}} \cdot \mu_{\mathbf{new}}$$

As the amount of new data increases, the influence of the original model ($\mu_{\mathbf{org}}$) goes to zero.

$$\lim_{N_{new} \to \infty} \frac{\tau}{\tau + N_{new}} \cdot \mu_{\mathbf{org}} = 0$$

Likewise, as the amount of new data increases, the influence of the new parameter ($\mu_{\mathbf{new}}$) approaches 100%.

$$\lim_{N_{new} \to \infty} \frac{N_{new}}{\tau + N_{new}} \cdot \mu_{\mathbf{new}} = \mu_{\mathbf{new}}$$

Putting it in another way, a very desirable trait of MAP estimation is that as the amount of new data tends to infinity, MAP tends to MLE. Intuitively, this translates into: *The more data we feed the model, the less we rely on prior biases.*

$$\lim_{n \to \infty} \frac{\tau \cdot \mu_{org} + \sum_{n=1}^{N} \gamma_n \cdot \mathbf{x_n}}{\tau + \sum_{n=1}^{N} \gamma_n} = \frac{\sum_{n=1}^{N} \gamma_n \cdot \mathbf{x_n}}{\sum_{n=1}^{N} \gamma_n}$$

The main problem with MAP is that adaptation only happens locally. That is, if data for a certain phonetic label is not observed in the new data, the model of that phonetic label will be just copied from the old parameters. Therefore, usually MAP is good if you have a lot of data. Given that it's easier to get data from a new noise environment, MAP has seen more success in noise/channel adaptation compared to speaker adaptation.

### 3.4.3 i-vectors

I-vectors are the last traditional approach to solving the small-data problem I will cover in this chapter. I-vectors help deal with new acoustic data (i.e. new speakers, new background noise) that was never seen in training.

I-vectors are MAP-adjusted means from a speaker-independent GMM (i.e. a Universal Background Model (UBM)) which has been trained on a large dataset of speakers. I-vectors are appended to standard audio features (e.g. MFCCs) during decoding. [133, 126, 116, 136] Given a UBM and utterances from a new speaker, the UBM means are first adapted to those utterances via MAP estimation. Then, the means of the MAP-adapted GMM (now speaker-specific) may be extracted and used as a compact representation of how that speaker diverges from the norm (UBM). As such, an i-vector is a compact, fixed-dimensional representation of an audio file. I-vectors were initially used for speaker identification, and were only later integrated into speech recognition. I-vectors are very convenient for speaker identification because a single, averaged i-vector can be stored per speaker on disk.

Intuitively, i-vectors are useful in decoding because they allow the recognizer to adjust for speaker-specific characteristics. For example, a 6-year-old child will have certain vocal characteristics (i.e. Hz range, formant relations) which differ from adults. We as humans can pretty reliably say when a voice is a child's or adult's, even in another language. Knowing this information allows us to adjust our hearing to more reliably identify phonemes spoken by children (and adults, elderly, other accents). When we provide i-vectors at each time-step to a DNN classifier, we are effectively providing this speaker-specific information so that the system can learn regularities in different kinds of voices, and adjust it's phoneme classification depending on them.

## 3.5 Conclusion

In understanding speech, there is a wealth of information we can exploit as humans. We know if a recording was done over the phone or in a soundbooth, in a car or on the factory floor. We can also make a good guess as to the gender, sex, age, physical health, and even socio-economic status of the speaker. We can also identify what kind of accent the speaker has, and if they are happy or upset. All of these things, and more, we can identify from an audio recording, but almost none of these are directly useful when it comes to transcribing the speech of that speaker. As such, audio recordings contain a lot of information we can safely ignore if our sole task is to recognize which words were spoken. This chapter discussed the most common approaches to model relevant information and ignore irrelevant information in audio data. This becomes acutely relevant when there is a mis-match between training and testing data.

# Chapter 4

# An Overview of Multi-Task Learning in Speech Recognition

## 4.1 Roadmap

In this overview of Multi-Task Learning in Automatic Speech Recognition (ASR), we're going to cover a lot of ground quickly. First, we're going to define Multi-Task Learning and walk through a very simple example from image recognition[*]. Then, once we have a base as to what Multi-Task Learning is and what constitutes a task, we will move into a survey of the speech recognition literature.[†]

The literature survey focuses on acoustic modeling in particular. Speech Recognition has a long history, but this overview is limited in scope to the Hybrid (i.e. DNN-HMM) and End-to-End approaches. Both approaches involve training Deep Neural Networks, and we will focus on how Multi-Task Learning has been used to train them. We will divide up the literature along monolingual and multilingual models, and then finally we will touch on Multi-Task Learning in other speech technologies such as Speech Synthesis and Speaker Verification.

"Multi-Task Learning" denotes more than a model which can perform multiple tasks at inference time. Multi-Task Learning can be useful even when there is just

[*]Yes, there will be pictures of dogs!

[†]For a more complete overview of Multi-Task Learning itself, c.f. [128]

*one* target task of interest. Especially with regards to small datasets, a Multi-Task model can beat out a model which was trained to optimize the performance of just one task. Moreover – as demonstrated in Chapters 5-7 – Multi-Task Learning can be used even when there is one task explicitly labeled in the training data.

## 4.2 Introduction to Multi-Task Learning

### 4.2.1 Definition

Before we can define *Multi-Task Learning*, we first need to define what counts as a *task*. Some researchers may define a task as a set of data and corresponding target labels (i.e. a task is merely $(X, Y)$). Other definitions may focus on the statistical function that performs the mapping of data to targets (i.e. a task is the function $f : X \rightarrow Y$). In order to be precise, in this overview a task is defined as the combination of data, targets, and mapping function.

A *task* is the combination of:

1. Data: $X$, a sample of data from a certain domain
2. Targets: $Y$, a sample of targets from a certain domain
3. Mapping Function: $f : X \rightarrow Y$, a function which maps data to targets

The *targets* can be distinct label categories represented by one-hot vectors (e.g. classification labels), or they can be $N$-dimensional continuous vectors (e.g. a target for regression).[‡]

Given this definition of *task*, in this overview we define Multi-Task Learning as a training procedure which updates model parameters such that the parameters optimize

[‡]In reality these two kinds of targets are the same with regards to training neural networks via backpropagation. The targets for classification are just a special case of regression targets, where the values in the vector are 1.0 or 0.0.

performance on multiple tasks in parallel.[§] At its core, Multi-Task Learning is an approach to parameter estimation for statistical models [21, 19]. Even though we use multiple tasks during training, we will produce only one model. A subset of the model's parameters will be task-specific, and another subset will be shared among all tasks. Shared model parameters are updated according to the error signals of all tasks, whereas task-specific parameters are updated according to the error signal of only one task. It is important to note that a Multi-Task model will have both *task-dependent* and *task-independent* parameters. The main intuition as to why the Multi-Task approach works is the following: if tasks are related, they will rely on a common underlying representation of the data. As such, learning related tasks together will bias the shared parameters to encode robust, task-independent representations of the data.

Given this definition of *task* and this definition of *Multi-Task Learning*, we can start to think about the different ways in which a Multi-Task model can be trained. Probably the most common Multi-Task use-case is the classification of a single dataset ($X$) as multiple sets of target labels ($Y_1, Y_2 \dots Y_N$). This model will perform mappings from ($X$) into each of the label spaces separately. Another approach is the classification of multiple datasets sampled from various domains ($X_1, X_2 \dots X_N$) as their own, dataset-specific targets ($Y_1, Y_2 \dots Y_N$). Less commonly, it is possible to classify multiple datasets using one super-set of labels. These different approaches are represented with regards to vanilla feed-forward neural networks in Figure (4-1).

With regards to neural approaches (c.f. Figure (4-1)), Multi-Task models are comprised of three component parts: (1) `shared hidden layers` which serve as a task-independent feature extractor; (2) `task-specific output layers` which serve as task-dependent classifiers or regressors; (3) `task-specific input layers` which serve as feature transformations from domain-specific to domain-general representations. Neural Multi-Task models will always have some hidden layers shared among

[§]An exception to this is Multi-Task Adversarial Learning, in which performance on an auxiliary task is encouraged to be as poor as possible. In domain adaptation, an example of this may be forcing a neural net to be blind to the difference between domains. The Adverserial auxiliary task would be classification of `domain-type`, and the weights would be updated in a way to increase error as much as possible.

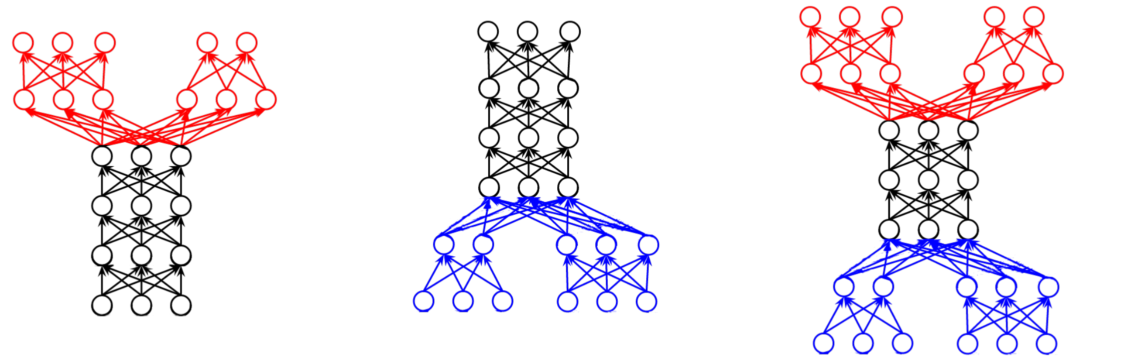

(a) Multi-Label (b) Multi-Data (c) Multi-Label + Multi-Data

Figure 4-1: Possible Neural Multi-Task Architectures. Black layers are task-independent layers, blue layers are task-dependent input layers, and red layers are task-dependent output layers. These figures are modified versions of a figure from [71].

tasks. This view of a Multi-Task neural network highlights the intuition behind the shared hidden layers - to encode robust representations of the input data.

With regards to domains in which we have very limited data (i.e. low-resource environments), Multi-Task parameter estimation promises gains in performance which do not require us to collect more in-domain data, as long as we can create new tasks. In the common scenario where an engineer has access to only a small dataset, the best way she could improve performance would be by collecting more data. However, data collection takes time and money. This is the promise of Multi-Task Learning in low-resource domains: if the engineer can create new tasks, then she does not need to collect more data.

### 4.2.2 An Example

This section gives an example of a Multi-Task image recognition framework, where we start with a single task, create a second task, and train a model to perform both tasks. Starting with a single task, suppose we have the access to the following:

1. Data: $X_1$, a collection of photographs of dogs from one camera

2. Targets: $Y_1$, a finite set of `dog_breed` labels (e.g. terrier, collie, rottweiler)

3. Mapping Function: $f_1 : X_1 \rightarrow Y_1$, a vanilla feedforward neural network which returns a set of probabilities over labels of `dog_breed` to a given photograph

In order to create a new task, we either need to collect some data ($X_2$) from a new domain, create new targets ($Y_2$), or define a new mapping function ($f_2 : X_1 \rightarrow Y_1$). Furthermore, we would like to create a *related task*, with the hopes of improving performance on the original task. There's several ways we can go about making a new task. We could use the same set of labels (`dog_breed`), but a collect new set of pictures from a different camera. We could try classifying each photograph according to the size of the dog, which would mean we created new labels for our existing data. In addition to our vanilla feed-forward network, we could use a convolutional neural network as a mapping function and share some of the hidden layers between the two networks.

Assuming we don't want to collect more data and we don't want to add a new mapping function, the easiest way to create a new task is to create a new set of target labels. Since we only had a single set of labels available (i.e. `dog_breed`), we can manually add a new label to each photo (i.e. `dog_size`) by referencing an encyclopedia of dogs.[¶] So, we started with a single dataset of photos of dogs ($X_1$) and a single set of classification labels ($Y_1$) for the dog's breed, and now we've added a new set of labels ($Y_2$) for a classification task of the dog's size. A few training examples from our training set ($X_1, Y_1, Y_2$) may look like what we find in Figure (4-2).

Given this data and our two sets of labels, we can train a Multi-Task neural network to perform classification of both label sets with the vanilla feed-forward architecture shown in Figure (4-3). This model now has two task-specific output layers and two task-specific penultimate layers. The input layer and following three hidden layers are shared between both tasks. The shared parameters will be updated via the combined error signal of both tasks.

[¶]This is an example of using domain or expert knowledge to create a new task, where the expert knowledge is contained in the encyclopedia. One could also hire a dog expert to label the images manually. Either way, we are exploiting some source of domain-specific knowledge (i.e. knowledge of the physiology of different dog breeds).

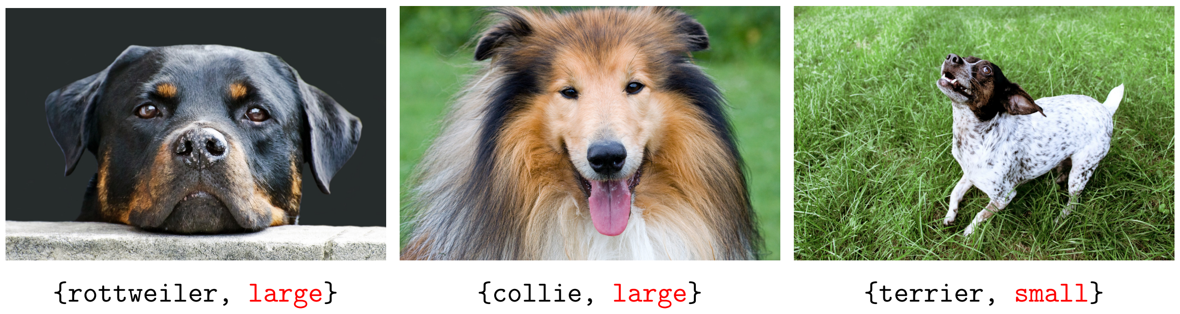


Figure 4-2: Three pictures of dogs from our dataset ($X_1$), where each picture has been labeled with separate targets: `{dog_breed, dog_size}`

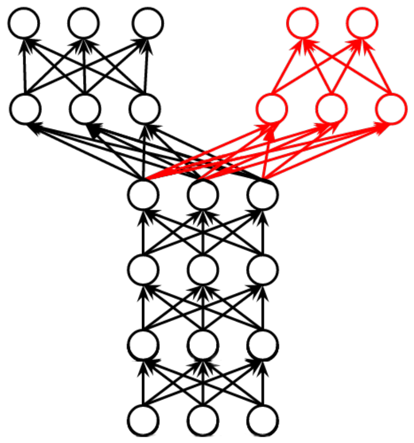


Figure 4-3: Multi-Task DNN for classifying pictures of dogs according to both `dog_breed` and `dog_size`. Any additional task by definition brings along with it additional parameters, because a subset of model parameters must be task-specific. Task-specific parameters for the new task of `dog_size` classification are shown in red.

This example came from image recognition, but now we will move onto to our overview of Multi-Task Learning in Automatic Speech Recognition. As we will see in what follows, researchers have trained Multi-Task Acoustic Models where the auxiliary tasks involve a new data domain, a new label set, or even a new mapping function.

## 4.3 Multi-Task Learning in ASR

The Multi-Task Learning discussed here deals with either acoustic modeling in Hybrid (i.e. DNN-HMM) ASR, or it deals with End-to-End ASR.[‖] The Acoustic Model accepts as input a window of audio features ($X$) and returns a posterior probability distribution over phonetic targets ($Y$). The phonetic targets can be fine-grained context-dependent units (e.g. triphones from a Hybrid model), or these targets may be simply characters (e.g. as in End-to-End approaches). The following survey will focus on how Multi-Task Learning has been used to train these acoustic models, with a focus on the nature of the tasks themselves.

Past work in Multi-Task acoustic modeling for speech recognition can be split into two broad categories, depending on whether data was used from multiple languages or just one language. In this survey, we will refer to these two branches of research as *monolingual* vs. *multilingual* approaches. Within each of those two branches, we find sub-branches of research, depending on how the auxiliary tasks are crafted. These major trends are shown in Figure (4-4), and will be discussed more in-depth below.

Within *monolingual* Multi-Task acoustic modeling we can identify two trends in the literature. We find that researchers will either (1) predict some additional linguistic representation of the input speech, or (2) explicitly model utterance-level characteristics of the utterance. When using additional linguistic tasks for a single language, each task is a phonetically relevant classification: predicting triphones vs. predicting monophones vs. predicting graphemes [12, 135, 76, 24, 25, 148]. When explicitly modeling utterance-specific characteristics, researchers either use adversarial learning to force the model to "forget" channel, noise, and speaker characteristics, or the extra task is a standard regression in order to pay extra attention to these features [139, 137, 150, 132, 98, 142, 111, 53, 27, 170].

Within *multilingual* Multi-Task acoustic modeling we can also identify two main veins of research: (1) using data from some source language(s) or (2) using a pre-trained model from some source language(s). When using data from source languages,

[‖]In the traditional, Hybrid ASR approach (i.e. DNN Acoustic Model + N-gram language model), there's not a lot of room to use MTL when training the language model or the decoder.

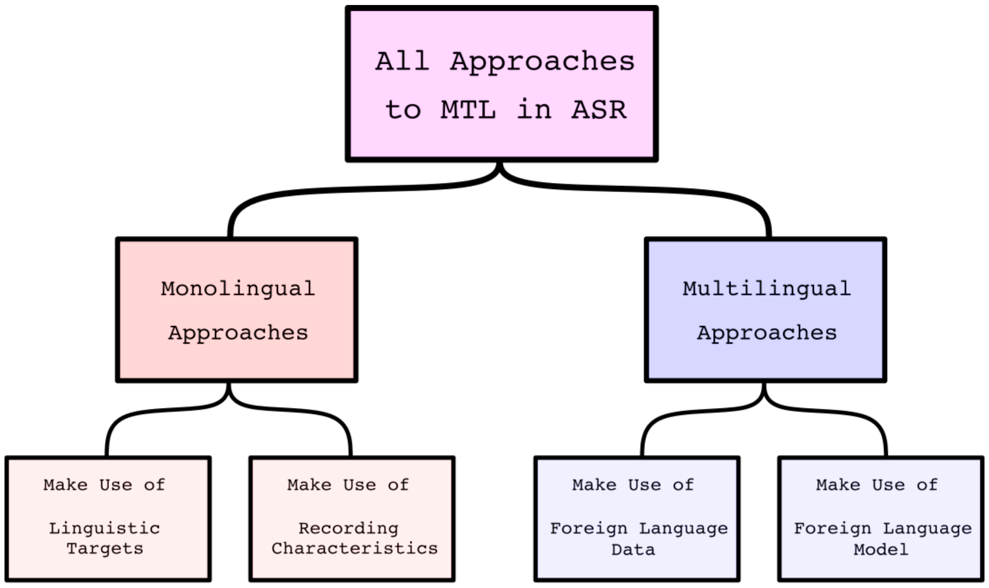

All Approaches to MTL in ASR
Monolingual Approaches
Multilingual Approaches
Make Use of Linguistic Targets
Make Use of Recording Characteristics
Make Use of Foreign Language Data
Make Use of Foreign Language Model

Figure 4-4: Major Trends in the Research on Multi-Task Learning in Automatic Speech Recognition. Here, “Recording Characteristics” refers to general characteristics of the audio file (i.e. the “recording”), not the quality of the “recording” setup or “recording” equipment.

most commonly we find researchers training a single neural network with multiple output layers, where each output layer represents phonetic targets from a different language [75, 71, 152, 99, 59, 97, 166, 123, 77, 142]. As such, these Acoustic Models look like the prototype shown in Figure (4-1a). When using a pre-trained model from some source language(s) we find researchers using the source model as either a teacher or as a feature extractor for the target language [43, 32, 60, 84, 155, 165, 68]. The source model extracts embeddings of the target speech, and then the embedding is either used as the target for an auxiliary task or the embedding is concatenated to the standard input as a kind of feature enhancement.

### 4.3.1 Monolingual Multi-Task ASR

With regards to monolingual Multi-Task Learning in ASR, we find two major tracks of research. The first approach is to find tasks (from the same language) which are linguistically relevant to the main task [141, 135, 76, 12, 4, 24, 25, 23, 11, 143, 5, 114]. These studies define abstract phonetic categories (e.g. fricatives, liquids, voiced consonants), and use those category labels as auxiliary tasks for frame-level classification.

The second major track of research in monolingual Multi-Task acoustic modeling involves explicit modeling of speaker, channel, or noise characteristics [139, 137, 150, 132, 98, 142, 111, 53, 27, 170]. These studies train the Acoustic Model to identify these characteristics via an additional classification task, or they encourage the model to ignore this information via adversarial learning, or they force the model to map data from the input domain to another domain (e.g. from noisy audio $\rightarrow$ clean audio).

#### Abstract Linguistic Units as Tasks

All the studies in this section have in common the following: the model in question learns an additional classification of the audio at the frame-level. That is, every chunk of audio sent to the model will be mapped onto a standard ASR category such as a triphone or a character *in addition to* an auxiliary mapping which has some linguistic relevance. This linguistic mapping will typically be a broad phonetic class (think vowel vs. consonant) of which the typical target (think triphone) is a member.

Good examples of defining additional auxiliary tasks via broad, abstract phonetic categories for English can be found in [135] and later [76]. With regards to low-resource languages, some researchers have created extra tasks using graphemes or a universal phoneset as more abstract classes [24, 25, 23].

A less linguistic approach, but based on the exact same principle of using more abstract classes as auxiliary targets, [12] used monophone alignments as auxiliary targets for DNN-HMM acoustic modeling (in addition to the standard triphone alignments). The authors observed that standard training on context-dependent triphones could easily lead to over-fitting on the training data. When monophones were added as an extra task, they observed $3-10\%$ relative improvements over baseline systems. The intuition behind this approach is that two triphones belonging to the same phoneme will be treated as completely unrelated classes in standard training by backpropagation. As such, valuable, exploitable information is lost. In follow up studies, [11, 143, 5] made the linguistic similarities among DNN output targets explicit via linguistic phonetic concepts such as place, manner, and voicing as well as phonetic context embeddings.

However, the benefits of using linguistic targets vary from study to study, and in their survey paper, [114] concluded that “Using even broader phonetic classes (such as plosive, fricative, nasal, . . .) is not efficient for MTL speech recognition”. In particular, they were referring to the null findings from [141].

In a similar vein, Multi-Task Learning has been used in an End-to-End framework, in an effort to encourage explicit learning of hierarchical structure of words and phonemes. Oftentimes these hierarchical phonemic levels (e.g. phonemes vs. words) are trained at different levels of the model itself [45, 131, 85, 148, 102]. Figure (4-5) displays the approach taken in [131].

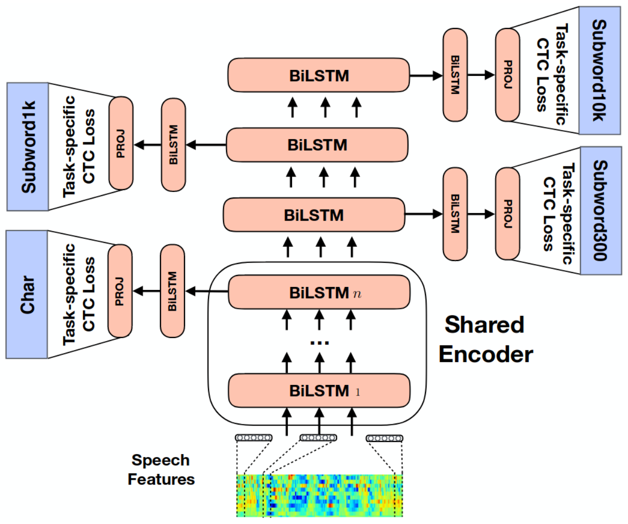


Figure 4-5: Multi-Task Hierarchical Architecture from Sanabria and Metze (2018)

All of these studies have in common the following: they encourage the model to learn abstract linguistic knowledge which is not explicitly available in the standard targets. Whatever the standard target may be (e.g. triphones, graphemes, etc.) the researchers in this section created abstract groupings of those labels, and used those new groupings as an additional task. These new groupings (e.g. monophones, sub-word units, etc) encourage the model to learn the set of underlying features (e.g. voicing, place of articulation, etc.) which distinguish the main targets.

### Regression on Better Features as New Task

Class labels are the most common output targets for an auxiliary task, but [111, 53, 27, 170] took an approach where they predicted de-noised versions of the input audio from noisy observations (c.f. Figure (4-6)). The effect of this regression was that the Acoustic Model cleaned and classified each input audio frame in real time.

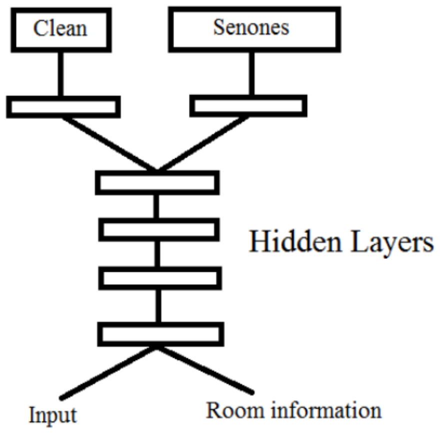


Figure 4-6: Regression and Classification Neural Network Architecture from Giri et al. (2015)

In a similar vein, [34] trained an Acoustic Model to classify standard senomes targets as well as regress an input audio frame to bottleneck features of that same frame. Bottleneck features are a compressed representation of the data which have been trained on some other dataset or task — as such bottleneck features should contain linguistic information. In a very early study, the authors in [96] predicted enhanced audio frame features as an auxiliary task (along with the speaker's gender).

### Extra Mapping Function as Extra Task

In End-to-End ASR, [82] created a Multi-Task model by adding a mapping function (CTC) to an attention-based encoder-decoder model. This is an interesting approach because the two mapping functions (CTC vs. attention) carry with them pros and cons, and the authors demonstrate that the alignment power of the CTC approach can be leveraged to help the attention-based model find good alignments faster. Along similar lines, [95] trained an Acoustic Model to make use of both CTC and Sequential Conditional Random Fields. These works did not create new labels or find new data, but rather, they combined different alignment and classification techniques into one model.

A famous example of monolingual MTL using multiple mapping functions is the most common Kaldi implementation of the so-called "chain" model from [118]. This implementation uses different output layers on a standard feed-forward model, one output layer calculating standard Cross Entropy Loss, and the other calculating a version of the Maximum Mutual Information Criterion.

### New Domain as New Task

If we consider the different characteristics of each recording as domain memberships, then any extra information we have access to (e.g. age, gender, location, noise environment), can be framed as domain information, and this information can be explicitly modeled in a Multi-Task model. Using a Multi-Task adversarial framework, [139, 137, 150] taught an Acoustic Model to forget the differences between different noise conditions, [132, 98] taught their model to forget speakers, and [142] taught the model to forget accents.

In low-resource domains, it is often tempting to add data from a large, out-of-domain dataset into the training set. However, if the domains are different enough a mixed training set may hurt performance more than it helps. Multi-Task Learning lends itself well to these multi-domain scenarios, allowing us to regulate how much influence the out-of-domain data has over parameter estimation during training. Usually we will want to down-weight the gradient from a source domain task if the source dataset is large or if the task is only somewhat related.

The researchers in [121] investigated the low-resource domain of Cantonese aphasic speech. Using a small corpus of aphasic Cantonese speech in addition to two corpora of read Cantonese speech, the researchers simply trained a Multi-Task model with each corpus as its own task (i.e. data from each corpus as classified in its own output layer). Similarly, in an effort to better model child-speech in a low-resource setting, the authors in [146] created separate tasks for classification of child vs. adult speech, in addition to standard phoneme classification.

### Discussion of Monolingual Multi-Task Learning

In this section we've covered various examples of how researchers have incorporated Multi-Task Learning into speech recognition using data from a single language. Two major threads of work can be identified: (1) the use of abstract linguistic features as additional tasks, or (2) the use of speaker and other recording information as an extra task.

With regards to the first track of work, researchers have created abstract linguistic target labels by defining linguistic categories by hand, by referring to the traditional phonetic decision tree, or by automatically finding relevant sub-word parts. Performance improvements with this approach have been found to be larger when working with small datasets [11, 23]. The intuition behind this line of work is the following: Forcing a model to classify input speech into broad linguistic classes should encourage the model to learn a set of underlying phonetic features which are useful for the main classification task.

A discriminative Acoustic Model trained on standard output targets (e.g. triphones, characters) is trained to learn that each target is maximally different from every other target. The label-space at the output layer is N-dimensional, and every class (i.e. phonetic category) occupies a corner of that N-dimensional space. this means that classes are learned to be *maximally* distinctive. In reality, we know that some of these targets are more similar than others, but the model does not know that. Taking the example of context-dependent triphones, the Acoustic Model does not have access to the knowledge that an `[a]` surrounded by `[t]`'s is the same vowel as an `[a]` surrounded by `[d]`'s. In fact, these two `[a]`'s are treated as if they belong to completely different classes. It is obvious to humans that two flavors of `[a]` are more similar to each other than an `[a]` is similar to an `[f]`. However, the output layer of the neural net does not encode these nuances. Discriminative training on triphone targets will loose the information that some triphones are more similar than others. One way to get that information back is to explicitly teach the model that two `[a]` triphones belong to the same abstract class. This is the general intuition behind this first track of monolingual Multi-Task work in speech recognition.

The second track of monolingual Multi-Task acoustic modeling involves explicit modeling of speaker, noise, and other recording characteristics via an auxiliary task. While all of these variables are extra-linguistic, studies have shown that either paying extra attention to them (via an auxiliary classification task) or completely ignoring them (via adversarial learning) can improve overall model performance in terms of Word Error Rate. This is a somewhat puzzling finding. Learning speaker information [96, 26] can be useful, but also forgetting speaker information [132, 98] can be useful.

If we think of why this may be the case, we can get a little help from latent variable theory. If we think of any recording speech as an observation that has been generated by multiple underlying variables, we can define some variables which generated the audio, such as (1) the words that were said, (2) the speaker, (3) the environmental noise conditions, (4) the acoustics of the recording location, and many, many others. These first four factors are undoubtedly influencers in the acoustic properties of the observed recording. If we know that speaker characteristics and environmental noise had an influence on the audio, then we should either explicitly model them or try to remove them altogether. Both approaches show improvement over a baseline which chooses to not model this extra information at all, but as discovered in [1], if the dataset is large enough, the relative improvements are minor.

### 4.3.2 Multilingual Multi-Task ASR

Multilingual Multi-Task ASR can be split into two main veins, depending on whether (1) the data from a source language is used or (2) a trained model from the source language is used. We find that the first approach is more common, and researchers will train Acoustic Models (or End-to-End models) with multiple, task-dependent output layers (i.e. one output layer for each language), and use the data from all languages in parameter estimation. These models typically share the input layer and all hidden layers among tasks, creating a kind of language-universal feature extractor. This approach has also been extended from multiple languages to multiple accents and dialects. The second vein of multilingual Multi-Task Learning involves using an Acoustic Model from one language as a teacher to train an Acoustic Model from some target language. Most commonly, this source model can be used to generate phoneme-like alignments on the target language data, which are in turn used as targets in an auxiliary task. More often than not, we find multilingual Multi-Task approaches used in a low-resource setting.

#### Multiple Languages as Multiple Tasks

The earliest examples of Multi-Task Learning with multiple languages can be found in [75] and [71] (c.f. Figure 4-7). These studies focused on improving performance on all languages found in the training set, not just one target language. Every language was sampled to the same audio features, and as such the neural networks only required one input layer. However, the network was trained to classify each language using language-specific phoneme targets. Taking this line of research into the world of End-to-End speech recognition, [33] showed that a CTC model trained on multiple languages, and then tuned to one specific language can improve performance over a model trained on that one language in a low-resource setting.

In a similar vein of work, instead of optimizing performance on all languages present in the training set, researchers have aimed to perform best on one particular target language. See [156] for a survey of advances in this area.

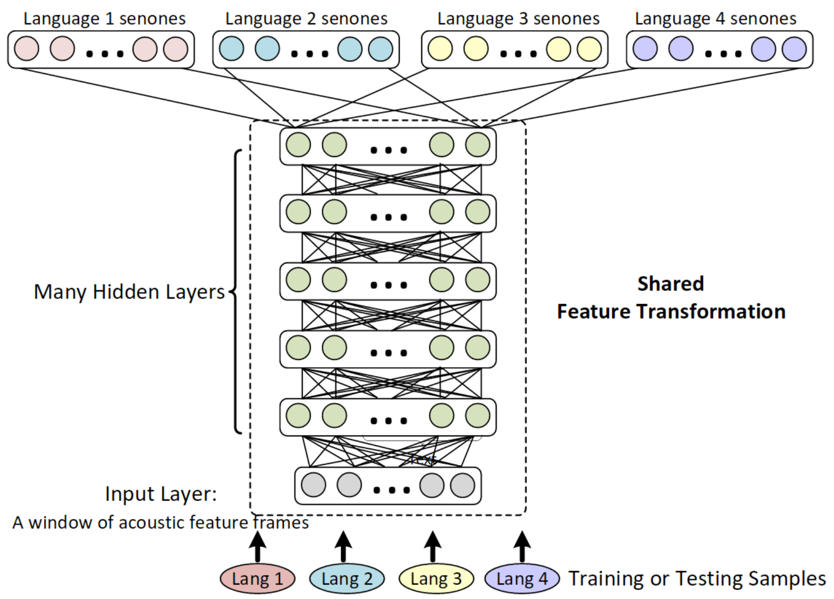


**Figure 1:** Architecture of the shared-hidden-layer multilingual DNN

Figure 4-7: Multilingual Multi-Task Acoustic Model Architecture from Huang et al. (2013)

Addressing the use-case where audio is available for a target language, but native-speaker transcribers are not easy to find, [41] employed non-native speakers to transcribe a target language into non-sense words, according to how they perceived the language. Using these non-native transcriptions in addition to a small set of native-speaker transcripts, the authors trained a Multi-Task model to predict phonemes from both native or non-native transcripts. The intuition as to why this approach works is that non-native speakers will perceive sounds from a foreign language using their native phonemic system, and enough overlap should exist between the two languages to help train the acoustic model.

In the relatively new field of spoken language translation, where speech from one language is mapped directly to a text translation in a second language, the researchers in [159, 3] created multiple auxiliary tasks by either recognizing the speech of the source language (i.e. standard ASR), or by translating the source language into one or more different languages.

### Multiple Accents as Multiple Tasks

In a vein of research which belongs somewhere between monolingual and multilingual speech recognition, the authors in [166, 123, 77, 142] used Multi-Task Learning to perform multi-accent speech recognition. The researchers in [166] trained a model to recognize English, with separate output layers for British English vs. American English. These two tasks were trained in parallel with a third task, accent identification. Combining all three tasks led to optimal output (c.f. Figure 4-8). The authors in [123] recognized phonemes of different English accents at an intermediate hidden layer, and then accent-agnostic graphemes at the output layer.

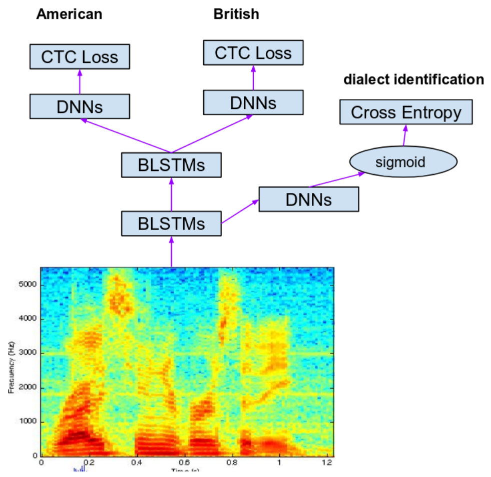


Figure 4-8: Multi-Accent Deep Neural Network Architecture from Yang et al. (2018)

### Multilingual Model as Feature Extractor

So-called *bottleneck features* have also been developed to aid in low-resource acoustic modeling, which often incorporate Multi-Task Learning. These bottleneck features are activations from a condensed hidden layer in a multilingual acoustic model. First a multilingual Acoustic Model is trained, and then data from a new language is passed through this DNN, and the bottleneck activations are appended as additional features to the original audio [43, 32, 60, 84, 155, 165]. In this way, a kind of universal feature extractor is trained on a large multilingual dataset. The bottleneck features themselves are the product of Multi-Task Learning.

### Source Language Model as Teacher

Instead of using data from multiple languages, it is possible to use the predictions from a pre-trained source model as targets in an auxiliary task. In this way, knowledge located in the source dataset is transferred *indirectly* via a source model, as opposed to *directly* from the dataset itself.

In a very recent approach, [68] trained a classifier on a well-resourced language to identify acoustic landmarks, and then used that well-resourced model to identify acoustic landmarks in a low-resourced language. Those newly discovered acoustic landmarks were then used as targets in an auxiliary task. This approach can be thought of as a kind of Multi-Task Student-Teacher (c.f. [160]) approach, where we "distill" (c.f. [72]) knowledge from one (larger) model to another via an auxiliary task.

### Discussion of Multilingual Multi-Task Learning

Surveying the literature of Multi-Task Learning in multilingual speech recognition, we can identify some common findings among studies. Firstly, we observe positive effects of pooling of as many languages as possible, even when the languages are completely unrelated. This pooling of unrelated languages may seem strange at first - why should languages as different as English and Mandarin have anything in common? However, abstracting away from the linguistic peculiarities of each language, all languages share some common traits.

All spoken languages are produced with human lungs, human mouths, and human vocal tracts. This means that all languages are produced in an acoustic space constrained by the anatomy of the human body. If we can bias a model to search for relevant patterns only within this constrained space, then we should expect the model to learn useful representations faster. Likewise, the model should be less likely to learn irrelevant correlated information about environmental noise which occurs outside this humanly-producible acoustic space. This is one intuition as to why the combination of unrelated languages is helpful: any extra language will add inductive bias for the relevant search space of human speech sounds.

Nevertheless, these studies do show a tendency that closely related languages help each other more than unrelated languages. Both [75, 33] concluded that improvement was greater when the languages were more similar. However, they still found that phonetically distinct languages were able to transfer useful bias to each other when a large enough dataset was used for the source language. With regards to how much Multi-Task Learning helps relative to size of the target language dataset, the authors in [71, 75] saw larger relative improvements when the dataset for the target language was smaller (but they still observed improvements on large datasets).

In conclusion, Multi-Task Learning for multilingual acoustic modeling yields largest improvements when: (1) the dataset for the target language is small, (2) the auxiliary language is closely related, and (3) the auxiliary language dataset is large.

### 4.3.3 Multi-Task Learning in Other Speech Technologies

In addition to Automatic Speech Recognition, Multi-Task Learning has found its way into other speech technologies. The use of Multi-Task Learning is less established in the following speech technology fields, and as such we find a very interesting mix of different applications and approaches.

#### Speech Synthesis

Working on speech synthesis, the research team in [74] used Multi-Task Learning to train neural speech synthesis systems for a single language. These models predicted both the acoustic features (spectral envelope) as well as log amplitude of the output speech. Additionally, these researchers recombined the outputs of both tasks to improve the quality of the final synthesized speech. In a similar vein, authors in [163] employed Multi-Task Learning of vocoder parameters and a perceptual representation of speech (along with bottleneck features) to train a deep neural network for speech synthesis. Working with input features which are not speech or text, but rather ultrasonic images of tongue contours, the authors in [149] trained a model to perform both phoneme classification as well as regression on the spectral parameters of a vocoder, leading to better performance on both tasks. Recently, in their work on modeling the raw audio waveform, the authors in [61] trained their original WaveNet model to predict frame-level vocoder features as a secondary task.

### Speech Emotion Recognition

Working on emotion recognition from speech, [110] demonstrate that a model can be used to identify multiple (believed to be orthogonal) emotions as separate tasks. The authors in [87] take an emotion recognition task which is typically a regression, and discover new, discrete targets (via k-means clustering) to use as targets in later auxiliary tasks. Using classification of "gender" and "naturalness" as auxiliary tasks, [81] also found improvements in spoken emotion recognition via Multi-Task training. Recently, the authors in [93] trained a model to predict the first and second most salient emotions felt by the evaluator.

### Speaker Verification

With regards to speaker verification, the authors in [92] used phoneme classification as an auxiliary task, and the authors in [40] trained speaker embeddings by jointly optimizing (1) a GAN to distinguish speech from non-speech and (2) a speaker classifier.

Combing both speech recognition and speaker verification, [26] trained an Acoustic Model to perform both tasks and found improvement. In an adversarial framework, [157] taught their model to forget the differences between domains in parallel to identifying speakers.

### Miscellaneous Speech Applications

Extending the work from Multi-Task speech recognition to Key-Word Spotting, the researchers in [109] combined parameter-copying and MTL. They first took an Acoustic Model from large-vocabulary English recognition task, re-initialized the weights immediately proceeding the output layer, and retrained with two output layers, one layer predicting only the phonemes in the Key-Word of interest, and another layer predicting senomes from the large vocabulary task.

To predict turn-taking behavior in conversation, the authors in [67] trained a model to jointly predict backchannelling and use of filler words.

Predicting the severity of speech impairment (i.e. dysarthia) in the speech of patients with Parkinson's disorder, the researchers in [153] trained a model to predict the level of impairment in various articulators (e.g. lips, tongue, larynx, etc.) as multiple tasks.

The researchers in [164] trained a model to both separate speech (from a multi-speaker monaural signal) in addition to an auxiliary task of classifying every audio frame as single-speaker vs. multi-speaker vs. no-speaker.

The researchers in [69] trained a model to both localize speech sources as well as classify incoming audio as speech vs. non-speech.

## 4.4 Conclusion

Multi-Task Learning is a technique for adding inductive bias during parameter estimation via the learning of auxiliary tasks. In order for Multi-Task Learning to work well, related tasks must be available for some given dataset.

In this survey we've discussed the main veins of work in speech recognition, and also applications in other speech technologies. With regards to speech recognition, we categorized Multi-Task approaches as being either multilingual or monolingual. Multilingual approaches exploit bias from one or more source language by either using a source dataset during training or by using predictions of a pre-trained source model. Monolingual approaches, on the other hand, use either abstract linguistic features at the acoustic frame level or general characteristics of the recording itself during training. All approaches involve the updating of task-dependent and task-independent parameters. We find it is often the case that Multi-Task Learning is applied to low-resource scenarios, where bias from related tasks can be crucial for successful model training.

# Chapter 5

# Multi-Task Acoustic Modeling via Linguistic Categories

## 5.1 Introduction

The current study investigates integrating auxiliary tasks into Multi-Task Acoustic Model training, where the tasks are hand-crafted by an expert linguist. This approach involves manipulation of a stage in the ASR pipeline: the phonetic decision tree [169]. The decision tree creates labeled data for the DNN acoustic model, and encodes contextual information about the data. This information is domain and language-specific, and becomes more fine-grained further down the tree (cf. Figure (5-1)).

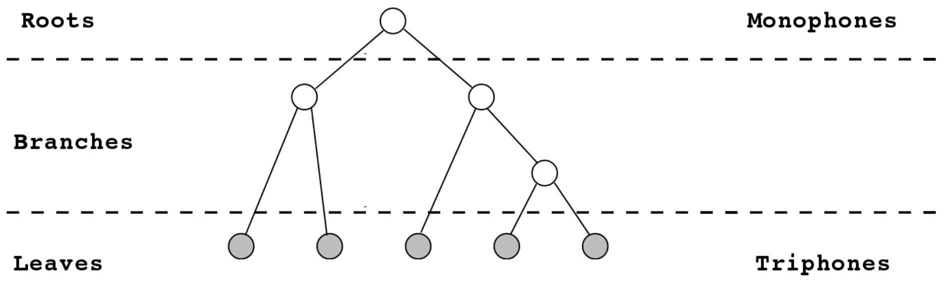


Figure 5-1: Standard Phonetic Tree Label Abstractions

The labels created by the decision tree (i.e. triphones) encode information which is very specific to the source domain (or language), and may not be the best representation of the data for an auxiliary task which leads to better generalization or language-transfer. Nodes closer to the roots of the tree represent more abstract levels of the data, and therefore encode more language-general information. However, there is a hard limit to the level of abstraction we can reach working with the traditional phonetic decision tree. The most abstract information is found at the roots, which in ASR terms is the monophone (i.e. the phoneme). There do exist more abstract representations of human speech than isolated phonemes, but the standard phonetic tree roots don't go that deep.

### 5.1.1 Linguistic Features

According to linguistic theory, phonemes are not isolated units, but rather a bundle of features such as vocal cord vibration, tongue placement, and lip rounding. Speech sounds can be modeled more abstractly than phonemes if we use these more basic phonetic features as classification targets. For example, the distinct phonemes `[d]` and `[t]` share all features except for one: the timing of vocal cord vibration. If we force a DNN classifier to recognize the overlapping features in these two phonemes, we bias the DNN to learn more basic linguistic abstractions. As such, the linguistic auxiliary tasks in this dissertation can be viewed as adding structure into phonetic trees above the level of the phoneme as in Figure (5-2).

These linguistically inspired auxiliary tasks generate targets which are defined by asking questions about higher level feature bundles instead of phoneme identity. Decision trees are still generated in the traditional Kaldi pipeline, but a split in the tree branches is made along questions such as: *Is the feature bundle to the right a [dental-plosive]?* instead of traditional questions such as: *Is the phoneme to the left a [k]?*

Past research in this area created tasks by predicting each linguistic feature in isolation.[141] This is like taking a 3-dimensional label `[x,y,z]`, and predicting `[x]` and `[y]` and `[z]` as additional tasks. The problem with this approach for human speech is that the predictive power of any one dimension is very weak. In the following,

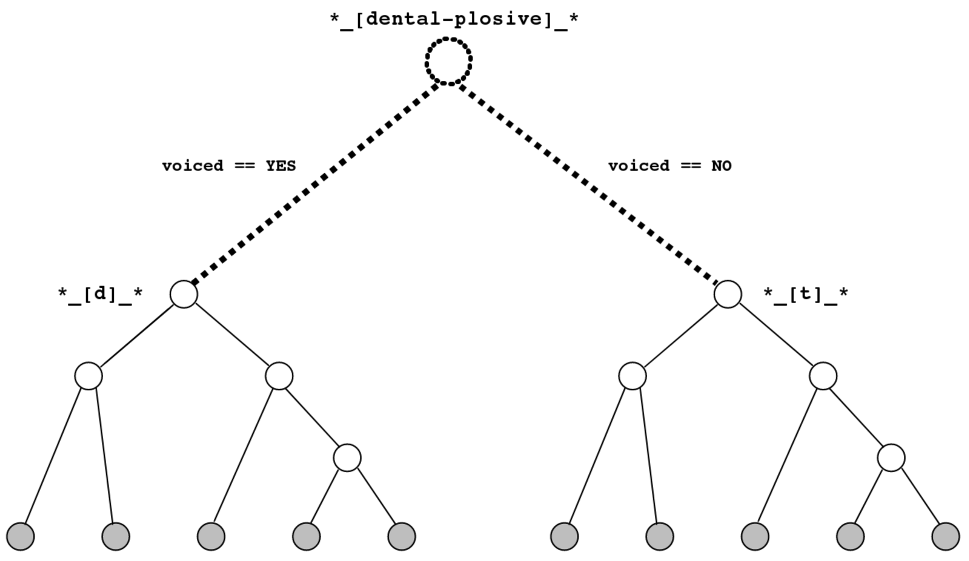


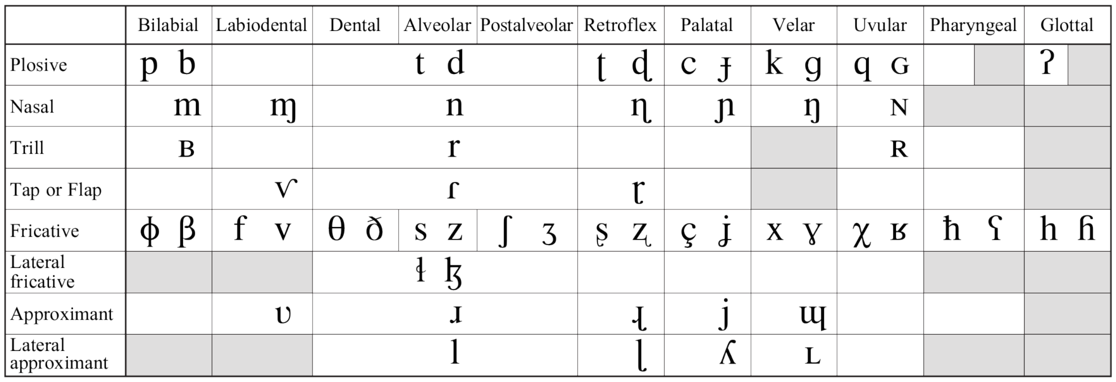
Figure 5-2: Higher Label Abstractions via Linguistics

I create tasks by *removing* one dimension for each task, projecting into `N-1`-dimensional spaces. That is, instead of `[x,y,z]` $\rightarrow$ `[x]` and `[y]` and `[z]`, I projected `[x,y,z]` $\rightarrow$ `[x,y]` and `[y,z]` and `[x,z]`. These `N-1`-dimensional spaces have more predictive power, and also force the net to learn the importance of each dimension.

The following experiments generate labels for each data point by collapsing the standard phoneme labels along one of three linguistic dimensions. The three dimensions are:

1. Place of Articulation
2. Manner of Articulation
3. Vocal Cord Vibration

These feature dimensions are represented in the International Phonetic Alphabet (Figure (5-3)) as columns, rows, and cell partitions. Place of articulation is represented via columns, manner of articulation is represented via rows, and voicing information is inside the cells of the table, where a symbol aligned to the left is voiceless and a symbol to the right is voiced.



| | Bilabial | Labiodental | Dental | Alveolar | Postalveolar | Retroflex | Palatal | Velar | Uvular | Pharyngeal | Glottal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Plosive | p b | | | t d | | ʈ ɖ | c ɟ | k ɡ | q ɢ | | ʔ |
| Nasal | m | ɱ | | n | | ɳ | ɲ | ŋ | ɴ | | |
| Trill | ʙ | | | r | | | | | ʀ | | |
| Tap or Flap | | ⱱ | | ɾ | | ɽ | | | | | |
| Fricative | ɸ β | f v | θ ð | s z | ʃ ʒ | ʂ ʐ | ç ʝ | x ɣ | χ ʁ | ħ ʕ | h ɦ |
| Lateral fricative | | | | ɬ ɮ | | | | | | | |
| Approximant | | ʋ | | ɹ | | ɻ | j | ɰ | | | |
| Lateral approximant | | | | l | | ɭ | ʎ | ʟ | | | |

Figure 5-3: International Phonetic Alphabet (2015)

In order to generate a new task, all information from one dimension is removed from the phonetic dictionary. A task defined with labels from collapsed targets therefore has no explicit access to the missing feature dimension. By collapsing the IPA columns, we remove information about place of articulation. By collapsing the rows, we remove information about manner of articulation. By collapsing each cell's internal categories (such that there is no left vs right distinction), we remove information about voicing.

For example, after removing voicing information, the phonemes [p] and [b] merge into one phoneme (i.e. a bilabial plosive). Likewise, the phonemes [t] and [d] would merge into one phoneme (i.e. an alveolar plosive). After removing information about place of articulation, all the phonemes in the Nasal row would merge into one phoneme (i.e. a voiced nasal). The phonemes in the Plosive row would be merged into two new categories (i.e. voiced plosives or un-voiced plosives). More details of these operations are presented below (c.f. Figures (5-5,5-6,5-8).

## 5.2 Training Procedure

The training procedure contains two logical steps:

1. Generate multiple sets of labels (i.e. different tasks).

2. Train Acoustic Model via Multi-Task Learning with the new labels.

### 5.2.1 Generate Alignments

In order to generate new labels for our data, we need to first train GMM acoustic models. There are many components to the GMM training pipeline, but the following experiments make a change to only one part of the pipeline: the phonetic dictionary (which dictates the questions available to the phonetic decision tree). GMM Acoustic Models are trained as outlined in Chapter (2), and as such the reader can reference that information for the entire training pipeline. Here I will go into detail on the changes made to the phonetic dictionary, which allow us to infuse abstract linguistic knowledge into the training process, resulting in linguistically informed training labels for our Multi-Task Learning later on.

The phonetic dictionary defines the pronunciation of each word in terms of a phoneme set. An excerpt from the actual phonetic dictionary used to train Librispeech is displayed in Figure (5-4). This phonetic dictionary file is two columns wide, and as many rows as possible pronunciations of words.* The written forms of words are located on the left and their pronunciations on the right. The pronunciations are made up of the phoneme set decided by the researcher. For well-resourced languages there exist standard phoneme sets and phonetic dictionaries, such as the CMUDict for American English (used in this study). [158]

```
ABANDON    AH0 B AE1 N D AH0 N
ABANDONED  AH0 B AE1 N D AH0 N D
ABBEY      AE1 B IY0
ABBOT      AE1 B AH0 T
```

Figure 5-4: Standard American English Lexicon.

For each linguistically defined label set, a new phonetic dictionary file was created by collapsing phonemes along some dimension. Figure (5-5) displays a phonetic

*The number of rows may be greater than the number of words, because one word may have two or more possible pronunciations.

dictionary collapsed on voicing; the distinction between T and D is lost.

```
ABANDON     AH0 P AE1 N T AH0 N
ABANDONED   AH0 P AE1 N T AH0 N T
ABBEY       AE1 P IY0
ABBOT       AE1 P AH0 T
```

Figure 5-5: Lexicon without voicing information.

The collapsing procedure is performed on American English, but can be easily extended to any new language (given knowledge of the language's phonology). The specifics of how the lexicons were modified for each task are discussed below. Given each new phonetic dictionary (i.e. new phoneme set) a GMM-HMM system was trained and used to align the new labels to the training data.

The GMM-HMM training procedure was very standard. Monophones were trained from a flat-start and aligned to the training data iteratively over 25 iterations via the Baum-Welch algorithm. These monophones were allotted a total of 1,000 possible Gaussian components. Given the alignments from the monophone labels, a context-dependent triphone system was trained over another 25 iterations of Baum-Welch training. The splits in the phonetic decision tree were made not along phonemes (as is standard), but along the newly defined feature bundles (i.e. collapsed phonemes). The decision tree was allotted a total of 1,000 possible leaves and 2,000 possible Gaussian components. The final alignments from the triphone system are then used as an auxiliary task in the following Multi-Task Learning DNN training framework.

### Voicing Auxiliary Task

Voicing information is removed from the data labels. The only remaining information comes from place and manner. Figure (5-6) displays the phoneme categories which are collapsed from the removal of voicing information.

|  |  |  |  |
|---|---|---|---|
| B P | --> | P | bilabial plosives |
| CH JH | --> | CH | alveo-palatal affricates |
| D T | --> | T | alveolar plosives |
| DH TH | --> | TH | interdental fricatives |
| F V | --> | F | labio-dental fricatives |
| G K | --> | G | velar plosives |
| S Z | --> | S | alveolar fricatives |
| SH ZH | --> | SH | alveo-palatal fricatives |

Figure 5-6: Phonemes collapse after removal of voicing information.

### Place Auxiliary Task

Place of articulation information is removed from the data labels. The only remaining information comes from voicing and manner. Figure (5-7) displays the phoneme categories which are collapsed from the removal of place information.

|  |  |  |  |
|---|---|---|---|
| F TH SH S HH | --> | F | voiceless fricatives |
| V DH Z ZH | --> | V | voiced fricatives |
| P T K | --> | P | voiceless plosives |
| B D G | --> | B | voiced plosives |
| M N NG | --> | N | voiced nasals |
| L R | --> | R | voiced laterals |
| Y W | --> | Y | voiced approximants |

Figure 5-7: Phoneme-collapse after removal of place information.

### Manner Auxiliary Task

Manner of articulation information is removed from the data labels. The only remaining information comes from voicing and place. Figure (5-8) displays the phoneme categories which are collapsed from the removal of manner information. I decided to make two groups of voiced alveolars because collapsing all of them into one target has very little phonetic justification.

```
B M V W --> W         voiced labials
P F     --> P         voiceless labials
D Z     --> D         voiced alveolar
N L R   --> R         voiced alveolar2
T S     --> T         voiceless alveolar
ZH JH   --> JH        voiced postalveolar
SH CH   --> CH        voiceless postalveolar
NG G    --> G         voiced velar
```

Figure 5-8: Phoneme-collapse after removal of manner information.

### 5.2.2 Train DNN Acoustic Model

Given the multiple sets of aligned audio data, a Deep Neural Network Acoustic Model is trained via Multi-Task Learning to classify audio frames as various labels. For example, given two sets of alignments of labels to data, a single DNN will classify each training example as two separate targets. Training is performed via standard Stochastic Gradient Descent. Below is an example of a DNN with two tasks. As shown in Figure (5-9) left task represents standard labels and the right task represents labels from collapsed phonemes (collapsed vocal cord vibration).

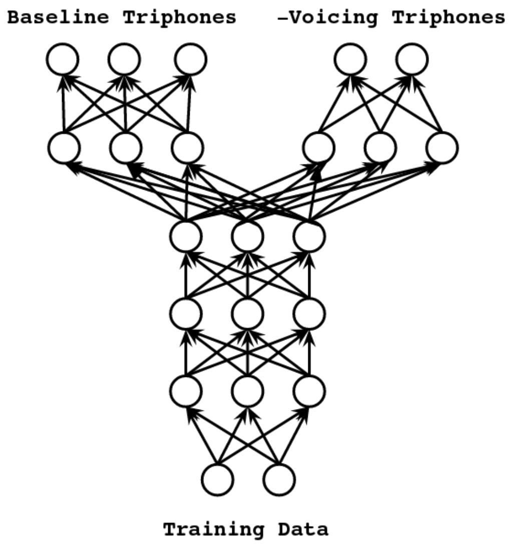


Figure 5-9: Multi-Task DNN Acoustic Model

The DNNs in the current study have five hidden layers, 500 nodes per hidden layer, and each task has its own output layer and penultimate output layer. The activations at each layer are ReLU. Stochastic Gradient descent was used to update parameters over two epochs of all the data.

## 5.3 Data

To measure how well the resulting Acoustic Models perform, I use a set of speech corpora which exhibit some interesting differences between training and testing data. These training and testing data will differ in either who the *speaker* is, or what *language* is being spoken. The following table shows which data sets are used for each audio condition (Table (5.1)). The data used all come from audiobooks, either English language or Kyrgyz language. As such, the results allow us to safely exclude speaking styles or noise conditions as cause of differences. Librispeech-A contains 4.86 hours of audio, and Librispeech-B contains 0.5 hours of audio (from two speakers not represented in Librispeech-A).[†] Kyrgyz Audiobook contains 1.6 hours of audio.

| | Corpus | |
|---|---|---|
| | **Train** | **Test** |
| **Speaker** | LibriSpeech-A | LibriSpeech-B |
| **Language** | LibriSpeech-A | Kyrgyz Audiobook |

Table 5.1: Speech Corpora

## 5.4 Monolingual Experiments

Three sets of new labels are generated via the phonetic decision tree. The new tasks (i.e. new triphone alignments) are added during training to the neural net Acoustic Model as auxiliary tasks. The main task is always the same: English triphones. In these low-resource, monolingual experiments, the goal is to improve generalization for

[†]LibriSpeech-A and LibriSpeech-B are both drawn from the Mini LibriSpeech corpus, which is freely available at `https://openslr.org/31/`.

unseen speakers. The intuition explored here is that higher-level acoustic features from one speaker will generalize better to another speaker. During training, the multiple tasks are trained in parallel, but at inference time, only the main task is used to decode speech from new speakers. In the following, experiments are shown for auxiliary tasks modeled as either monophones and triphones. The main task is always modeled as triphones.

### 5.4.1 Results

#### Gaussian Model Analysis

Each set of labels is generated by a separate GMM system. As such, we can investigate the performance of these individual Gaussian systems *before* we use the labels to train the neural net acoustic model. These results are interesting because they demonstrate how appropriate the new labels are for the end-task of speech recognition. Table (5.2) shows the performance of each of these models on held out data from two speakers (i.e. one man, one woman). All decoding was performed with the same unigram language model trained on the transcripts of the data. A unigram is a very weak language model, and as such the performance of the Acoustic Models is highlighted.

| | **Baseline** | **-Voicing** | **-Place** | **-Manner** |
|---|---|---|---|---|
| **%WER** | 51.53 | 55.90 | 66.85 | 69.74 |

Table 5.2: Gaussian Mixture Model Performance

As we can see, the original phoneme-based model performs the best of all with a Word Error Rate of 51.53%. Even though this model has more individual phonemes, it does not have more parameters than any other model. All models were allotted the same number of triphones (i.e. the same number of leaves in the phonetic decision tree). The differences in performance then, don't come from model complexity but from the predictive power of the newly defined dimensions. Furthermore, the performance of the models does not correlate with the number of phonemes which were collapsed. Specifically, the phoneme set was reduced by 8 when voicing was collapsed, by 11 when manner was collapsed and by 15 when place was collapsed. Even though more phonemes were collapsed by removing place information, those experiments outperformed manner collapsing, where fewer phonemes were lost.

The best performing "linguist-crafted" GMM model had all voicing information removed. This model could only rely on manner of articulation and place of articulation information to assign labels. In isolation, these two dimensions are good predictors, and this model experienced only a 4.37% decrease in accuracy compared to the baseline (trained on traditional phonemes).

**Multi-Task DNN Results**

The results for the monolingual experiments are displayed in Table (5.3). The rows in this table correspond to different label sets (i.e. different collapsings of the phonetic dictionary), and the columns correspond to the representation of the auxiliary task labels (triphones vs. monophones).

There are three ways we collapsed labels: removing *voicing* information from the phonetic dictionary, removing *place* information from the phonetic dictionary, or removing *manner* information from the phonetic dictionary. The results show experiments where every possible combination of auxiliary tasks is added via Multi-Task Learning.

We first add each task as a separate auxiliary task such that the DNN has two output layers (main + aux) during training, yeilding three experiments. Next, we add all possible combinations of any two tasks (main + aux1 + aux2), yeilding in three more experiments. Finally, we append every available auxiliary task (main + aux1 + aux2 + aux3), yeilding one more experiment. As such, below is presented a total of seven experiments in addition to the Single Task Baseline (c.f. Table (5.3)).

The two columns in Table (5.3) correspond to Word Error Rates for either monophones or triphones. Both monophones and triphones were investigated because monophones are direct representations of the phonetic dictionary, while triphones incorporate context. The monophones best represent the linguistic information from the phonetic dictionary. The triphones are clustered algorithmicly, and as such may differ from the original linguistic features encoded in the phonetic dictionary.

| | **WER%** | |
|---|---|---|
| **Auxiliary Tasks** | Triphones | Monophones |
| STL Baseline | 41.67 | |
| Voice | **41.16** | 42.36 |
| Place | 42.66 | **40.61** |
| Manner | 42.03 | 41.70 |
| Voice + Place | 42.90 | **41.49** |
| Voice + Manner | 42.45 | 42.66 |
| Place + Manner | 42.66 | 41.82 |
| Voice + Manner + Place | 42.42 | 42.72 |

Table 5.3: Model performance on speakers unseen during training. Results are measured as percent Word Error Rate. All models were trained on the same 4.86 hours of Librispeech audio, and tested on the same two held out speakers. These experiments illustrate the model's ability to generalize to new speakers. Bolded values indicate improvement over the baseline.

Starting with results from the triphone experiments (i.e. the left column), as we can see in Table (5.3), the addition of most auxiliary tasks does not guarantee improvement over the baseline Single-Task Learning model. Models with lower WERs than the baseline are displayed in bold. Out of the seven experiments with triphones, only one showed improvement over the baseline: removing voicing information from the targets (i.e. .51% improvement). Both other auxiliary tasks (collapsed place and collapsed manner) resulted in worse WER's compared to the baseline. By adding any two auxiliary tasks during training, performance dropped by about 1% compared to the baseline. Combining all three auxiliary tasks dropped performance by 0.75%.

With regards to the addition of auxiliary tasks modeled as monophones (i.e. the right column in Table (5.3)), we find two conditions which improve performance over the baseline: (1) a 1.06% improvement by adding a task without place information, and (2) a 0.18% improvement by adding two additional tasks (one task without place and one task without voicing).

Overall, for both monophones and triphones, we do not observe the kind of improvement we would expect from Multi-Task Learning with related tasks. All these tasks are clearly related, and related tasks should lead to improvement on the main task. One reason these tasks may not be helping as much as we'd expect is the following: *the main task does not have priority in influence over the hidden weights*. Every iteration of backpropagation sends an update back through the shared hidden layers. The amount of update depends on the magnitude of the gradient, and the magnitude of the gradient depends on the difference between the activation of the final layer of the neural network and the target vector. These target vectors are "one-hot" vectors, and therefore, the activations are encouraged to produce a value of 1.0 for the correct label, and values of 0.0 for all other labels. We can explicitly favor the main task during gradient descent by adjusting the target one-hot vectors such that the main task produces a larger gradient. In the following, all the auxiliary (i.e. non-main) tasks are trained on one-hot vectors where the correct label has a value of 0.33 instead of 1.0. This means that if the task predicts the correct label with a probability of 0.33, no gradient will be sent through the weights as an update. The main task, however, will continue to update weights, striving to achieve an activation of 1.0 for the target label. The following results come from the exact same experimental setup as above, but where each auxiliary task is down-weighted. The results for weighted-auxiliary task experiments are shown in Table (5.4).

| | **WER%** | |
|---|---|---|
| **Auxiliary Tasks** | Triphones | Monophones |
| STL Baseline | 41.67 | |
| Voice | **41.00** | **40.43** |
| Place | **41.37** | **41.46** |
| Manner | **40.43** | **41.34** |
| Voice + Place | **41.31** | **41.28** |
| Voice + Manner | **41.25** | 42.18 |
| Place + Manner | 42.03 | 42.48 |
| Voice + Manner + Place | **41.64** | 41.88 |

Table 5.4: **Weighted Task** Model performance on speakers unseen during training. Results are measured as percent Word Error Rate. All auxiliary tasks were down-weighted by 1/3, such that a training example from the main task had three times as much influence over hidden layers during backprop compared to any other, auxiliary task. These experiments illustrate the model's ability to generalize to new speakers. Bolded values indicate improvement over the baseline.

As you can see in comparison to the unweighted experiments in Table (5.3), by down-weighting the auxiliary tasks we see a general improvement in Word Error Rates. All experiments in which a single auxiliary task was added improved over the baseline. The two-auxiliary task and three-auxiliary task experiments did not all improve over the baseline, but they did perform on average better than their unweighted counterparts in Table (5.3). It may be the case that given the addition of multiple auxiliary tasks, they should be down-weighted even more, such that their combined influence does not interfere with the main task.

## 5.5 Multilingual Experiments

### 5.5.1 Results

#### Gaussian Model Analysis

In the multilingual experiments, there is only one set of labels trained for the target language, Kyrgyz. All the collapsing of phonemes happens for the source language (i.e. English) such that the source language is represented more abstractly to maximize transfer to the target language.

The same GMM models from the monolingual experiments above were used to generate labels for the English data. The Kyrgyz data comes from 1.6 hours of audiobook recordings. The language model used in Kyrgyz decoding was trained on a Wikipedia dump, and is a bigram model. The Kyrgyz GMM used to generate triphone alignments produces a Word Error Rate of 57.17% on the held-out testing data.

| | **Kyrgyz Triphones** |
|---|---|
| **%WER** | 57.17 |

Table 5.5: Kyrgyz Gaussian Mixture Model Performance

| | **Baseline** | **-Voicing** | **-Place** | **-Manner** |
|---|---|---|---|---|
| **%WER** | 51.53 | 55.90 | 66.85 | 69.74 |

Table 5.6: LibriSpeech Gaussian Mixture Model Performance

**Multi-Task DNN Results**

The results on Kyrgyz language held-out test data are shown in Table (5.7). These results come from decoding using a neural network which predicts only Kyrgyz triphones from audio. That is, only the main task is used in decoding. The additional tasks, like above for the monolingual experiments, are only used during training to update the shared hidden layers in parallel. There is one additional task here which does not apply to the monolingual experiments: an auxiliary task of English phonemes. The English phonemes are added as a kind of second baseline, because they test the effects of adding specifically linguistic information via Multi-Task Learning, or just the effects of adding a second language into training.

The rows in Table (5.7) represent different combinations of auxiliary tasks, and the columns represent the level of source language labels (monophones or triphones). The first set of results shown here trained Multi-Task neural networks where each task was weighted equally. That is, all one-hot target labels aimed for a value of 1.0.

| | **WER%** | |
|---|---|---|
| **Auxiliary Tasks** | Triphones | Monophones |
| STL Baseline | 53.07 | |
| Phonemes | 53.95 | **52.78** |
| Voice | 54.05 | 53.85 |
| Place | 55.22 | 53.95 |
| Manner | 53.37 | 53.27 |
| Voice + Place | 55.22 | 53.46 |
| Voice + Manner | 55.12 | 53.46 |
| Place + Manner | 55.51 | 53.66 |
| Voice + Manner + Place | 54.15 | 54.44 |

Table 5.7: Language Transfer experiments. Results are measured as percent Word Error Rate. These experiments test the model's ability to transfer useful bias from English to Kyrgyz via shared hidden layers. The main task was 1.6 hours of Kyrgyz audio. The auxiliary tasks all came from the English language (Mini Librispeech corpus), and the labels were modeled via various linguistic concepts. Bolded values indicate improvement over the baseline.

Only one experimental condition produced an improvement of 0.29% over the baseline: standard English monophones. As above, this has to do with the relative weighting of tasks. However the problem is worse here, because the source and target language come from uneven datasets. The target language (Kyrgyz) has a dataset of 1.6 hours, whereas the English dataset is 4.8 hours long. This means that if every target label has a one-hot vector with 1.0 as the target value, then the English dataset exerts, for a single epoch, over three times as much influence during gradient descent compared to the target task (Kyrgyz). When we down-weight the target vectors for the auxiliary tasks (i.e. the English tasks), we see a very different picture in Table (5.8).

| | **WER%** | |
|---|---|---|
| **Auxiliary Tasks** | Triphones | Monophones |
| STL Baseline | 53.07 | |
| Phonemes | **51.80** | **51.61** |
| Voice | **52.39** | 53.46 |
| Place | **51.90** | **52.29** |
| Manner | **52.00** | **51.80** |
| Voice + Place | **52.68** | **52.78** |
| Voice + Manner | **51.22** | **51.32** |
| Place + Manner | **50.83** | 53.66 |
| Voice + Manner + Place | **52.78** | **52.39** |

Table 5.8: **Weighted Task** Language Transfer experiments. Results are measured as percent Word Error Rate. These experiments test the model's ability to transfer useful bias from English to Kyrgyz via shared hidden layers. The main task was 1.6 hours of Kyrgyz audio. The auxiliary tasks all came from the English language (Mini Librispeech corpus), and the labels were modeled via various linguistic concepts. Bolded values indicate improvement over the baseline.

In these multi-lingual experiments where the auxiliary tasks are down-weighted, we observe improvement in almost every case.

## 5.6 Conclusion

These results show that using linguistic features as targets for auxiliary tasks in Multi-Task Learning can improve performance on the main task, and that the relative weighting of tasks must be taken into consideration. Notably, we found that a linguistic representation is better for language transfer from source to target language.

# Chapter 6

# Multi-Task Acoustic Modeling via Non-Linguistically Engineered Tasks

## 6.1 Introduction

Performance for a low-resource language on speech recognition can be improved by adding training data from another, resource-rich language. In the Multi-Task Learning (MTL) framework, data from a related source domain updates hidden layers in parallel with the target task [20]. In ASR, the targets for this additional language have typically been states of context-dependent triphones, defined by some tree clustering algorithm [75, 71, 59].

MTL works in situations when tasks are related. For example, the two image recognition tasks (1) find doors and (2) find doorknobs perform better when trained together, because doorknobs are highly predictive of doors, and vice versa [20]. By forcing a neural net to recognize both objects in the same image, the hidden layers will be biased towards more generalizable representations of the data.

Doors and doorknobs are obviously related, but in general it is difficult to create related tasks for a new classification problem. The current study investigates auxiliary tasks which are not hand-crafted by an expert or human, but can be automatically extracted from a stage in the traditional ASR pipeline (i.e. the phonetic decision tree [169]). The decision tree creates labeled data for the DNN acoustic model, and encodes contextual information about the data. This information is language-specific, and gets more fine-grained further down the tree.

The current research builds off the intuition that the labels created by the decision tree (i.e. triphones) encode information which is very specific to the source language, and may not be the best representation of the data for language-transfer. Nodes closer to the roots of the tree represent more abstract levels of the data, and therefore encode more language-general information. Given a phonetic decision tree in a source language (English) the current study investigates more abstract data labels for MTL transfer to a target language (Kyrgyz).

In addition to phonetic detail in the training labels, the relative weighting of the source task vs. target task during backprop affects performance outcomes. If tasks come from separate datasets, the task with the biggest dataset will have most influence during backprop. To avoid an auxiliary task with a large dataset dominating the target task in training, we can weight training labels, such that more important examples will have a larger gradient.

## 6.2 Contributions of this Chapter

The following chapter investigates low-resource multilingual Acoustic Model training with Multi-Task Learning (MTL) for Automatic Speech Recognition. The main question of this research is: *What is the best way to represent a source language with MTL to improve performance on the target language?* The two parameters of interest are (1) the level of detail at which the source language is modeled, and (2) the relative weighting of source vs. target languages during backprop.

Results show that when the source task is weighted *higher* than the target task, a *more* detailed task representation (i.e. the triphone) leads to better performance on the target language. On the other hand, when the source task is weighted *lower*, then a *less* detailed level of source task representation (i.e. the monophone) is better for performance in the target language. Given all levels of detail in the source task, a 1-to-1 weighting ratio of source-to-target leads to best results on average. After an analysis of performance on training and validation data, we see that less detailed (i.e. more simple) tasks are easier to learn, and as such, they quickly settle into a good local minimum, and are less likely to budge. However, this local minimum may not be best for the target task.

The target language is Kyrgyz, and the source language is English. Both data come from audiobooks, English from Mini LibriSpeech [108] and Kyrgyz from the Bizdin.kg project. Kyrgyz is a Turkic language spoken mostly in Central Asia, where it has official language status in the Kyrgyz Republic. Kyrgyz is spoken by approximately 5 million people world-wide .

## 6.3 Background

Past work on MTL for acoustic modeling can be divided into two main categories: monolingual vs. multilingual. Multilingual MTL acoustic modeling involves training a single DNN with multiple output layers, where each output layer represents triphones from one language. Monolingual MTL acoustic modeling involves designing multiple tasks for a single language, where each task is linguistically relevant (e.g. triphones vs. monophones vs graphemes). Multilingual MTL aims for domain transfer, but monolingual MTL aims for robust generalization from the training data.

The earliest examples of MTL with multiple languages can be found in [75] and [71]. They were interested in improving performance on all languages, not just one target language. More recently, [59] studied the effect of adding data from a single, well-resourced language to some low-resourced language.

With regards to monolingual MTL, research has aimed to find tasks (from the same language) which are phonetically relevant to the main task [12]. The aim being to improve generalization to new data. Both [135] and later [76] looked at a very similar approach, defining additional auxiliary tasks in MTL via broad, abstract phonetic categories for English. With regards to low-resource languages, [24] and later [25] created extra tasks using graphemes or a universal phoneset as extra targets.

## 6.4 DNN-Hybrid Training as Model Transfer

The standard DNN-Hybrid approach uses an initial GMM-HMM system to generate the labeled data for supervised DNN training. This reliance of the DNN on GMM alignments is actually a form of model transfer, where the DNN is trained to perform the exact same classification as its GMM predecessor. The DNN not only learns the frame alignments from the individual GMMs, but also the structure of the phonetic decision tree used to define the labels, as shown in Figure (6-1)*.

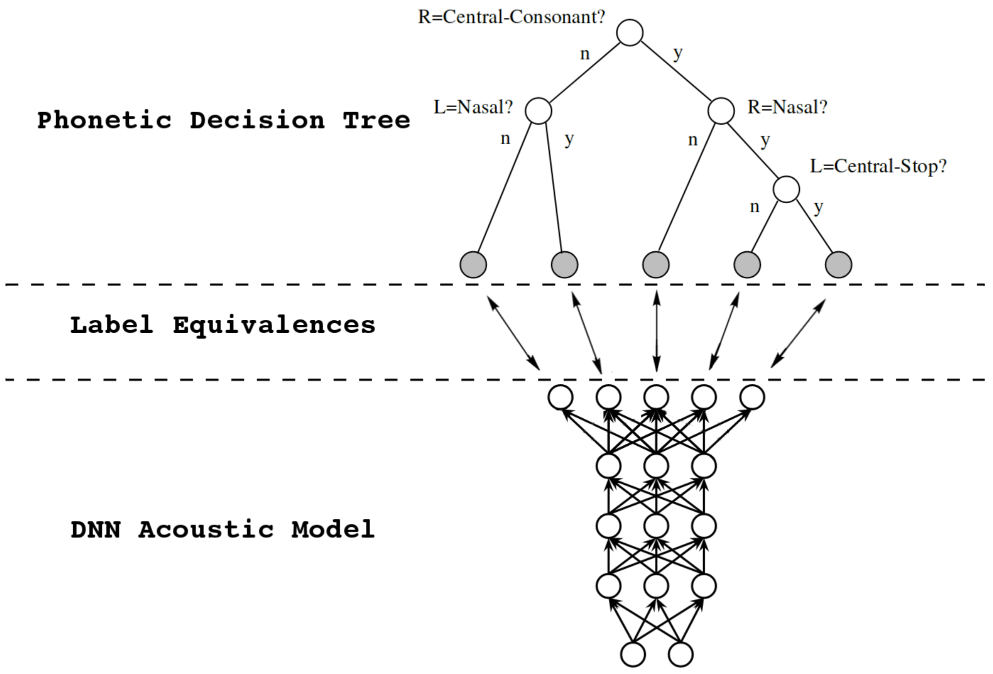


Figure 6-1: GMM→DNN Model Transfer

However, all hierarchical knowledge inherent to the decision tree is lost to the DNN. The DNN only sees the leaves of the tree, but none of the relationships among those leaves. The branches and roots are lost, which contain more abstract, language-general knowledge. The current study extracts the hierarchical knowledge of this tree via MTL, by modeling various levels of the tree as separate tasks.

*The original decision tree graphic comes from [168], and the original neural net graphic comes from [71]

## 6.5 Experiments

The following experiments tease out (1) the level of detail at which the source language should be modeled and (2) the amount of weighting which should be given to the target language training examples. With regards to the first question, the experiments here are crafted to answer the question: *How much phonetic detail should the source language be modeled at to best transfer inductive bias to the target language?* We can model the source language with lots of contextual detail (i.e. the triphone), with abstracted, context-independent detail (i.e. the monophone), or somewhere in between (what I dub the "half"-phone) (cf. Figure (6-2)[†]).

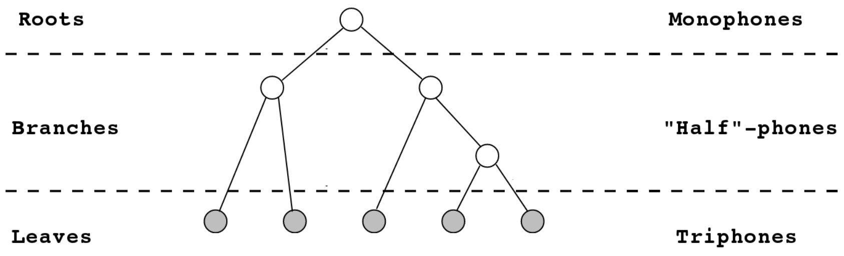


Figure 6-2: Logical Tree Parts

The second question is: *How should the target and source languages be weighted during training?* If we train two languages in parallel, the language with more data will dominate in the fight for influence over shared hidden layers during backprop. To my knowledge, the importance of relative weighting has not been investigated in ASR acoustic modeling (although thoroughly discussed in Caruana's 1997 dissertation [20]).

To investigate weighting further, I examine the following source-to-target weighting schemes: 1-to-2, 1-to-1, and 2-to-1. These weights are instantiated during training via a weight to the target output label, where the label is a one-hot vector. For example, given 1000 hours of source language and 1 hour of target language, to achieve a 1-to-1 ratio in training, I would multiply the target labels from the target language by 1000, resulting in target vectors such as $[0, 0, 0, 0, 1000, 0, 0, \ldots]$ instead

[†]Original figure from [168].

of $[0, 0, 0, 0, 1, 0, 0, \ldots]$.

### 6.5.1 Data

Two speech corpora are used in the following experiments:

1. $\approx$ 5 hours of English (4.86 hours of Mini LibriSpeech)

2. $\approx$ 1.5 hours of Kyrgyz (1.59 hours of audiobook)

### 6.5.2 Model Building

All models were build using the Kaldi toolkit as Time-Delay Neural Networks (TDNNs) via the `nnet3` approach [117, 113]. [‡]

In GMM training, monophones (for each language) were allotted 1,000 Gaussian components, and trained over 25 iterations of EM. These monophones were then expanded into context-dependent triphones via a phonetic decision tree, with a maximum of 2,000 leaves & 5,000 Gaussians (Mini LibriSpeech reached 1584 leaves, and Kyrgyz reached 752). The resulting tied-state clusters (i.e. leaves) are then trained as context-dependent triphones over 25 iterations of EM. Given the alignments from the GMM-HMM models, a 5-layer, 500-dimensional TDNN is trained over 10 epochs of backprop on a single GPU instance.

Each auxiliary task is implemented as a separate output layer along with a separate, penultimate hidden layer. All other hidden layers of the TDNN are trained in parallel. A declining learning rate was used, with an initial $\alpha_{initial} = 0.0015$ and a final $\alpha_{final} = 0.00015$.

During testing, *only* the main task is used. The additional tasks are dropped and the baseline Kyrgyz triphones are used in decoding. This highlights the purpose of the extra tasks: to force the learning of robust representations in the hidden layers during training; they serve as "training wheels" which are then removed once the net is ready.

[‡]All code made be found at `https://github.com/JRMeyer/multi-task-kaldi`.

### Baseline Model

All the following architectures will be compared to the performance of a baseline model of identical architecture (5 hidden layers, 500-dimensional layers, ReLU activations, same linear objective function). The output targets are standard context-dependent triphones trained on Kyrgyz audio. To account for any advantage multiple output layers may bring about, the baseline contains two output layers, where the tasks are identical. In this way, random initializations in the weights and biases for each task are accounted for.

### Auxiliary Tasks

The auxiliary tasks all train on English language data from the Mini LibriSpeech corpus. Investigating the intuition that labels generated by a standard triphone phonetic decision tree are not the best representation of data for transfer learning, the auxiliary tasks here investigate different levels in the decision tree's branches. I split the LibriSpeech phonetic decision tree into three logical parts (cf. Figure (6-2)):

1. roots (standard monophones)

2. branches (custom "half"-phones)

3. leaves (standard triphones)

The "half"-phones were created by halving the optimal number of leaves from the triphone system (i.e. 1584 leaves) and re-training a new GMM-HMM system with half the optimal number of leaves (1/2 * 1584 = 792 leaves). All the other parameters were left unchanged (number of Gaussian components, iterations of EM, etc.). An overview of the auxiliary tasks can be found in Table (6.1).

By forcing the neural net to recognize higher levels in the English source tree, the net will learn representations which are more abstract, and therefore more likely to be relevant to another language.

Table 6.1: Auxiliary Tasks

| **Logical Tree Part** | **Level of Phonetic Detail** | **№ of Tasks** |
|---|---|---|
| Roots | Monophones | 1 |
| Branches | Half-phones | 1 |
| Leaves | Triphones | 1 |
| Lower Tree | Monophones + Half-phones | 2 |
| Upper Tree | Half-phones + Triphones | 2 |
| Whole Tree | Monophones + Half-phones + Triphones | 3 |

### Weighting Procedure

The addition of each above task adds approximately 5 hours of training data to the standard training of a Single Task Model on Kyrgyz. As such, a weighting procedure was used to balance the relative influence of source vs. target training data on backprop. For example, to reach a one-to-one ratio, where one hour of Kyrgyz is equal to one hour of English, I multiplied every Kyrgyz target one-hot vector by 3.06. The exact weighting scheme is shown in Table (6.2).

Table 6.2: Source:Target Data Weighting Scheme

| **Source:Target Ratio** | **Target Weighting** |
|---|---|
| 2:1 | 1.53x |
| 1:1 | 3.06x |
| 1:2 | 6.12x |

## 6.5.3 Results

All results come from performance on the same held-out 30-minute section of Kyrgyz audiobook. Decoding is performed with a bigram backoff language model trained on a Wikipedia Kyrgyz dump, and contains, 103,998 unigrams and 56,6871 bigrams. The bigram language model, lexicon, and main-task decision tree are built into a standard decoding graph (i.e. a Weighted Finite State Transducer) in the traditional Kaldi pipeline.

The experimental results are shown in Table (6.3) as percent Word Error Rate (WER) relative to the baseline model. All experiments show improvement over the baseline. Each column has in bold the model which performed best (the bottom row has also the bolded best average weighting).

Table 6.3: Reduction in Word Error Rates (WER) Relative to Single Task (STL) Baseline. Values in bold shown an improvement over the baseline.

| **Auxiliary (Source Lang) Tasks** | **Source:Target** *1-to-2* | *1-to-1* | **Weighting** *2-to-1* |
|---|---|---|---|
| STL Baseline | | 50.54 | |
| Monophones | **48.20** | **47.32** | **47.41** |
| Halfphones | **48.68** | **46.73** | **48.68** |
| Triphones | **49.37** | **47.12** | **46.73** |
| Monophones + Halfphones | **48.20** | **48.49** | **48.10** |
| Halfphones + Triphones | **50.05** | **48.00** | **47.90** |
| Monophones + Halfphones + Halfphones | **48.88** | **48.20** | **48.59** |

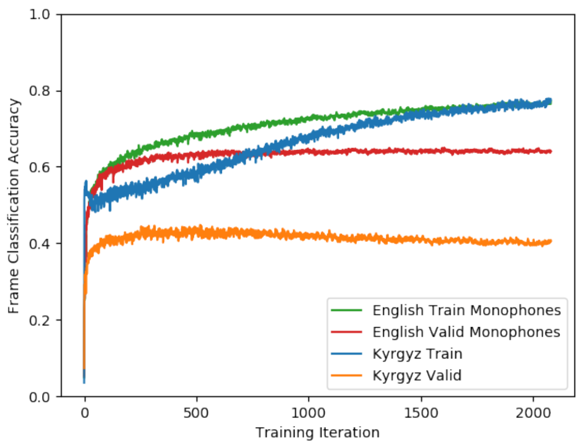
We see that on average, across all tasks and task combinations, a 1-to-1 weighting performed the best with an average 2.90% improvement over the baseline. Averaged over all weighting schemes, the best auxiliary task was English monophones (most abstract task).

The best overall combination of weighting and source task detail was a tie between (1) triphones + 2-to-1 weighting and (2) half-phones + 1-to-1 weighting. Even though on average the abstract source task labels (monophones) performed better, the more detailed source tasks achieved best WER in a single run (3.81% improvement).

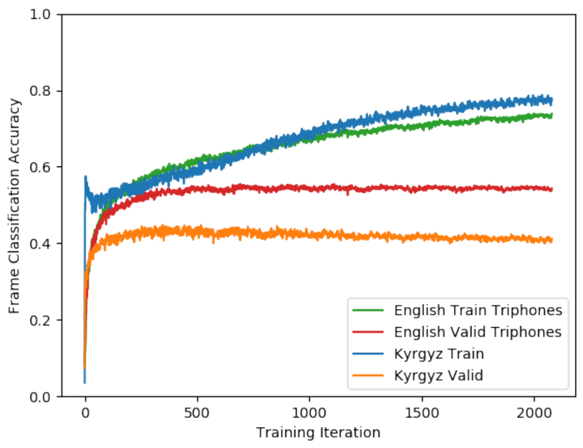
Comparing performance among combinations of auxiliary tasks, we see that MTL training with just one auxiliary task always performed better than two or three extra source tasks.

## 6.6 Discussion

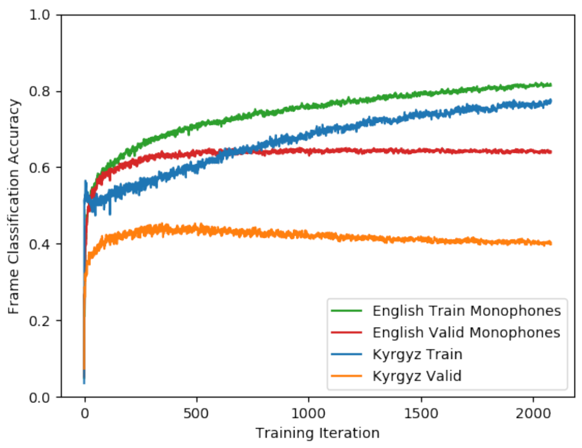
Perhaps the most interesting finding is the interaction between level of detail in individual source tasks and relative weighting during training. We find that in general, the less detail in the source task, the less weight we should give it during training. The monophones were the exception to the rule, getting best performance with 1-to-1 weighting. The following discussion will focus on this interaction (i.e. experiments represented in the first three columns of Table (6.3)).



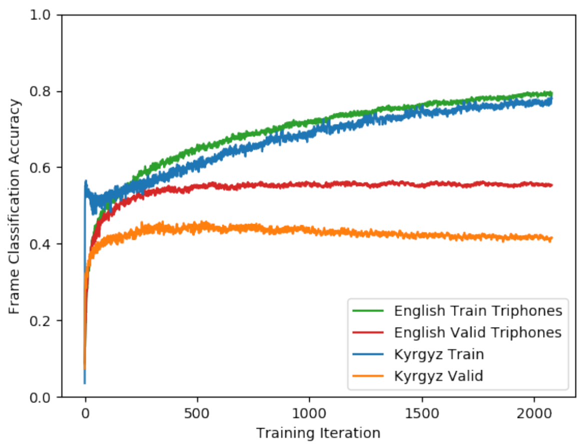
(a) 1-to-2 Weighting // Monophones



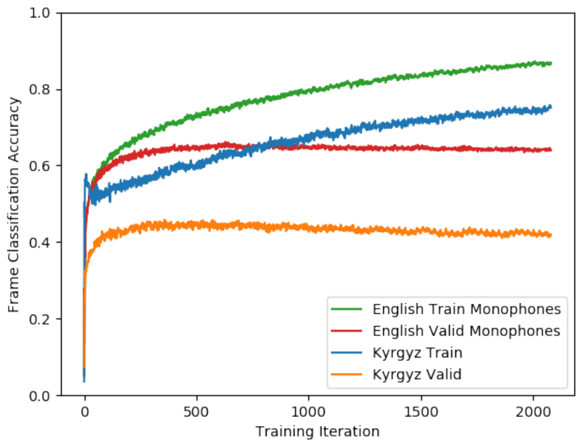
(b) 1-to-2 Weighting // Triphones



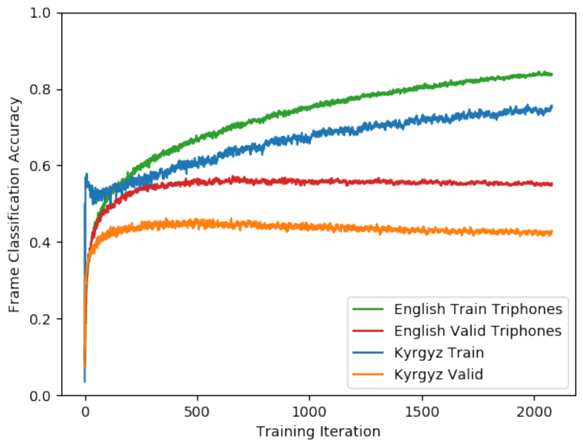
(c) 1-to-1 Weighting // Monophones



(d) 1-to-1 Weighting // Triphones



(e) 2-to-1 Weighting // Monophones



(f) 2-to-1 Weighting // Triphones

Figure 6-3: Source:Target Weighting vs. Source Task Detail

A task with fewer labels is typically easier to learn, finds a local minimum more quickly, and is less willing to budge once it is settled. A more difficult source task will take longer to learn, and as such, the target task will be able to exert more influence on the shared hidden layers. We see this behavior during training in Figure (6-3).

Figure (6-3) shows neural net performance during training (i.e. frame-level classification accuracy) on two separate auxiliary tasks (cf. Figure (6-3) columns) and three separate weighting schemes (cf. Figure (6-3) rows). The source task is weighted heavier in the last row, the target task is weighted more in the first row, and the source and target are weight equally in the center row. The left column shows experiments with English monophones as source task, and the right column shows English triphones as a source task. The Green vs Blue lines represent training data accuracy (for English vs. Kyrgyz respectively). The Red vs. Orange lines represent validation data accuracy (for. English vs. Kyrgyz respectively).

We see most overfitting to the target language data when the source task is least weighted and most detailed (cf. Figure (6-3d)). This model performed substantially worse than all others displayed here, with only a 1.17% WER improvement over baseline. Looking up to Figure (6-3c) we see the opposite extreme, where the source task is most weighted and least detailed. The gap between source and target tasks is widest in this setting, because the source task finds a good local minimum quickly, and the target data never has enough weight to influence the hidden layers. This is not the optimal weighting for this task (3.13% vs. 3.22% improvement).

These results indicate that merely adding related tasks in training via MTL is not enough to guarantee optimal transfer - one must consider task difficulty and weight accordingly.

Multi-Task Learning promises a very simple solution to a very hard problem, but it does not always deliver. It would seem that as long as we can add relevant tasks to our net, through the good graces of backprop, a best solution will automatically be discovered. It would also seem that we are guaranteed to get better results as long as we keep adding related tasks. This study shows that the picture is not so simple.

Starting with three additional tasks which are clearly related *a priori*, this study investigated each task's import and the dynamics of task combinations. Each task represents a level of abstraction from the typical training labels, from fine-grained (triphones), to more abstract (half-phones) to completely context-free (monophones). In addition to these three levels (deduced from a given decision tree), I tested logical combinations of abstractions: the entire tree, the top half of the tree, and the bottom half of the tree. None of these combinations outperformed the tasks added individually.

The interaction of task weighting and task detail is perhaps the most interesting finding presented here. With more labels, the task is inherently more difficult and the model has more parameters to train. As such, models with fewer labels found their local minimum more quickly, and were more likely to exert influence over shared hidden layers than the target task (i.e. from a smaller dataset). In similar MTL setups, care should be taken to weighting auxiliary tasks relative to their simplicity, even if tasks are related.

# Chapter 7

# Multi-Task Acoustic Modeling via Discovered Tasks

## 7.1 Contributions of this chapter

The current chapter investigates Acoustic Model training via Multi-Task Learning, where the auxiliary tasks are discovered in a lightly supervised setting. Specifically, I train a Multi-Task DNN Acoustic Model in the Hybrid framework, such that the Acoustic Model has two separate output layers which represent (1) traditional phonemes defined by a phonetic decision tree or (2) clusters of audio discovered by k-means clustering followed by a mapping to existing target labels. Given $\tilde{1}$.5 hours of audio, I observe a best-case 1.5% relative improvement in Word Error Rate. However, given the standard deviation of experimental runs, the improvement is not statistically significant from a paired t-test. Nevertheless, this line of research promises easily scalable improvement in Word Error Rate, and can be easily applied in a completely unsupervised setting, and as such warrants further exploration.

## 7.2 Introduction

The current study investigates auxiliary tasks which are not hand-crafted by a human, but automatically discovered from training data in two steps. First, unsupervised clustering (i.e. k-means clustering) is performed on the audio training data, then the existing triphone labels are mapped onto the discovered clusters. The second step ensures there will be some level of relatedness between the discovered clusters and the original labels. The cluster identities are then assigned as target labels for an additional task for Acoustic Model training via backpropagation.

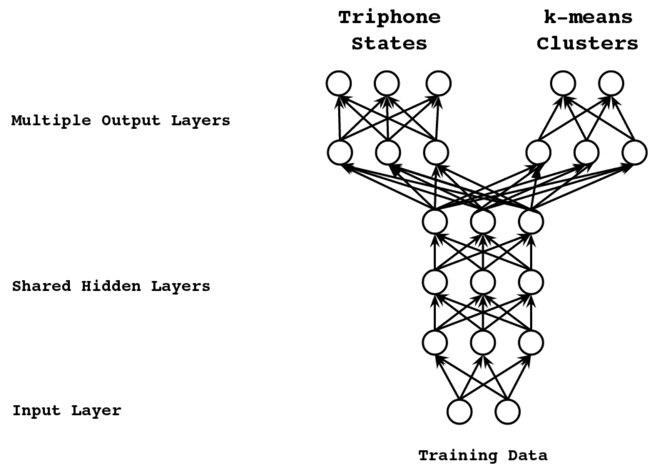


Figure 7-1: Multi-Task Acoustic Model Architecture investigated in the current study. Figure reproduced from [71].

## 7.3 Experiments

All Acoustic Models trained in this study are Time-Delay Neural Networks (TDNN), and are used in the standard DNN-HMM Hybrid approach for decoding. In general, the training procedure is standard in that the main task is trained on the alignments from a previously trained GMM-HMM model, where the alignment target is the state of a context-dependent triphone. Acoustic model training only differs from the standard pipeline in that the auxiliary task targets do not come directly from the phonetic decision tree, but rather from k-means clusters.

The intuition behind using k-means clustering is the following: unsupervised class discovery will surely lead to different classes compared to standard triphone GMM-HMM state alignment via the Baum-Welch algorithm. These k-means class assignments will be hopefully similar enough to the standard triphone labels that the clusters encode linguistically relevant categories. If the clusters are in fact linguistically relevant, then according to past work in MTL we should observe improved performance on the main task.

Instead of using raw k-means cluster labels for the auxiliary task, we first map all training examples which share a HMM triphone label to the same cluster. This ensures a certain degree of similarity between the main and auxiliary task. This effectively results in a merging of triphone labels, leading to a higher degree of linguistic abstraction.

### 7.3.1 Auxiliary Task Discovery

For both the main task and the auxiliary task, feature processing follows the exact same setup. Audio features are extracted as 13 dimensional PLP features with a 25ms Hamming window at a 10ms shift. The resulting feature vectors are then spliced to contain a context of 16 frames to the left and 12 frames to the right, and CMVN normalization is applied.

Given the extracted audio features, new labels for the auxiliary task were discovered in two main steps. First, we perform k-means clustering with a pre-set number of cluster centroids.* The number of cluster centroids is a hyperparameter, and in our system is not learned. All the experiments reported here were run using a number of cluster centroids that we found appropriate relative to the number of triphones discovered by a standard GMM-HMM setup on the same data set. This first step is completely unsupervised, and does yield us with target labels (i.e. cluster IDs) which can be used for classification as an auxiliary task. However, to ensure some level of relatedness between this discovered auxiliary task and the main task of triphone classification, we perform an additional step.

In this second step, we perform a mapping of k-means cluster IDs to triphone state IDs. Given all training examples aligned to a given triphone state, the most commonly assigned k-means centroid is chosen as new target label for those examples.

*TensorFlow k-means scripts: https://github.com/JRMeyer/kaldi-tf

As such, training examples aligned to the same triphone will share the same k-means cluster. After mapping, the resulting labels will represent groupings of data which correspond to either individual triphone states or clusters of triphone states. That is, the resulting labels will never represent a fraction of the data assigned to a triphone. If, for example, 100 training examples are aligned to the same triphone state in the original GMM-HMM alignment, then those 100 training examples will be assigned to the same cluster after mapping. It is possible that other training examples are also assigned to that same cluster, but the key is that those 100 training examples will always share the same new label after mapping.

This label-mapping procedure is illustrated below for one segment of audio. First, we cluster the audio with k-means (Figure (7-2)), and separately we perform alignment in the standard GMM-HMM framework (Figure (7-3)). Finally, we find the most common cluster ID for each triphone state, and re-assign that cluster ID as a new label, as shown in Figure (7-4).

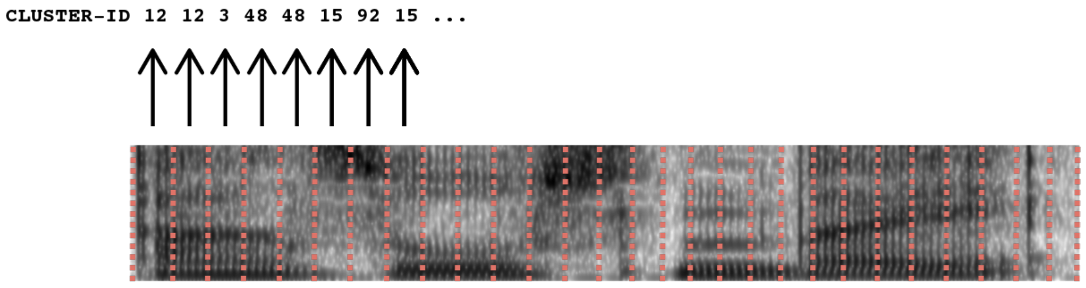


Figure 7-2: K-Means Discovered Clusters

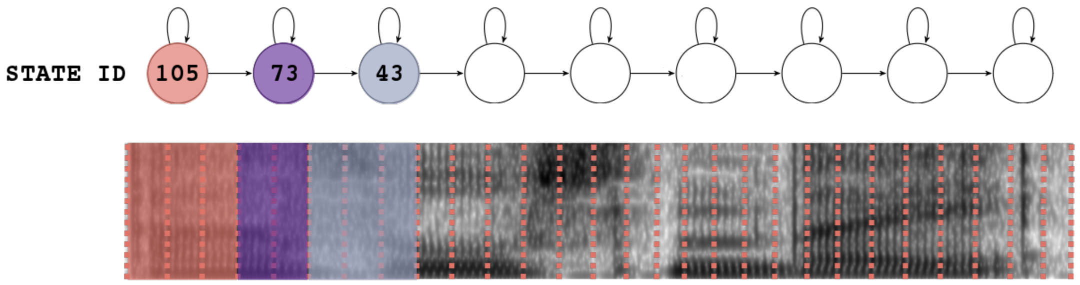


Figure 7-3: GMM-HMM aligned Triphone States

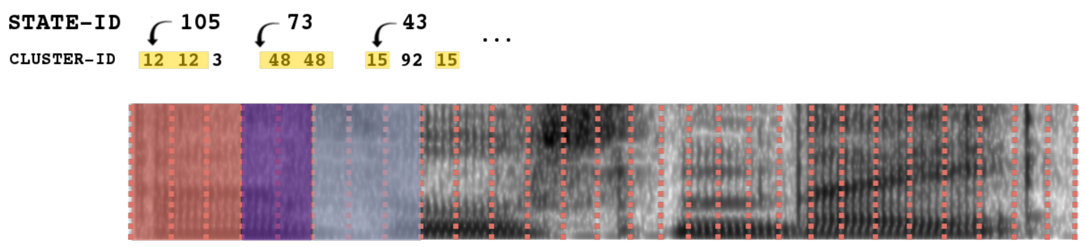


Figure 7-4: Voting & Mapping of Triphone-States → Clusters

### 7.3.2 Model Architecture

All experiments are trained and tested using the Hybrid DNN-HMM framework. Our experimental Acoustic Models are deep neural networks which have two separate output layers instead of just one (c.f. Figure (7-1)). The two output layers predict target classification labels for either (1) the main task (i.e. standard Kaldi triphone IDs) or (2) the auxiliary task (i.e. target labels which have been discovered via k-means clustering).

During GMM alignment, monophones were allotted 1,000 Gaussian components, and trained over 25 iterations of EM. These monophones were then expanded into context-dependent triphones via a phonetic decision tree, with a maximum of 2,000 leaves & 5,000 Gaussians. The resulting tied-state triphones are then trained over 25 iterations of EM.[†].

Final models are trained in Kaldi as `nnet3` Time-Delay Neural Networks (TDNNs) via a cross-entropy objective function [117, 113]. Given the alignments from the GMM-HMM models in addition to the discovered cluster assignments, an 11-layer, 532-dimensional TDNN with ReLU activation functions is trained over 5 epochs of backprop on a single GPU instance.

The main task (i.e. triphone state classification) and auxiliary task (i.e. cluster assignment classification) are implemented as separate output and penultimate layers. All other hidden layers of the TDNN are trained in parallel. A declining learning rate

[†]The GMM alignment script (along with all other experimental code) can be found on GitHub at https://github.com/JRMeyer/multi-task-kaldi

was used, with an initial learning rate of 0.0015 and a final learning rate of 0.00015.

### 7.3.3 Data

The Kyrgyz speech corpus used in the following experiments is the same as used in the previous chapters: an audiobook of a single female speaker of Kyrgyz. A total of 511 utterances of transcribed speech were used in training, and a held out 100 utterances were reserved for testing. Given that we used k-folds cross validation for (with k==5), the length of the train and test sets changed for each k, but the number of utterances was fixed.

## 7.4 Qualitative Analysis of Newly-Found Labels

These newly-discovered data clusters should be useful as an auxiliary classification task only in so far as they capture linguistically relevant groups of audio. If these new labels correspond to sounds such as phonemes, then that information should be useful to the main task.

Previously, researchers found these auxiliary target labels by either defining linguistic categories by hand or by using monophones from the phonetic decision tree. Training a model to identify these additional characteristics is thought to help because the model learns a set of underlying phonetic features which are useful for classifying the main targets.

These attempts at modeling abstract phonetic information all get at the same problem. A neural Acoustic Model trained on standard targets (e.g. triphones, characters) is trained to learn that each target is maximally different from every other target. In reality, we know that some of these targets are more similar than others, but the model does not know that.

Taking the example of triphone targets, the Acoustic Model does not have access to the knowledge that an `[a]` surrounded by `[t]`'s is the same vowel as an `[a]` surrounded by `[d]`'s. In fact, these two `[a]`'s are treated as if they belong to completely different classes. It is obvious to humans that two realizations of `[a]` are more similar to each other than an `[a]` is similar to an `[f]`. However, the output layer of the Acoustic Model does encode this knowledge. Discriminative training on triphone targets will loose the information that some triphones are more similar than others. One way to get that information back is to explicitly teach the model that two `[a]` triphones belong to the same abstract class. Given that all realizations of `[a]` share some phonetic features (e.g. presence of formant frequencies), a model which is forced to classify all `[a]`s together should be biased towards learning phonetic representations of the underlying commonalities. If the tasks are related, then these representations should be useful for completing other tasks.

As such, if our k-means clustering and subsequent mapping led to linguistically useful groups of the data, we should expect to see improvements in acoustic modeling. The following analysis is a glance into the contents of the clustering and mapping process for one experiment (i.e. one run from the five k-folds runs). In this study we ran three versions of k-means clustering, where $k = 256, 1024, 4096$. As we found out, for this dataset 1024 seemed to work the best, which is why we chose three settings grouped around 1024.

The results reported in the qualitative analysis, we took the entire training dataset, and first discovered 1024 clusters via k-means clustering. After the clustering phase, we took out data and performed standard GMM-HMM forced alignment, using a phoneme dictionary which was generated by some simple grapheme-to-phoneme rules.

Simultaneously, for the exact same training data fold, we generated triphone alignments (no feature or model adaptation was performed). The resulting phonetic decision tree yielded 672 leaves (i.e. triphone states).

At this point, we have 1024 clusters and 672 triphone states. As such, before we even perform our mapping of triphones onto clusters, we know that we will end up with *at most* 672 new labels. After clustering, alignment, and the final mapping procedure, we ended up with a total of 185 new labels.

These labels are equivalent to clusters of triphone states, and we should expect that the triphones which were grouped together are phonetically coherent. If they are not phonetically related, then there's no reason to expect useful bias will be learned from this auxiliary task. Out of the 185 newly-discovered labels, 75 of them corresponded to data from individual triphone states. That is, compared to the GMM-HMM model, these classes of labels were effectively copied over to the new task. Out of the 185, nine (9) of the new labels represented groupings of multiple triphones from a single monophone. This means that 4.8% of the new classes newly-found via the clustering and mapping procedure effectively re-discovered the phonetic decision tree without having access to it.

Moving on in the analysis, we decided to take a very course linguistic distinction (consonants vs. vowels), and see how many of our new classes corresponded to groupings of either only vowels or only consonants. We found that 39 new classes (i.e. 21.08%) were consistent in this way (c.f. Table (7.1) for a list of unique clusters). Furthermore, within these vowel-only or consonant-only clusters we found some logical groupings of similar phonemes, such as groups of nasal consonants or groups of back vowels.

In total, but only investigating the contents of these classes, and considering at most the linguistic distinction of vowel vs. consonant, we found that 84.21% (i.e. 123/185) of newly discovered classes had some linguistic justification.

| | Vowels | | Consonants | | |
|---|---|---|---|---|---|
| | a j | a u | k r | g n m | |
| BACK VOWELS → | a o | a ih | k p | s sh ch | |
| | e j | e ih | r ng | t k s p | |
| | e y | o u | d ch | m ng | ← NASALS |
| | u ih y | u ih | t k | t k h | |
| | i e y | o ih | d z | t k s | |
| | a e oe j ih | j ih | l z | t ch d | |
| | a ih o u y | | n p | t k zh b | |
| | | | | t g b s sh z zh | |

Table 7.1: Discovered Clusters of Triphone-States

## 7.5 Results

### 7.5.1 Hyperparameter Choice

#### Task-Weighting in Loss Function

The hyperparameters which we adjusted over our experiments are the following: (1) the number of clusters used in k-means clustering, and (2) a weight, $\alpha$, which controls the influence of each task during training via backpropagation. The settings for $\alpha$ were chosen based on common choices in the literature, which tend from $\alpha = 0.1$ to $\alpha = 0.3$.

Using the Cross Entropy loss function, we experimented with two different Loss weighting schemes as shown in Equations (7.1) and (7.2) . In the literature, we found that commonly Equation (7.1) is used, which down-weights the main and auxiliary tasks relative to each other, effectively normalizing their influence over training.

In addition to this standard Loss function, we experimented with a second Loss function, Equation (7.2), on the intuition that given such a small dataset, down-weighting the main task may non be effective.

$$\mathcal{L}_1 = (1 - \alpha) \cdot \mathcal{L}_{MAIN} + (\alpha) \cdot \mathcal{L}_{AUX} \tag{7.1}$$

$$\mathcal{L}_2 = \mathcal{L}_{MAIN} + (\alpha) \cdot \mathcal{L}_{AUX} \tag{7.2}$$

For each experimental run, a set of 511 utterances was used as the training set, and a held out set of 100 utterances were used for testing. There were five (5) experimental runs per hyperparameter setting.

#### Number of k-means Clusters

The number of clusters were chosen on a very simple principle: Given that our GMM-HMM alignments yielded on average between 600 and 700 triphone states, we chose a number of clusters which was nearly double that (i.e. 1024), and then expanded a range around this value. We chose this middle value of 1024 k-means clusters, and we experimented with a low of 256 clusters and a high of 4096 clusters.

### 7.5.2 Decoding

During decoding, *only* the main task is used. This highlights the purpose of the auxiliary task: to force the learning of robust representations during training; the auxiliary task serves as "training wheels" which are removed once the model is ready. The auxiliary task exists solely for the purpose of finding good representations for the main task during training.

Decoding is performed with a unigram language model trained on a Wikipedia Kyrgyz dump, and contains, 103,998 unigrams. The reason we chose to use a unigram model instead of a higher-order N-gram model is mostly due to limited access to computing systems. Nevertheless, from a smaller set of experiments on tri-gram systems, we observed the same general patterns.

The language model, lexicon, and main-task decision tree are built into a standard decoding graph (i.e. a Weighted Finite State Transducer) in the traditional Kaldi pipeline. The experimental results are shown in Table (7.2) and Table (7.3) as average percent Word Error Rate ($\pm$ standard deviation) over the five testing folds from our k-folds cross validation. The WER of the best performing model for each Loss function is displayed in bold in each table.

First, in Table (7.2) we present the results from the experiments using the standard relative Loss function in Equation (7.1). Second, in Table (7.3) we present the results from the experiments using the simplified Loss function in Equation (7.2).

With regards to the results in Table (7.2), we find small improvements in the average WER of four of the nine of the experiments. However, if we look to the

| | $\alpha = 0.1$ | $\alpha = 0.2$ | $\alpha = 0.3$ |
|---|---|---|---|
| Single Task Baseline | | 57.55 $\pm 1.82$ | |
| + **256 clusters** | 57.93 $\pm 1.63$ | 57.04 $\pm 1.58$ | 57.66 $\pm 1.24$ |
| + **1024 clusters** | 57.69 $\pm 3.78$ | 56.99 $\pm 3.08$ | 57.60 $\pm 0.79$ |
| + **4096 clusters** | 57.25 $\pm 2.87$ | 58.07 $\pm 1.35$ | 57.45 $\pm 0.32$ |

Table 7.2: WER% for Traditional Weighting Scheme. The results in each cell show average percent Word Error Rate (± standard deviation) over five testing folds from k-folds cross validation.

standard deviation of WERs, for any experiment which showed improvement, the standard deviation of that experiment was greater than the relative performance gains. We performed a paired t-test for each experimental run in comparison to the baseline, and none of the improvements were statistically significant.

At this point, we decided to modify the Loss function from the tied, relative task weighting in Equation (7.1) to the absolute down-weighting of the auxiliary task in Equation (7.2). The results from these former experiments are displayed in Table (7.3).

| | $\alpha = 0.1$ | $\alpha = 0.2$ | $\alpha = 0.3$ |
|---|---|---|---|
| Single Task Baseline | | 57.55 $\pm 1.82$ | |
| + **256 clusters** | 57.33 $\pm 2.49$ | 58.02 $\pm 2.09$ | 57.18 $\pm 0.56$ |
| + **1024 clusters** | 57.74 $\pm 3.06$ | 56.88 $\pm 1.33$ | 57.13 $\pm 1.55$ |
| + **4096 clusters** | 57.56 $\pm 2.53$ | 57.49 $\pm 3.17$ | 57.31 $\pm 1.31$ |

Table 7.3: WER% for Simple Weighting Scheme. The results in each cell show average percent Word Error Rate (± standard deviation) over five testing folds from k-folds cross validation.

The experimental results in Table (7.3) using the simpler Loss function in Equation (7.2) show a similar pattern to the original results. We do find a relative improvement with this new Loss function compared to the former, but nonetheless the standard deviations are too wide to show significance (i.e. from the paired t-test).

## 7.6 Discussion

The current study presents an initial step towards unsupervised task discovery for Multi-Task acoustic modeling. As more work in Automatic Speech Recognition moves towards End-to-End modeling (which does not use frame-level alignments) this line of work promising in that if we find truly unsupervised auxiliary tasks, they can be seamlessly integrated into End-to-End modeling.

Currently, End-to-End speech recognition systems require massive amounts of data to be trained, because they jointly learn each component of traditional models (i.e. pronunciation dictionary, language model, acoustic model). Traditional Acoustic Models also require less data because they make use of the triphone alignments from a previously trained GMM-HMM model. End-to-end models don't even have access to frame-level alignments. However, continuing this line of research to it's logical application to End-to-End systems, if we can find phoneme-like representations in the data in an unsupervised setting, then we can add frame-level targets or embedding targets for each audio frame.

For the current experiments, we worked in the traditional Hybrid DNN-HMM framework. The experiments here showed small average relative improvements, but the variation in performance was too great to give us statistical significance (as per a paired t-test).

Ideally we would not have the number of k-means clusters be a hyperparameter, but find some way to automatically discover the relevant number of classes. A number of promising approaches to unsupervised phoneme discovery can be found in the literature.[‡] It may be the case that we do not even need to find discrete clusters of phoneme categories, but rather, phonetic embeddings which encode information useful in discrimination.

[‡]Consult further the publications from the ZeroSpeech2015 and ZeroSpeech2017 challenges.

# Chapter 8

# End-to-end Cross-lingual Transfer Learning*

## 8.1 Introduction

In previous chapters of the dissertation, I investigate the beneficial effects of Multi-Task Learning on training Acoustic Models. This chapter takes a related, but different approach to the low-resource problem. Instead of assuming that the ASR practitioner has access to both a source and target dataset, in the following work on Transfer Learning, only a trained source model and target dataset are assumed.

In development scenarios Transfer Learning is often preferred over Multi-Task Learning because Transfer Learning typically requires fewer compute and fewer data resources (e.g. datasets and storage). For example, in lieu of using a 3,000+ hour dataset of English, one can make use of a model trained on that dataset. To train the original model from scratch, one would need approximately 2 days of wall-clock-time with 8 TitanX Pascal GPUs (each with 12GB of VRAM). The cost of just performing one training run will be expensive, but if the researcher factors in the reality of having to perform parameter-search over multiple training runs, the cost becomes prohibitive. By using a pre-trained highly performant model we don't have to incur those costs,

*This chapter is a modified version of a paper co-authored by Josh Meyer, Francis Tyers, Reuben Morais, and Kelly Davis.

because the original model-trainers already did.

Furthermore this chapter is an advancement from previous chapters in that it deals with End-to-End training. In previous chapters all experiments were set squarely within the hybrid, DNN-HMM framework. This chapter explores Connectionist Temporal Classification (CTC) networks which take audio as input and return characters as output, without any phonetic dictionary.

## 8.2 Road Map

This chapter investigates an End-to-End Transfer Learning approach for Automatic Speech Recognition which bypasses the need for linguistic resources or domain expertise. Certain layers are copied from a trained source model, new layers are initialized for a target language, the old and new layers are stitched together, and the new layers are trained via gradient descent. The effects of fine-tuning the original, copied layers are also investigated here.

Finally, the quality of the embedding spaces learned by each layer of the original DeepSpeech model are investigated. Specifically, this chapter presents results from linguistically motivated logistic regression tasks which are trained on top of feature-spaces at different model depths.

Results are presented for transfer-learning from English to the following twelve languages: German, French, Italian, Turkish, Catalan, Slovenian, Welsh, Irish, Breton, Tatar, Chuvash, and Kabyle.

The trained, source model used here is the v0.3.0 release of Mozilla's pre-trained English DeepSpeech[†]. This model is a unidirectional variation of Baidu's first DeepSpeech paper [65], trained via CTC loss on approximately 3,000 hours of English. The speech data for the 12 target languages was collected via Mozilla's Common Voice initiative.[‡]

End-to-end approaches for ASR are especially interesting for low-resource languages,

[†]`https://github.com/mozilla/DeepSpeech/releases/tag/v0.3.0`
[‡]`http://voice.mozilla.org`

because they do not require a pronunciation dictionary. However, all End-to-End training techniques (CTC [65, 2], Transducer [124, 9], Attention [22, 6], Convolutional [30, 119]) require thousands of hours of data to train, because the model must learn a mapping of audio directly to characters without any explicit intermediate linguistic representations. As such, End-to-End approaches are interesting for smaller datasets, but the typical training approach will not work. Transfer learning is an ideal solution for this problem because it is possible to leverage the knowledge from a source domain in order to bootstrap a model for the target task. Furthermore, the approach doesn't require access to a source domain dataset, but rather, a source domain model — which is often desirable due to copyright and privacy concerns.

## 8.3 Background

Speech recognition training techniques have been developed with high-resource languages in mind. The End-to-End approaches require upwards of 10,000 hours of transcribed audio [65], or even more with data-augmentation [2]. Nevertheless, low-resource languages started getting attention in the research community with IARPA's BABEL program [49], and various working approaches have been developed since then.

Previous low-resource work exists for End-to-End ASR [8, 147, 127, 31], as well as traditional DNN-HMM Hybrid ASR [32, 60, 84, 155, 52]. Typically these techniques involve leveraging data from as many languages as possible and then fine-tuning the model to one target language of interest. With regards to required resources, the End-to-End teacher-student approach in [31] is most similar to our approach in that it assumes only a trained source model and target language data. The algorithm in [52] is most similar to our copy-paste transfer approach. While these approaches may work, they typically assume that the developer has access to multilingual data and enough GPUs to make use of that data — in practice this is often not the case.

## 8.4 Overview of Transfer Learning

This paper investigates the effects of transfer learning for low-resource ASR.[§] The goal of transfer learning is to use source-domain knowledge as an inductive bias during parameter estimation for a target-domain task [107]. Source-domain knowledge can be found either *directly* in a source-domain dataset, or *indirectly* in a trained, source-domain model. While using a source dataset allows more fine-grained control of how bias is transferred (e.g. as in Multi-Task Learning), it is easier in practice to use a source model (given concerns about licensing, compute time, disk space, etc).

This current work is in many ways an ASR analog to the [167] work on transferability of ImageNet models and the work of [106] on transferability of the mid-level layers for image recognition tasks. The authors in [167] reached two major conclusions with regards to the transferability of features for image classification. Firstly, given the co-adaptation of layers in the source model, transferring layers isn't always as simple as deep vs. shallow. With regards to ImageNet, it was the case that multiple adjacent layers were co-adapted such that splitting them apart resulted in a degradation of performance on the target task. Secondly, higher level features may be too specific to the source domain to transfer well to the target task. Both of these findings are investigated with regards to End-to-End ASR by transferring pre-trained layers from a source domain (i.e. English) to a target task (i.e. target language).

Speech recognition and image classification are fundamentally different tasks. All spoken languages are produced by the human vocal tract, and as such they have features in common. However, the smallest unit of spoken language (i.e. the phoneme) is not language-independent, and not all acoustic features are contrastive in all languages. For End-to-End ASR models, it is unknown where language-independent representations are encoded. This study is an initial foray into this research direction. The embedding space of each layer of an End-to-End ASR model is investigated by training logistic regressions of linguistically-motivated tasks on top of each layer.

[§]For an overview of transfer learning in ASR, see [38].

## 8.5 Experimental Set-up

### 8.5.1 Model Architecture

All reported results were obtained with Mozilla's DeepSpeech toolkit[¶] — an open-source implementation of a variation of Baidu's first DeepSpeech paper [65]. This architecture is an End-to-End sequence-to-sequence model trained via stochastic gradient descent with a CTC loss function. The model is six layers deep: three fully connected layers followed by a unidirectional LSTM layer followed by two more fully connected layers (c.f. Figure 8-1). All hidden layers have a dimensionality of 2048 and a clipped ReLU activation. The output layer has as many dimensions as characters in the alphabet of the target language (including any desired punctuation as well as the blank symbol used for CTC). The input layer accepts a vector of 19 spliced frames (9 past frames + 1 present frame + 9 future frames) with 26 MFCC features each (i.e. a single, 494-dimensional vector).

### 8.5.2 Transfer Learning Procedure

The Transfer Learning technique in the current study is implemented as follows: a certain number of trained layers are copied from a pre-trained English DeepSpeech model, then a certain number of new layers are initialized and appended to the old layers. Finally, the newly initialized layers are updated via gradient descent according to the target language data. Optionally, the original layers are fine-tuned to the target language.

Newly appended layers are initialized via Xavier Initialization [54]. The number of appended layers and their architecture (i.e. Feed-Forward vs. LSTM) are chosen to be exact copies of the layers which were not copied from the original model. That is, all models investigated here are 6 layers deep: three fully connected layers followed by a unidirectional LSTM layer followed by two more fully connected layers. The Transfer Procedure is shown in Figure 8-2.

[¶]`https://github.com/mozilla/DeepSpeech`

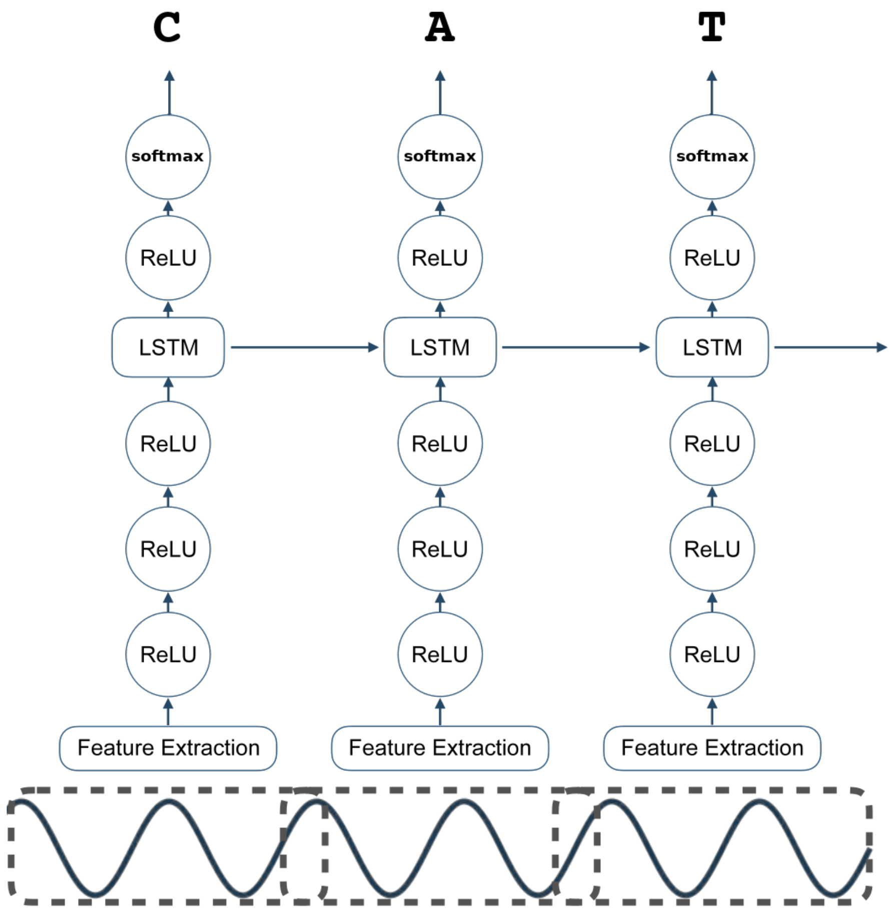


Figure 8-1: Architecture of Mozilla's DeepSpeech ASR system. A six-layer unidirectional CTC model, with one LSTM layer.

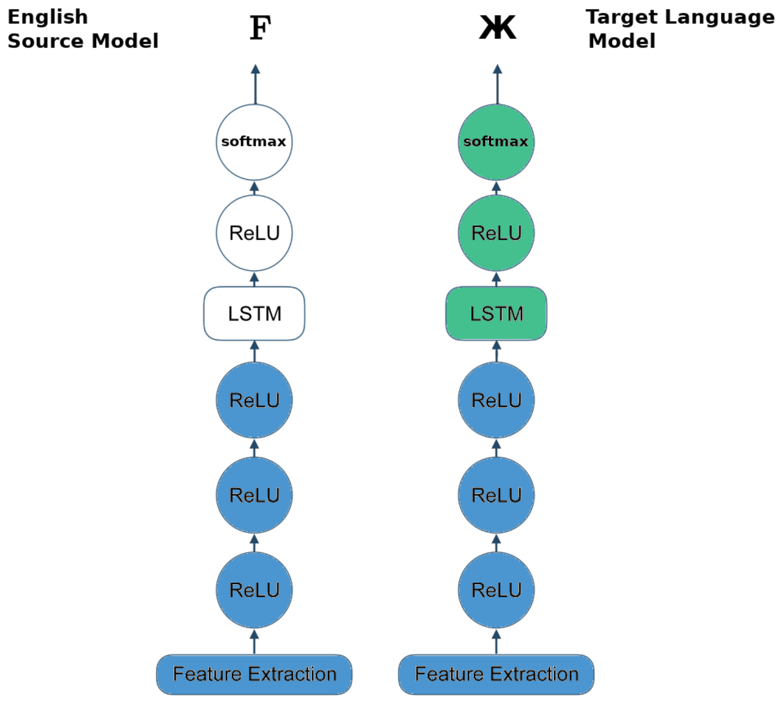


Figure 8-2: "Copy-Paste" Transfer Learning. In this example, a subset of model parameters trained on a language using the Latin alphabet is used to create a new model for a new language, written with the Cyrillic alphabet. Blue model parameters have been copied from the source model, and the green model parameters have been re-initialized from scratch.

### 8.5.3 Training Hyperparameters

All models were trained with the following hyperparameters on a single GPU. I use a batch-size of 24 for train and 48 for development, a dropout rate of 20%, and a learning rate of 0.0001 with the ADAM optimizer. The new, target-language layers were initialized via Xavier initialization [54]. Early stopping was determined when the validation loss had either (1) increased over a window of 5 validation iterations, or (2) the 5th loss in a window had not improved more than a mean loss threshold of 0.5 *and* the window of 5 validation losses showed a standard deviation of $< 0.5$.

### 8.5.4 Language Model

A language model is used in beam-search decoding, as outlined in [66]. For each language, a trigram backoff language model is trained from the raw text of Wikipedia using `kenlm` [70]. Singleton 2-grams and 3-grams were pruned to keep the compiled trie and binary ARPA files under 2G each.

### 8.5.5 Languages

The languages in our sample are typologically and genetically diverse. There are eight Indo-European languages, including three Romance languages (French, Italian, and Catalan), three Celtic languages (Welsh, Breton, and Irish), one Slavic language (Slovenian), and one Germanic language (German). In addition to this there are three Turkic languages (Turkish, Chuvash, and Tatar) and one Berber language (Kabyle). All of the languages with the exception of Tatar and Chuvash are written with the Latin alphabet. Both Tatar and Chuvash are written with the Cyrillic alphabet. Each language's writing system has a different number of characters, and all have characters which are not found in the English alphabet. All languages are written left to right.

In terms of morphological typology, they range from fusional (in the case of the Indo-European languages and Kabyle) to agglutinative (in the case of the Turkic languages). The languages also display a range of interesting morphophonological phenomena, such as initial consonant mutations in the Celtic languages — where the first consonant of a word changes depending on morphosyntactic context — and vowel harmony in the Turkic languages — where vowels in an affix agree with the last vowel in the stem. These phenomena are both kinds of long-range dependencies.

### 8.5.6 Data

All the data used in this study come from Mozilla's Common Voice initiative.[‖] The Common Voice data is crowd-sourced via a web app, where users read a visually presented sentence off their screen. The recording is then verified via a voting system

[‖]`http://voice.mozilla.org`

in which other users mark a {transcript, utterance} pair as being correct or incorrect. The text is particularly messy, containing digits (i.e. [0-9]) as well as punctuation. The absolute values of the results presented here are therefore lower than state-of-the-art for some languages, because this chapter concerns the relative effects of transfer learning, not SOTA itself. The text processing was identical across all languages to ensure as much comparability as possible. The audio itself is particularly noisy, often donated from smartphones in everyday noise environments. The language donation efforts are often spear-headed by a small group of individuals. As such, the early datasets are biased towards a small number of speakers, as can be seen in Table 8.1.

We made dataset splits (as can be seen in Table 8.1) such that one speaker's recordings are only present in one data split. This allows us to make a fair evaluation of speaker generalization, but as a result some training sets have very few speakers, making this an even more challenging scenario. The splits per language were made as close as possible to 80% train, 10% dev, and 10% test.

| | | **Dataset Size** | | | | | |
|---|---|---|---|---|---|---|---|
| | | Audio Clips | | | Unique Speakers | | |
| **Language** | **Code** | Dev | Test | Train | Dev | Test | Train |
| Slovenian | `sl` | 110 | 213 | 728 | 1 | 12 | 3 |
| Irish | `ga` | 181 | 138 | 1001 | 4 | 12 | 6 |
| Chuvash | `cv` | 96 | 77 | 1023 | 4 | 12 | 5 |
| Breton | `br` | 163 | 170 | 1079 | 3 | 15 | 7 |
| Turkish | `tr` | 407 | 374 | 3771 | 32 | 89 | 32 |
| Italian | `it` | 627 | 734 | 5019 | 29 | 136 | 37 |
| Welsh | `cy` | 1235 | 1201 | 9547 | 51 | 153 | 75 |
| Tatar | `tt` | 1811 | 1164 | 11187 | 9 | 64 | 3 |
| Catalan | `ca` | 5460 | 5037 | 38995 | 286 | 777 | 313 |
| French | `fr` | 5083 | 4835 | 40907 | 237 | 837 | 249 |
| Kabyle | `kab` | 5452 | 4643 | 43223 | 31 | 169 | 63 |
| German | `de` | 7982 | 7897 | 65745 | 247 | 1029 | 318 |

Table 8.1: Number of audio clips and unique speakers per language per dataset split.

Results from this dataset are particularly interesting in that (1) the text and audio are challenging, (2) the range of languages is wider than most any openly available speech corpus, and (3) the amount of data per language is from very small (less than training 1,000 clips for Slovenian) to relatively large (over 65,000 clips for German), as can be seen clearly in Figure 8-3.

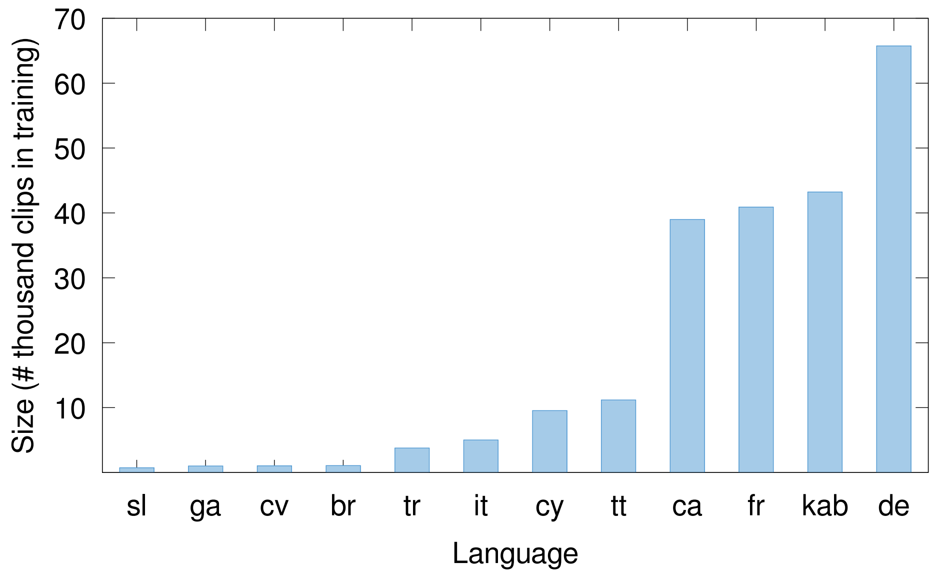


Figure 8-3: Number of audio clips per train split for each language. Audio clips for any individual speaker are only found in one split of the data (i.e. dev / test / train).

## 8.6 Results

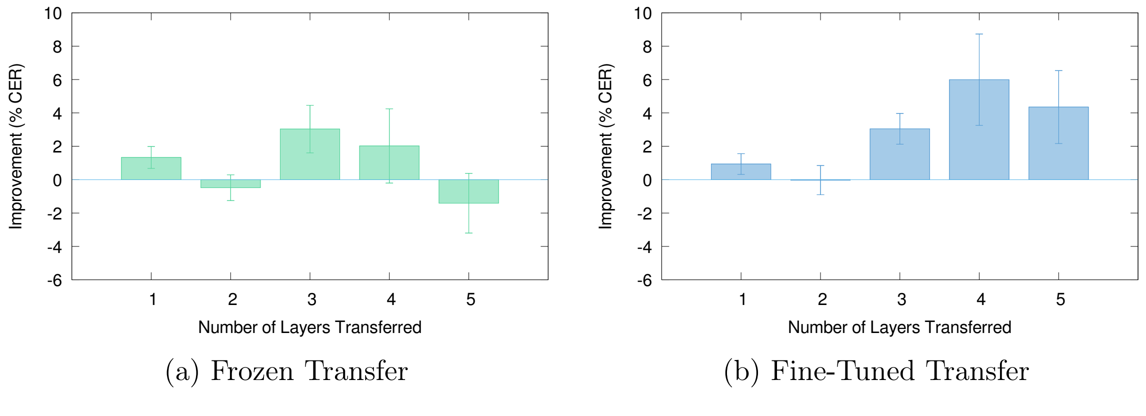


Figure 8-4: Mean and standard deviation for CER improvement for different # layers transferred from English.

The following results will be presented as follows: First, I discuss results from experiments in which the copied parameters *are not* updated. These experiments are refered to in prose, tables, and figures as *frozen* parameter transfer. Second, I discuss results from experiments in which the copied parameters *are* updated. I refer to these experiments as *fine-tuned* parameter transfer.

*A priori*, frozen transfer is interesting in that parameter estimation (per epoch) is faster than fine-tuning. Fewer calculations are required at every iteration of backpropagation with frozen transfer, because the gradient does not need to be calculated for the copied layers. From a model-interpretability point of view, frozen transfer offers us information as to what the different layers of the source model may be encoding. Results from frozen transfer will allow us to compare findings in this chapter to [167], investigating co-adaptation between layers as well as feature specificity to the source domain towards the output layer.

End-to-end speech recognition is by definition a mapping of audio directly onto characters. While in practice researchers often report results which incorporate some kind of language model during decoding, during parameter search all End-to-End models are trained to map audio directly onto characters. As such, the End-to-End model should learn to perform complicated hierarchical tasks which are handled by independently trained sub-modules in traditional speech recognition.

In traditional, "hybrid" speech recognition, various submodules are used to (1) identify context-dependent units (triphones) within continuous speech, (2) map those units onto higher-order sub-word parts (phonemes), and (3) map those sub-word parts onto written words in the target language. One might expect that End-to-End models are also capable of encoding these various higher-ordered representations, but I have very little idea as to where these representations are stored. Furthermore, some of these learned functions are more useful for multilingual transfer than others. If I can make an educated guess as to which layers of End-to-End models encode useful information, then transfer learning experiments may become much more efficient. Frozen transfer learning can give us a glimpse as to where these functions may have been learned in the source model.

Fine-tuned transfer learning is interesting in that it has been shown to perform out-perform frozen transfer on tasks like image recognition [167]. As such, while fine-tuning will usually perform better in terms of accuracy, frozen transfer takes fewer compute resources to train, and may lend us insight as to where the models has learned more or less transferable features.

### 8.6.1 Frozen Parameter Transfer

The following experiments were performed by "slicing" the original model at different depths. The first experiment serves as our absolute baseline, where the entire six-layer CTC model is trained from scratch without any transfer from English (c.f. column headed by 'None'). All transferred layers are contiguous, starting from the input layer. For example, the column with header '2' displays results where the first two layers (i.e. input layer + first hidden layer) are copied from English, and then four new layers are added on top. In all cases, the layers from English remain frozen during backprop, and new layers are updated with the gradient from the target language data. Early stopping was determined by a held-out validation set. The results from Frozen Transfer Learning can be found in Table 8.2.

With regards to improvement over the baseline model, for all of the twelve languages investigated here at least one transfer learning experiment outperformed the model trained from scratch. After this general finding, interpreting these results from a first glance is difficult, but if we dig in a little, some trends do emerge. The results in Table 8.2 only show the best result in bold, but if the improvements are averaged over the baseline for all languages, we find a clearer visualization of these results in Figure 8-4a. The bars show percent CER improvement averaged over all languages, and the tick marks display the standard deviation of this improvement.

Two of the five transfer scenarios (i.e. transfer of layer [1] or layers [1-3]) show reliable improvement over a baseline trained from scratch. Two other transfer scenarios tend to lead to interference (i.e. layers [1-2] and layers [1-5]). Transfer of the first four layers (crucially including the LSTM layer) can lead to either improvement or interference.

| | **Character Error Rate** | | | | | |
|---|---|---|---|---|---|---|
| | Number of Layers Copied from English | | | | | |
| **Lang.** | None | 1 | 2 | 3 | 4 | 5 |
| `sl` | 23.35 | 23.93 | 25.30 | 18.87 | **17.53** | 26.24 |
| `ga` | 31.83 | 29.08 | 36.14 | **27.22** | 29.07 | 32.27 |
| `cv` | 48.10 | 46.13 | 47.83 | 38.00 | **35.23** | 42.88 |
| `br` | 21.47 | 19.17 | 20.76 | 18.33 | **17.72** | 21.03 |
| `tr` | 34.66 | **32.98** | 35.47 | 33.00 | 33.66 | 36.71 |
| `it` | 40.91 | 39.20 | 41.55 | **38.16** | 39.40 | 43.21 |
| `cy` | 34.15 | 32.46 | 33.93 | **31.57** | 35.26 | 36.56 |
| `tt` | 32.61 | 29.20 | 30.52 | **27.37** | 28.28 | 31.28 |
| `ca` | 38.01 | **36.44** | 38.70 | 36.51 | 42.26 | 47.96 |
| `fr` | 43.33 | **43.30** | 43.47 | 43.37 | 43.75 | 43.79 |
| `kab` | 25.76 | 25.57 | 25.97 | **25.45** | 27.77 | 29.28 |
| `de` | 43.76 | 44.48 | 44.08 | 43.70 | 43.77 | **43.69** |

Table 8.2: Frozen Transfer Learning Character-error rates (CER) for each language, in addition to a baseline trained from scratch on the target language data. Bolded values display best model per language. Shading indicates relative performance per language, with darker indicating better models.

Similar to the findings of [167], transfer of the first and second layer fails most likely due to co-adaptation between the second and third layers. This conclusion is based on the fact that transfer of the proceeding layer (i.e. layer [1]), as well as transfer up to the following layer (i.e. layers [1-3]) both show improvement. If transfer at layer [2] failed due to over-specificity, then logically we should not observe transfer at layer [3] showing reliable improvement. Furthermore, on average transfer up to layer [5] of DeepSpeech shows interference. Interference of transfer at this layer is due to over-specificity to the source domain (i.e. English).

For languages with the largest training sets (Kabyle, French, Catalan, and German), the relative improvement from frozen transfer was smaller compared to languages with less data. The best-case improvement in CER for each language is presented (along with the results from fine-tuning experiments) in Figure 8-5.

### 8.6.2 Fine-Tuned Parameter Transfer

In the following experiments, the copied layers from English are updated via gradient descent (i.e. fine-tuned) according to the training data from the target language. Aside from this difference, the fine-tuned and frozen experiments followed the exact same training procedure. Results from all fine-tuned experiments are presented in Table 8.3.

The first result that stands out (in contrast to Table 8.2) is that for almost all of the twelve languages, the best transfer scenario is when the first four layers from a trained English model are copied. Upon closer inspection, we find that for the three languages which are the exception to this rule (Irish, French, and German), the difference in improvement between the best result and the fourth layer is small (i.e. less than half of a percent in CER).

Furthermore, looking at the averaged results (including standard deviation), we find that it is very much the case that the fourth layer leads to largest improvements on average (c.f. Figure 8-4b). As with the frozen parameter transfer experiments, even with fine-tuning the largest languages show smaller relative improvements compared to languages with less data (c.f. Figure 8-5). Frozen parameter transfer is typically

| | **Character Error Rate** | | | | | |
|---|---|---|---|---|---|---|
| | Number of Layers Copied from English | | | | | |
| **Lang.** | None | 1 | 2 | 3 | 4 | 5 |
| `sl` | 23.35 | 21.65 | 26.44 | 19.09 | **15.35** | 17.96 |
| `ga` | 31.83 | 31.01 | 32.2 | 27.5 | 25.42 | **24.98** |
| `cv` | 48.1 | 47.1 | 44.58 | 42.75 | **27.21** | 31.94 |
| `br` | 21.47 | 19.16 | 20.01 | 18.06 | **15.99** | 18.42 |
| `tr` | 34.66 | 34.12 | 34.83 | 31.79 | **27.55** | 29.74 |
| `it` | 40.91 | 42.65 | 42.82 | 36.89 | **33.63** | 35.10 |
| `cy` | 34.15 | 31.91 | 33.63 | 30.13 | **28.75** | 30.38 |
| `tt` | 32.61 | 31.43 | 30.80 | 27.79 | **26.42** | 28.63 |
| `ca` | 38.01 | 35.21 | 39.02 | 35.26 | **33.83** | 36.41 |
| `fr` | 43.33 | 43.26 | 43.51 | 43.24 | 43.20 | **43.19** |
| `kab` | 25.76 | 25.5 | 26.83 | 25.25 | **24.92** | 25.28 |
| `de` | 43.76 | 43.69 | 43.62 | **43.60** | 43.76 | 43.69 |

Table 8.3: Fine-Tuned Transfer Learning Character-error rates (CER) for each language, in addition to a baseline trained from scratch on the target language data. Bolded values display best model per language. Shading indicates relative performance per language, with darker indicating better models.

considered appropriate when the target dataset is very small, and as such fine-tuning is prone to overfit a model with a large number of parameters [167], but our experiments show that fine-tuning always leads to an improvement over frozen transfer when used in conjunction with early stopping.

Summarizing findings from the importance of model depth for transfer (c.f. Figure 8-4), we find that (1) transfer of any segment of the first three (fully connected) layers leads to near identical performance between frozen vs. fine-tuned, (2) transfer of the fourth (LSTM) layer leads to large stable gains when fine-tuned, but unreliable gains when frozen, and (3) transfer of the fifth (fully connected) layer leads to reliable gains with fine-tuning, and interference when frozen.

### 8.6.3 Importance of Data Size for Transfer

There is an observable correlation between the size of the training dataset and the effectiveness of transfer learning (c.f. Figure 8-5). Generally speaking, the more target data available, the less transfer learning helps. However, with fine-tuned transfer learning in particular, the performance from the largest datasets tends to the performance of the baseline. This is a very desirable feature, meaning that on average, transfer learning with fine-tuning will either improve or match the performance of a baseline, but performance should not drop significantly.

## 8.7 Model Interpretability: Investigating Embeddings

The Transfer Learning results presented in this chapter seem to indicate that the first three layers of the source model encode good features for language transfer. That is, these features transfer well from English to any of the 12 languages investigated. However, the above experiments do not get at the substance of what those three layers are encoding. Furthermore, the fourth (LSTM) layer transfers well with fine-tuning, suggesting that the layer has learned a combination of English-specific and language-generic representations. However, at this point there isn't enough evidence to verify these intuitions. Even for the fifth layer, we can only assume that frozen transfer fails

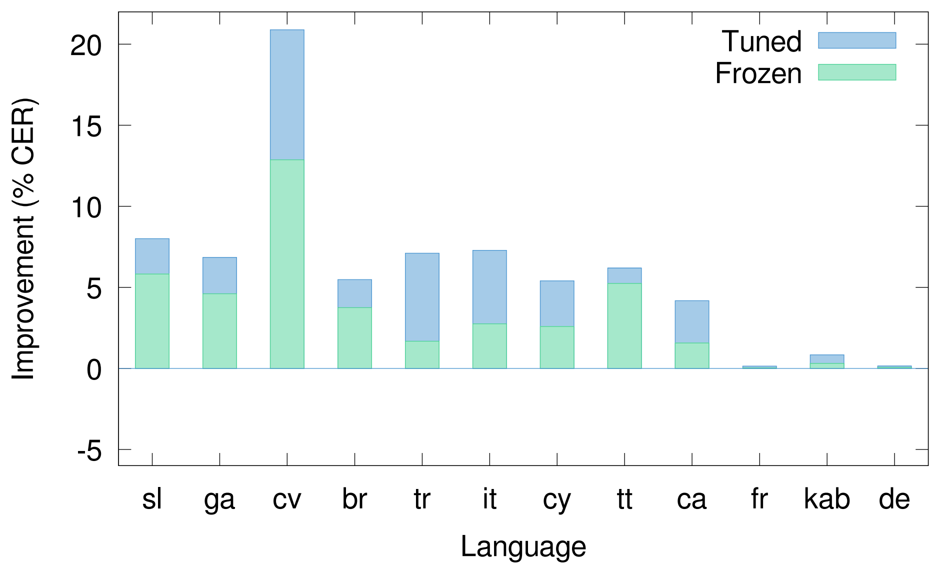


Figure 8-5: Largest improvement from Transfer Learning relative to the baseline, for each language. Languages are ordered from left → right in ascending order of size of training dataset. These improvements represent the bolded values in Table 8.2 and Table 8.3.

due to specificity to English. The follow-up experiments presented in this section aim to provide evidence for or against these intuitions.

For eleven out of twelve languages, frozen transfer of the first three layers shows improved Character Error Rate over a baseline (French being the exception with a 0.04 decrease in CER). This finding holds for even the larger languages (i.e. Catalan, Kabyle, German). These first three layers are purely feed-forward, and as such they should not be able to encode sequential higher-order information from English (e.g. spelling rules). This may be a clue as to why the first three layers transfer so well, regardless of the target language. Furthermore, when transferring these three layers, CER improvement is essentially identical between frozen or fine-tuned transfer (i.e. less than 0.017 CER difference between average improvement of fine-tuned vs. frozen transfer). This means that information from the target language data is adding nothing extra to the estimation of these layers.

We do find that the LSTM layer can be useful in frozen transfer, but not always. For seven of the eight smallest languages, frozen transfer of the LSTM layer showed improvement over the baseline (i.e. Slovenian, Irish Gaelic, Chuvash, Breton, Turkish, Italian, Tatar, but *not* Welsh). For the four largest languages, frozen transfer up to the LSTM layer either interferes or adds no real improvement over the baseline. This seems to indicate that the LSTM layer encodes some information useful for transfer (i.e. language-agnostic sequential information). Given that the fourth layer is the only recursive layer in Mozilla's variant of DeepSpeech, this layer must encode all sequential information, both language-agnostic and language-specific. An example of language-specific sequential information is spelling rules — they must be learned for every language. An important kind of language-generic sequential information is co-articulation — adjacent phonemes effect each other's acoustic signal based on the physiology of the human vocal tract. All kinds of sequential information are learned in this one layer.

Given these experimental observations and linguistic hypotheses on language-agnostic and language-specific features, I chose to evaluate the usefulness of embeddings at different layers on two new tasks: classification of audio as speech vs. non-speech, and classification of audio as English vs. German. If the features encoded at a certain layer are useful in performing these tasks, then the model has learned something about human speech which is generic (speech vs. noise) or language-specific (English vs. German).

As in all the experiments above, the v0.3.0 release of Mozilla's English DeepSpeech is used as a source model, but instead of appending more 2048x2048 hidden layers, a logistic regression is added to the final layer, and trained over three epochs with Cross-Entropy loss (all other hyperparameters are identical to above transfer experiments). This is shown in Figure 8-6 and with the loss function as defined in Equation 8.1. This approach is equivalent to using DeepSpeech as a feature extractor, and estimating parameters of a logistic regression over those features.

$$\mathcal{L} = Y \cdot -\log(P) + (1 - Y) \cdot -\log(1 - P) \tag{8.1}$$

$$P = \sigma((H_n \cdot W_p) + B_p) \tag{8.2}$$

where

$(P)$ = model prediction

$(Y)$ = ground-truth target label

$(H_n)$ = activations of model at layer $n$

$(W_p)$ = weights of regression layer

$(B_p)$ = biases of regression layer

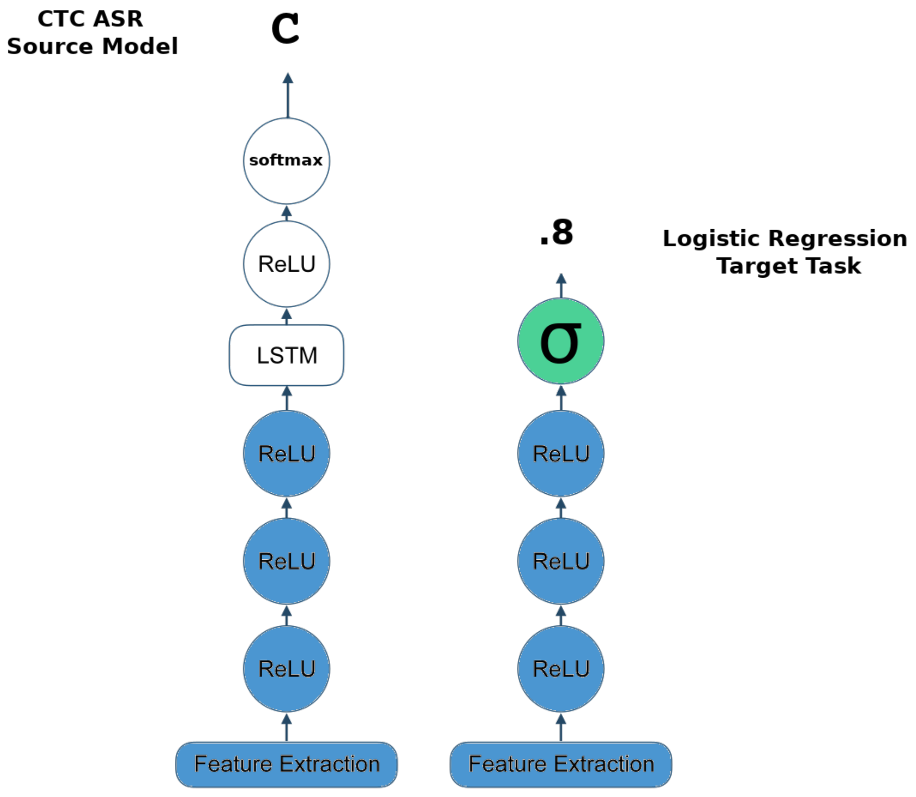


Figure 8-6: An example of Frozen Transfer from trained English CTC model to logistic regression tasks. In this example, three fully connected layers are transferred from the original model. Parameters shown in blue are copied and held frozen, parameters shown in green are estimated via Cross-Entropy Loss.

### 8.7.1 Speech vs. Noise Classification

For the first task, a regression is trained to distinguish urban background noise from audio samples of 13 Common Voice languages (i.e. the 12 mentioned languages plus English itself). Using multilingual data should reveal the degree to which these features are specific to English, or language-agnostic. Language-agnostic embeddings should perform this task better than tasks which are purely specific to English. In order to train the regression, a dataset of speech data is created by combining Common Voice samples for thirteen languages into a single "speech" dataset, and for non-speech data, a standard corpus of environmental city noises (UrbanSound8K [130]) is used.

For the speech data, the data is split into train and test sets of 5005, 442 audio clips respectively (i.e. 385, 34 clips per language). For the non-speech data, the train and test splits are made to be 5000, 435. The source DeepSpeech model is sliced at varying depths, keeping the source layers frozen, and then a single, fully-connected layer with a single output and logistic activations appended. Results are displayed in Table 8.4.

| **Classification Accuracy** | | | | | |
|---|---|---|---|---|---|
| Number of Layers Copied from English | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 51.01 | 93.68 | 92.82 | **95.30** | 94.55 | 93.53 |

Table 8.4: Speech vs. Non-Speech Audio Classification Accuracy (%) of a logistic regression trained on top of trained, frozen layers of an English CTC model. The bolded value shows the model with the highest accuracy. Shading corresponds to relative performance on classification, with darker being more accurate.

For the classification task of speech vs. non-speech, the first layer does not encode enough information to perform the task. A regression trained over embeddings from this first layer achieves an accuracy of 51.01% (i.e. essentially a coin-toss) on a held out test set. However, starting at layer [2], all depths contain enough information to classify the samples with $> 92\%$ accuracy. The best accuracy is seen when slicing the model at the LSTM (fourth) layer, likely due to being more effective at capturing long-term patterns, as each sample in the dataset is either all speech or all non-speech.

Furthermore, in the first task there is a slight drop in accuracy moving from the fourth layer to the fifth, and likewise a drop from the fifth to the sixth. This seems to indicate language-specificity (i.e. highly-tuned to English) at the output layers.

Both language-agnostic and language-specific encodings will be useful for speech vs. noise classification. However, we are also interested as to which layers encode purely language-agnostic information. To investigate this, we can make use of a language identification task.

## 8.7.2 Language Identification

The following section presents results from a language identification task trained on the embeddings of the various layers of a trained English model. Specifically, a logisitic regression is trained to classify audio as containing English or German language. The training dataset is created by taking an equal number of English and German samples from Common Voice, and then training a regression build on embeddings from varying model depths, following the same procedure detailed above.

By setting one of the two languages to English, and by extracting embedded features from a trained End-to-End English CTC as a source model, we can identify which embedding spaces are English-specific. If the layers do not encode English-specific information, then a logistic regression built on top of those features will perform this task poorly. If the layer activates differently for English than for another language, then we may conclude that the layer in question has encoded English-specific information.

For the other language, I chose German, given that it is the most closely related to English of the Common Voice languages. As such, distinguishing English and German should be a relatively harder task than distinguishing English from any of the other languages. Train and test datasets were made of {5000, 500} audio clips for each language (German and English). The classifier was added on top of the DeepSpeech layers and trained over three epochs with gradient descent from a Cross-Entropy loss function. The results are presented in Table 8.5.

We find that the features of the top three layers are better than the bottom three

| **Classification Accuracy** | | | | | |
|---|---|---|---|---|---|
| Number of Layers Copied from English | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 66.51 | 66.38 | 52.77 | **86.21** | 74.97 | 85.00 |

Table 8.5: English vs. German Audio Classification Accuracy (%) of a logistic regression trained on top of trained, frozen layers of an English CTC model. The bolded value shows the model with the highest accuracy. Shading corresponds to relative performance on classification, with darker being more accurate.

layers on this task. When slicing at the LSTM layer, there's a significant improvement in performance with an 86.21% classification accuracy. This finding means that the model is indeed more language-specific (i.e. English-specific) at the top layers. A particular interesting finding from Table 8.5 is that feature embeddings from layers [1-3] are essentially language-agnostic. These features are useless for distinguishing English and German. However, referencing the results from classification of speech and background noise (c.f. Table 8.4), we know that embeddings at layer [3] do encode information about human speech.

As such, the features learned at layer [3] are specific to language, but not to any one language (in this case, English). This confirms the stated intuition, given that we observed consistent improvements when transfering these features in an ASR task (c.f. layer [3] in Figure 8-4a and Figure 8-4b).

## 8.8 Discussion

In this chapter I present results from End-to-End ASR transfer learning experiments using a trained English CTC model as a source model and 12 other languages as target languages. I observe a stable, cross-lingual tendency that the first three layers of the trained model are crucial for successful transfer. In follow-up interpretability experiments, we find the first three layers robustly encode the difference between human speech and background noise, but they do not encode information useful for distinguishing languages. In this sense, these first three layers encode language-general information. Furthermore, transfer learning with fine-tuning can exploit and adapt useful bias from a fourth, LSTM layer, which is otherwise too specific to English. Embeddings from this LSTM layer are both useful for speech vs. non-speech classification as well as language discrimination.

For low-resource languages, transfer learning promises quick bootstrapping, avoiding the need of linguistic resources. Furthermore, supporting experiments which investigate the interpretability of End-to-End ASR models are crucial for advancing knowledge on how transfer learning works, and how it can be better used in speech recognition.

### 8.8.1 Relation to ULMFiT

Recent developments in NLP directly relate to the findings in this chapter. Specifically, the work on ULMFiT [73] can be seen as a similar investigation of Transfer in NLP, from a source task (i.e. General English language modeling) to a target task (i.e. text classification in a certain domain). The work in ULMFiT introduces an approach based on three key methods: (1) discriminative fine-tuning, (2) triangular learning rates, and (3) gradual unfreezing of source layers.

The gradual unfreezing of source layers is a very intuitive approach, and closely related to the work in this chapter. That is, the work in this chapter shows that varying language-agnostic and language-specific representations have been learnt in the End-to-End ASR model, and given this knowledge, a gradual unfreezing approach as proposed in ULMFiT would be a very exciting avenue for future work. The authors in [73] do not perform any experiments on model interpretability, but they base their unfreezing schedule on the basic knowledge that the original model encodes domain general information towards the input layer, and domain-/task-specific information towards the output layer. Combining information learned here about DeepSpeech's level of linguistic representations, a more informed unfreezing schedule may be posed.

Nevertheless, it is difficult to say how the results from ULMFiT can be translated into the cross-lingual ASR transfer domain. The former work only deals with English, and as such, the transfer is between tasks and domains, not languages. It may very well be the case that the approach suggested from ULMFiT will need significant modifications to work in the cross-lingual ASR transfer.

### 8.8.2 Relevance to the Linguist

Focusing on the follow-up interpretability experiments, the results have some direct interest to the linguistic community. The first three layers robustly encode the difference between human speech and background noise, but they do not encode information useful for distinguishing languages. In this sense, these first three layers encode language-general information. Linguistically, there's a lot to un-pack here. Focusing on just the embeddings of the model's third layer, we have three main findings:

1. Layer 3 embeddings transfer well to new languages (with and without fine-tuning)
2. Layer 3 embeddings encode differences between human speech and background noise
3. Layer 3 embeddings are not useful for language discrimination (English vs. German)

Furthermore, there is a very relevant piece of information about Layer 3's architecture:

1. Layer 3 is unable to encode any sequenctial information in the speech signal (it is a feed-forward layer, as well as all preceding layers)

As such, we know that Layer 3 is performing frame-level (not sequential) operations which are language-independent (good for ASR transfer, bad for language discrimination), and human speech-specific (good for speech vs. noise discrimination). The linguist will have many questions about these embeddings. Are these embeddings a sort of universal phoneme-extractor? Are these embeddings purely a speech vs. noise separator, or is there some kind of transformation being applied to the speech signal (e.g. something akin to allophone-to-phoneme conversion)? Given the results from these follow-up studies, we konw that there is some processing on the audio signal which is special to speech, but we don't yet know how low-level or high-level that processing is. It will be of interest to the psycholinguist to know how much this signal processing corresponds to the processing done by the human ear and brain. There's no *a-priori* reason to assume the human and neural model perform the same kind of processing, but in either case, the more we know about how the Acoustic Model performs signal processing, the better we can perform model transfer to new datasets from low-resourced languages.

# Chapter 9

# Conclusions

## 9.1 The Problem

The goal of this dissertation is to address the following question: *How can we best exploit small datasets to train Acoustic Models for Automatic Speech Recognition via Multi-Task Learning and Transfer Learning?* Among other reasons, I chose to work with the Multi-Task Learning and Transfer Learning because they do not require the collection of more in-domain data. As such, the methods investigated here may be applied to any dataset for speech recognition.

## 9.2 The Approaches Taken in this Dissertation

Firstly, in Chapter (5) I investigate Multi-Task methods which build off linguistic knowledge. The methods developed here may be used to train an Acoustic Model for any language. Secondly, in Chapter (6) I investigate Multi-Task methods which exploit a natural by-product of the traditional ASR pipeline — the phonetic decision tree. Thirdly, in Chapter (7) I use an unsupervised learning technique well-known in general machine learning — k-means clustering. I demonstrate that it is possible to generate new label-spaces which exert useful inductive bias during Multi-Task Acoustic Model training. Finally, in Chapter (8) using the well-known "copy-paste" Transfer Learning approach, I demonstrate that with an appropriate early-stopping criteria, cross-lingual transfer is possible to both large and small target datasets, even when the source and target language are sampled from completely different language families.

## 9.3 Implications and Future Directions

### 9.3.1 Multi-Task Experiments

In Chapter (5) we found that by collapsing the phoneme-set of a language along basic linguistic dimensions, we were able to create new labels which allow us to boost performance in acoustic modeling in a Multi-Task framework. We replicated this effect for both cross-lingual transfer and speaker transfer. Furthermore, we found that the weighting scheme of source-to-target tasks is crucial in achieving best performance. With regards to future directions, we would like to extend this approach to more languages and language families (not just English and Kyrgyz as presented here). Furthermore, another important direction is replicating these results on larger corpora. As shown in [1], certain Multi-Task results only hold for smaller datasets. The major bottleneck in the Chapter (5) methodology is that linguistic knowledge is required to create the new labels, which leads us to the approach in Chapter (6).

In Chapter (6), we found that it is indeed possible to create useful auxiliary tasks for cross-lingual Multi-Task acoustic modeling by merely slicing the phonetic decision tree an various depths. Given an English dataset as a source task and a Kyrgyz dataset as a target task, the English auxiliary labels were varied by keeping them as monophones, as triphones, or something in between. We find an interesting interaction between the depth of the phonetic decision tree and the weighting of labels during backprop on Word Error Rate in the main task. In general, the less phonetic detail in the source task, the less weight we should give it during training to achieve best results. As evidenced by the loss curves during training, a task with fewer labels (i.e. less phonetic detail) finds a local minimum more quickly than a task with more labels. A more difficult source task (i.e. more labels) takes longer to learn, and as such, the target task is not able to influence the estimation of the shared hidden layers. These results indicate that the addition of related tasks in Multi-Task acoustic modeling is not sufficient to guarantee successful transfer — the scientist must consider the relative difficulty of tasks and then weight the gradients accordingly.

In Chapter (7) we train a Multi-Task Acoustic Model, such that the model has two separate output layers which encode (1) traditional phonemes defined by a phonetic decision tree or (2) clusters of audio discovered by k-means clustering followed by a mapping to existing target labels. Given 1.5 hours of audio, we observed a 1.5% relative improvement in Word Error Rate, however, given the standard deviation of experimental runs the improvement is not statistically significant from a paired t-test. Nevertheless, this line of research promises easily scalable improvement in WER, and can be easily applied in a completely unsupervised setting. Future directions for this work include exploring unsupervised clustering methods which are appropriate for audio data. k-means clustering performs worse in highly-dimensional spaces, where the dimensions are themselves noisy. Some feature extraction and compression (e.g. LDA) should be performed before clustering methods are applied.

### 9.3.2 Transfer Experiments

In Chapter (8), we find that Cross-lingual Transfer Learning from a pre-trained English CTC model to any of the 12 twelve languages tested leads to improvement over a baseline trained only on the target language. Summarizing the main findings from this Chapter, (c.f. Figure 8-4), we find that:

1. Transfer of any segment of the first three (fully connected) layers leads to near identical performance between frozen vs. fine-tuned.

2. Transfer of the fourth (LSTM) layer leads to large stable gains when fine-tuned, but unreliable gains when frozen.

3. Transfer of the fifth (fully connected) layer leads to reliable gains with fine-tuning, and interference when frozen.

These results, taken together with the interpretability findings, lead us to the conclusion that the first half of DeepSpeech (i.e. from input layer to layer 3) encodes stable, language-independent but human-language-specific features. Also, all recurrent information is stored in an LSTM layer, which can be effectively adapted via fine-tuning.

Furthermore, we found a correlation between the size of the training dataset and the effectiveness of transfer learning (c.f. Figure 8-5). Generally speaking, the more target data available — the less transfer learning helps. However, with fine-tuned transfer learning in particular, the performance from the largest datasets tends to the performance of the baseline. This is a very desirable feature: transfer learning with fine-tuning will on average improve or match the performance of a baseline.

### 9.3.3 Model Interpretability Experiments

With regards to the model interpretability experiments in Chapter (8), the results have direct interest to the linguistic community. We find that the first three layers of an English End-to-End speech recognition model robustly encode the difference between human speech and background noise. However, these same encodings do not capture information useful for distinguishing languages. In this sense, these first three layers encode language-general information while remaining language-agnostic. Focusing on just these embeddings at the model's third layer, we have three main findings:

1. Layer 3 embeddings transfer well to new languages (with and without fine-tuning)
2. Layer 3 embeddings encode differences between human speech and background noise
3. Layer 3 embeddings are not useful for language discrimination (English vs. German)

Furthermore, there is a very relevant piece of information about Layer 3's architecture:

1. Layer 3 is unable to encode any sequential information from the speech signal. This is due to the fact that Layer 3 is a feed-forward layer, and so are all preceding layers.

As such, we know that Layer 3 is performing frame-level (not sequential) operations which are language-independent (good for ASR transfer, bad for language discrimination). The linguist will have many questions about these embeddings. Are these embeddings a sort of universal phoneme-extractor? Are these embeddings a speech vs. noise separator? Is there any higher-level transformation being applied to the speech signal (e.g. allophone-to-phoneme conversion)? Given the results from these follow-up studies, we know that there is some processing on the audio signal which is special to speech, but we don't yet know how low-level or high-level that processing is. While not within the scope of this dissertation, these specific questions may be addressed via comparative visualizations of the activations at Layer 3. For example, a promising future direction is be the comparison of model activations for pure background noise vs. pure human speech. If these layers are performing a signal separation of speech from background noise, we expect the model activations at these layers to be significantly lower for the background noise audio segments. Such an experiment is very straightforward given the resources presented in Chapter (8), namely: a trained release of Mozilla's DeepSpeech, the UrbanSound dataset, and the Common Voice dataset.

It will be of interest to the psycholinguist to know how much this signal processing corresponds to the processing done by the human ear and brain. There is no *a priori* reason to assume humans and neural models perform the same kind of audio processing. Nevertheless, the better we understand how the Acoustic Model performs signal processing, the better we can perform model transfer to new datasets.

## 9.4 Summary

This dissertation presents methods for exploitation of a source domain in order to perform speech recognition in a target domain. The target domain is low-resourced in that we only assume access to a limited dataset. The source domain is instantiated as either a dataset or a pre-trained model. Useful information exists in these source domains which can be used as inductive bias for learning in a target domain. Multi-Task Learning and Transfer Learning are leveraged to perform this domain exploitation.

With regards to Multi-Task Learning, I show that new source domains may be created (and subsequently exploited) using linguistic knowledge, a by-product in the traditional ASR pipeline, or an unsupervised clustering algorithm. These approaches are especially relevant to low-resource speech recognition in that the ASR practitioner can apply any of these methods to any speech dataset in any language. The progression of methods based on linguistic knowledge (Chapter (5)) to the ASR pipeline (Chapter (6)) to unsupervised clustering (Chapter (7)) corresponds to a diminishing of dependencies. The two dependencies in questions are (1) linguistic knowledge and (2) the traditional ASR pipeline. As the field of ASR moves from traditional (i.e. Hybrid) models to End-to-End models, methods which do not assume these dependencies will become more and more relevant. Of these three chapters on Multi-Task Learning, the final approach in Chapter (7) lends itself best to End-to-End speech recognition. Regardless of the framework in which we train an Acoustic Model, methods are more useful, impactful, and replicable if they assume fewer dependencies.

Lastly, with regards to the "copy-paste" Transfer Learning approach explored in Chapter (8), we have shown that robust cross-lingual transfer in End-to-End ASR is possible. This is (to date) the most cross-lingual transfer study in End-to-End speech recognition using openly licensed data (i.e. the Common Voice dataset), and as such these findings are of great relevance to the speech recognition community. Furthermore, this chapter includes particularly thought-provoking findings on the embedding spaces of a well-trained End-to-End speech recognition model. These findings suggest future research directions on the interpretability of neural models which operate over human speech signals.

Using any of the methods explored in this dissertation, an ASR practitioner should be able to improve speech recognition performance on a low-resourced language.

# Bibliography

[1] Yossi Adi, Neil Zeghidour, Ronan Collobert, Nicolas Usunier, Vitaliy Liptchinsky, and Gabriel Synnaeve. To reverse the gradient or not: An empirical comparison of adversarial and multi-task learning in speech recognition. *arXiv preprint arXiv:1812.03483*, 2018.

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep Speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[3] Antonios Anastasopoulos and David Chiang. Tied multitask learning for neural speech translation. *arXiv preprint arXiv:1802.06655*, 2018.

[4] Vipul Arora, Aditi Lahiri, and Henning Reetz. Phonological feature based mispronunciation detection and diagnosis using multi-task dnns and active learning. 2017.

[5] Leonardo Badino. Phonetic context embeddings for dnn-hmm phone recognition. In *INTERSPEECH*, pages 405–409, 2016.

[6] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

[7] James K Baker. Stochastic modeling as a means of automatic speech recognition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1975.

[8] Vladimir Bataev, Maxim Korenevsky, Ivan Medennikov, and Alexander Zatvornitskiy. Exploring end-to-end techniques for low-resource speech recognition. In *International Conference on Speech and Computer*, pages 32–41. Springer, 2018.

[9] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu. Exploring neural transducers for end-to-end speech recognition. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 206–213. IEEE, 2017.

[10] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[11] Peter Bell and Steve Renals. Complementary tasks for context-dependent deep neural network acoustic models. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[12] Peter Bell and Steve Renals. Regularization of context-dependent deep neural networks with context-independent multi-task training. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4290–4294. IEEE, 2015.

[13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[15] Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100, 2014.

[16] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[17] Peter F Brown. The acoustic-modeling problem in automatic speech recognition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1987.

[18] Peter F Brown. Speech recognition method including biased principal components, January 5 1988. US Patent 4,718,093.

[19] Rich Caruana. Algorithms and applications for multitask learning. In *ICML*, pages 87–95, 1996.

[20] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[21] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

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

[23] Dongpeng Chen. *Multi-task Learning Deep Neural Networks for Automatic Speech Recognition*. PhD thesis, Hong Kong University of Science and Technology, 2015.

[24] Dongpeng Chen, Brian Mak, Cheung-Chi Leung, and Sunil Sivadas. Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5592–5596. IEEE, 2014.

[25] Dongpeng Chen and Brian Kan-Wing Mak. Multitask learning of deep neural networks for low-resource speech recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(7):1172–1183, July 2015.

[26] Nanxin Chen, Yanmin Qian, and Kai Yu. Multi-task learning for text-dependent speaker verification. In *Sixteenth annual conference of the international speech communication association*, 2015.

[27] Zhuo Chen, Shinji Watanabe, Hakan Erdogan, and John R Hershey. Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[28] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*, 2014.

[29] Jordan Cohen, Terri Kamm, and Andreas G Andreou. Vocal tract normalization in speech recognition: Compensating for systematic speaker variability. *The Journal of the Acoustical Society of America*, 97(5):3246–3247, 1995.

[30] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[31] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, Tom Sercu, Kartik Audhkhasi, Abhinav Sethy, Markus Nussbaum-Thom, and Andrew Rosenberg. Knowledge distillation across ensembles of multilingual models for low-resource languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4825–4829. IEEE, 2017.

[32] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, Abhinav Sethy, Kartik Audhkhasi, Xiaodong Cui, Ellen Kislal, Lidia Mangu, Markus Nussbaum-Thom, Michael Picheny, et al. Multilingual representations for low resource speech recognition and keyword search. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 259–266. IEEE, 2015.

[33] Siddharth Dalmia, Ramon Sanabria, Florian Metze, and Alan W Black. Sequence-based multi-lingual low resource speech recognition. *arXiv preprint arXiv:1802.07420*, 2018.

[34] Amit Das, Mark Hasegawa-Johnson, and Karel Veselỳ. Deep auto-encoder based multi-task learning using probabilistic transcriptions. *Proc. Interspeech 2017*, pages 2073–2077, 2017.

[35] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[36] KH Davis, R Biddulph, and Stephen Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, 1952.

[37] Angel De La Torre, Antonio M Peinado, José C Segura, José L Pérez-Córdoba, M Carmen Benítez, and Antonio J Rubio. Histogram equalization of speech representation for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(3):355–366, 2005.

[38] Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

[39] Vassilios V Digalakis, Dimitry Rtischev, and Leonardo G Neumeyer. Speaker adaptation using constrained estimation of gaussian mixtures. *IEEE Transactions on speech and Audio Processing*, 3(5):357–366, 1995.

[40] Wenhao Ding and Liang He. Mtgan: Speaker verification through multitasking triplet generative adversarial networks. *arXiv preprint arXiv:1803.09059*, 2018.

[41] Van Hai Do, Nancy F Chen, Boon Pang Lim, and Mark Hasegawa-Johnson. Multi-task learning using mismatched transcription for under-resourced speech recognition. 2017.

[42] Homer Dudley and S Balashek. Automatic recognition of phonetic patterns in speech. *The Journal of the Acoustical Society of America*, 30(8):721–732, 1958.

[43] Stéphane Dupont, Christophe Ris, Olivier Deroo, and Sébastien Poitoux. Feature extraction and acoustic modeling: an approach for improved generalization across languages and accents. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 29–34. IEEE, 2005.

[44] Ellen Eide and Herbert Gish. A parametric approach to vocal tract length normalization. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 346–348. IEEE, 1996.

[45] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 2007.

[46] James W Forgie and Carma D Forgie. Results obtained from a vowel recognition computer program. *The Journal of the Acoustical Society of America*, 31(11):1480–1489, 1959.

[47] Sadaoki Furui. Generalization problem in asr acoustic model training and adaptation. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 1–10. IEEE, 2009.

[48] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

[49] Mark JF Gales, Kate M Knill, Anton Ragni, and Shakti P Rath. Speech recognition and keyword spotting for low-resource languages: Babel project research at cued. In *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[50] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

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

[52] Arnab Ghoshal, Pawel Swietojanski, and Steve Renals. Multilingual training of deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7319–7323. IEEE, 2013.

[53] Ritwik Giri, Michael L Seltzer, Jasha Droppo, and Dong Yu. Improving speech recognition in reverberation using a room-aware deep neural network and multi-task learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5014–5018. IEEE, 2015.

[54] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[55] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE, 1992.

[56] Yifan Gong. Speech recognition in noisy environments: A survey. *Speech communication*, 16(3):261–291, 1995.

[57] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[58] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[59] František Grézl and Martin Karafiát. Boosting performance on low-resource languages by standard corpora: An analysis. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 629–636. IEEE, 2016.

[60] Frantisek Grézl, Martin Karafiát, and Karel Vesely. Adaptation of multilingual stacked bottle-neck neural network structure for new language. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7654–7658. IEEE, 2014.

[61] Yu Gu and Yongguo Kang. Multi-task wavenet: A multi-task generative model for statistical parametric speech synthesis without fundamental frequency conditions. *arXiv preprint arXiv:1806.08619*, 2018.

[62] Çağlar Gülçehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. *The Journal of Machine Learning Research*, 17(1):226–257, 2016.

[63] Reinhold Haeb-Umbach and Hermann Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 13–16. IEEE, 1992.

[64] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[65] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep Speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[66] Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *arXiv preprint arXiv:1408.2873*, 2014.

[67] Kohei Hara, Koji Inoue, Katsuya Takanashi, and Tatsuya Kawahara. Prediction of turn-taking using multitask learning with prediction of backchannels and fillers. *Listener*, 162:364, 2018.

[68] Di He, Boon Pang Lim, Xuesong Yang, Mark Hasegawa-Johnson, and Deming Chen. Improved asr for under-resourced languages through multi-task learning with acoustic landmarks. *arXiv preprint arXiv:1805.05574*, 2018.

[69] Weipeng He, Petr Motlicek, and Jean-Marc Odobez. Joint localization and classification of multiple sound sources using a multi-task neural network. *Proc. Interspeech 2018*, pages 312–316, 2018.

[70] Kenneth Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July 2011.

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

[72] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[73] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[74] Qiong Hu, Zhizheng Wu, Korin Richmond, Junichi Yamagishi, Yannis Stylianou, and Ranniery Maia. Fusion of multiple parameterisations for dnn-based sinusoidal speech synthesis with multi-task learning. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[75] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE, 2013.

[76] Zhen Huang, Jinyu Li, Sabato Marco Siniscalchi, I-Fan Chen, Ji Wu, and Chin-Hui Lee. Rapid adaptation for deep neural networks through multi-task learning. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[77] Abhinav Jain, Minali Upreti, and Preethi Jyothi. Improved accented speech recognition using accent embeddings and multi-task learning. *Proc. Interspeech 2018*, pages 2454–2458, 2018.

[78] B-H Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of markov chains. *AT&T technical journal*, 64(6):1235–1249, 1985.

[79] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.

[80] DY Kim, S Umesh, MJF Gales, T Hain, and PC Woodland. Using vtln for broadcast news transcription. In *Eighth International Conference on Spoken Language Processing*, 2004.

[81] Jaebok Kim, Gwenn Englebienne, Khiet P Truong, and Vanessa Evers. Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning. *arXiv preprint arXiv:1708.03920*, 2017.

[82] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4835–4839. IEEE, 2017.

[83] Dennis H Klatt. Review of the arpa speech understanding project. *The Journal of the Acoustical Society of America*, 60(S1):S10–S10, 1976.

[84] Kate M Knill, Mark JF Gales, Shakti P Rath, Philip C Woodland, Chao Zhang, and S-X Zhang. Investigation of multilingual deep neural networks for spoken term detection. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 138–143. IEEE, 2013.

[85] Kalpesh Krishna, Shubham Toshniwal, and Karen Livescu. Hierarchical multitask learning for ctc-based speech recognition. *arXiv preprint arXiv:1807.06234*, 2018.

[86] Nagendra Kumar and Andreas G Andreou. Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition. *Speech communication*, 26(4):283–297, 1998.

[87] Duc Le, Zakaria Aldeneh, and Emily Mower Provost. Discretized continuous speech emotion recognition with multi-task deep recurrent neural network. *Interspeech, 2017 (to apear)*, 2017.

[88] Kai-Fu Lee, Hsiao-Wuen Hon, and Raj Reddy. An overview of the sphinx speech recognition system. In *Readings in speech Recognition*, pages 600–610. Elsevier, 1990.

[89] Li Lee and Richard C Rose. Speaker normalization using efficient frequency warping procedures. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 353–356. IEEE, 1996.

[90] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech & language*, 9(2):171–185, 1995.

[91] M Paul Lewis, Gary F Simons, Charles D Fennig, et al. *Ethnologue: Languages of the world*, volume 16. SIL international Dallas, TX, 2009.

[92] Yi Liu, Liang He, Jia Liu, and Michael T Johnson. Speaker embedding extraction with phonetic information. *arXiv preprint arXiv:1804.04862*, 2018.

[93] Reza Lotfian and Carlos Busso. Predicting categorical emotions by jointly learning primary and secondary emotions through multitask learning. *Proc. Interspeech 2018*, pages 951–955, 2018.

[94] Bruce T Lowerre. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1976.

[95] Liang Lu, Lingpeng Kong, Chris Dyer, and Noah A Smith. Multitask learning with ctc and segmental crf for speech recognition. *arXiv preprint arXiv:1702.06378*, 2017.

[96] Youyi Lu, Fei Lu, Siddharth Sehgal, Swati Gupta, Jingsheng Du, Chee Hong Tham, Phil Green, and Vincent Wan. Multitask learning in connectionist speech recognition. In *Proceedings of the Australian International Conference on Speech Science and Technology*, 2004.

[97] Marco Matassoni, Roberto Gretter, Daniele Falavigna, and Diego Giuliani. Non-native children speech recognition through transfer learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6229–6233. IEEE, 2018.

[98] Zhong Meng, Jinyu Li, Zhuo Chen, Yong Zhao, Vadim Mazalov, Yifan Gong, et al. Speaker-invariant training via adversarial learning. *arXiv preprint arXiv:1804.00732*, 2018.

[99] Aanchan Mohan and Richard Rose. Multi-lingual speech recognition with low-rank multi-task deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4994–4998. IEEE, 2015.

[100] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

[101] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[102] Takafumi Moriya, Sei Ueno, Yusuke Shinohara, Marc Delcroix, Yoshikazu Yamaguchi, and Yushi Aono. Multi-task learning with augmentation strategy for acoustic-to-word attention-based encoder-decoder speech recognition. *Proc. Interspeech 2018*, pages 2399–2403, 2018.

[103] K Nagata. Spoken digit recognizer for japanese language. *NEC research & development*, (6), 1963.

[104] Joao Neto, Luís Almeida, Mike Hochberg, Ciro Martins, Luis Nunes, Steve Renals, and Tony Robinson. Speaker-adaptation for hybrid hmm-ann continuous speech recognition system. In *Fourth European Conference on Speech Communication and Technology*, 1995.

[105] Harry F Olson and Herbert Belar. Phonetic typewriter. *The Journal of the Acoustical Society of America*, 28(6):1072–1081, 1956.

[106] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[107] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

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

[109] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for dnn-based keyword spotting. In *INTERSPEECH*, pages 760–764, 2016.

[110] Srinivas Parthasarathy and Carlos Busso. Jointly predicting arousal, valence and dominance with multi-task learning. *INTERSPEECH, Stockholm, Sweden*, 2017.

[111] Shahla Parveen and Phil Green. Multitask learning in connectionist robust asr using recurrent neural networks. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[112] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

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

[114] Gueorgui Pironkov, Stephane Dupont, and Thierry Dutoit. Multi-task learning for speech recognition: an overview. In *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)*, volume 192, 2016.

[115] Mark A Pitt, Keith Johnson, Elizabeth Hume, Scott Kiesling, and William Raymond. The buckeye corpus of conversational speech: labeling conventions and a test of transcriber reliability. *Speech Communication*, 45(1):89–95, 2005.

[116] Daniel Povey, Stephen M Chu, and Balakrishnan Varadarajan. Universal background model based speech recognition. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4561–4564. IEEE, 2008.

[117] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[118] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755, 2016.

[119] Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. wav2letter++: The fastest open-source speech recognition system. *arXiv preprint arXiv:1812.07625*, 2018.

[120] Lorien Y Pratt. *Transferring previously learned back-propagation neural networks to new learning tasks*. Rutgers University New Brunswick, NJ, 1993.

[121] Ying Qin, Tan Lee, Siyuan Feng, and Anthony Pak Hin Kong. Automatic speech assessment for people with aphasia using tdnn-blstm with multi-task learning. *Proc. Interspeech 2018*, pages 3418–3422, 2018.

[122] Lawrence R. Rabiner. First-hand: The hidden markov model, 2015. [Online; accessed 01-January-2018].

[123] Kanishka Rao and Haşim Sak. Multi-accent speech recognition with hierarchical grapheme based models. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4815–4819. IEEE, 2017.

[124] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 193–199. IEEE, 2017.

[125] Douglas A Reynolds. Channel robust speaker verification via feature mapping. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–53. IEEE, 2003.

[126] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[127] Andrew Rosenberg, Kartik Audhkhasi, Abhinav Sethy, Bhuvana Ramabhadran, and Michael Picheny. End-to-end speech recognition and keyword search on low-resource languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5280–5284. IEEE, 2017.

[128] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[129] Shigeki Sagayama, Koichi Shinoda, Mitsuru Nakai, and Hiroshi Shimodaira. Analytic methods for acoustic model adaptation: A review. In *Proc. Isca ITR-Workshop*, pages 67–76, 2001.

[130] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044. ACM, 2014.

[131] Ramon Sanabria and Florian Metze. Hierarchical multi task learning with ctc. *arXiv preprint arXiv:1807.07104*, 2018.

[132] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*, 2017.

[133] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, pages 55–59, 2013.

[134] Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1-2):31–51, 2001.

[135] Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6965–6969. IEEE, 2013.

[136] Mohammed Senoussaoui, Patrick Kenny, Najim Dehak, and Pierre Dumouchel. An i-vector extractor suitable for speaker recognition with both microphone and telephone speech. In *Odyssey*, page 6, 2010.

[137] Dmitriy Serdyuk, Kartik Audhkhasi, Philémon Brakel, Bhuvana Ramabhadran, Samuel Thomas, and Yoshua Bengio. Invariant representations for noisy speech recognition. *arXiv preprint arXiv:1612.01928*, 2016.

[138] Koichi Shinoda. Acoustic model adaptation for speech recognition. *IEICE transactions on information and systems*, 93(9):2348–2362, 2010.

[139] Yusuke Shinohara. Adversarial multi-task learning of deep neural networks for robust speech recognition. In *INTERSPEECH*, pages 2369–2372, 2016.

[140] Matthew A Siegler, Uday Jain, Bhiksha Raj, and Richard M Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA speech recognition workshop*, volume 1997, 1997.

[141] Jan Stadermann, Wolfram Koska, and Gerhard Rigoll. Multi-task learning strategies for a recurrent neural net in a hybrid tied-posteriors acoustic mode. In *Proc. of Interspeech 2005-Proc. Europ. Conf. on Speech Communication and Technology, Lisbon, Portugal*, 2005.

[142] Sining Sun, Ching-Feng Yeh, Mei-Yuh Hwang, Mari Ostendorf, and Lei Xie. Domain adversarial training for accented speech recognition. *arXiv preprint arXiv:1806.02786*, 2018.

[143] Pawel Swietojanski, Peter Bell, and Steve Renals. Structured output layer with auxiliary targets for context-dependent acoustic modelling. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[144] Remco Teunen, Ben Shahshahani, and Larry Heck. A model-based transformational approach to robust speaker recognition. In *Sixth International Conference on Spoken Language Processing*, 2000.

[145] Sebastian Thrun and Lorien Pratt. *Learning to learn.* Springer Science & Business Media, 2012.

[146] Rong Tong, Nancy F Chen, and Bin Ma. Multi-task learning for mispronunciation detection on singapore children's mandarin speech. *Proc. Interspeech 2017*, pages 2193–2197, 2017.

[147] Shubham Toshniwal, Tara N Sainath, Ron J Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, and Kanishka Rao. Multilingual speech recognition with a single end-to-end model. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4904–4908. IEEE, 2018.

[148] Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *arXiv preprint arXiv:1704.01631*, 2017.

[149] László Tóth, Gábor Gosztolya, Tamás Grósz, Alexandra Markó, and Tamás Gábor Csapó. Multi-task learning of speech recognition and speech synthesis parameters for ultrasound-based silent speech interfaces.

[150] Aditay Tripathi, Aanchan Mohan, Saket Anand, and Maneesh Singh. Adversarial learning of raw speech features for domain invariant speech recognition. *arXiv preprint arXiv:1805.08615*, 2018.

[151] Stavros Tsakalidis. *Linear transforms in automatic speech recognition: estimation procedures and integration of diverse acoustic data*. PhD thesis, Johns Hopkins University, 2005.

[152] Zoltan Tuske, David Nolden, Ralf Schluter, and Hermann Ney. Multilingual mrasta features for low-resource keyword search and speech recognition systems. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7854–7858. IEEE, 2014.

[153] JC Vásquez-Correa, T Arias-Vergara, JR Orozco-Arroyave, and E Nöth. A multitask learning approach to assess the dysarthria severity in patients with parkinson's disease.

[154] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

[155] Ngoc Thang Vu, David Imseng, Daniel Povey, Petr Motlicek, Tanja Schultz, and Hervé Bourlard. Multilingual deep neural network based acoustic modeling for rapid language adaptation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7639–7643. IEEE, 2014.

[156] Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*, pages 1225–1237. IEEE, 2015.

[157] Qing Wang, Wei Rao, Sining Sun, Leib Xie, Eng Siong Chng, and Haizhou Li. Unsupervised domain adaptation via domain adversarial training for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4889–4893. IEEE, 2018.

[158] Robert Weide. The carnegie mellon pronouncing dictionary [cmudict. 0.3]. pittsburg, pa: Department of computer science, 1993.

[159] Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. Sequence-to-sequence models can directly transcribe foreign speech. *CoRR*, abs/1703.08581, 2017.

[160] Jeremy HM Wong and Mark John Gales. Sequence student-teacher training of deep neural networks. 2016.

[161] Phil C Woodland. Speaker adaptation for continuous density hmms: A review. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001.

[162] Phil C Woodland and Steve J Young. The htk tied-state continuous speech recogniser. In *Third European Conference on Speech Communication and Technology*, 1993.

[163] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4460–4464. IEEE, 2015.

[164] Chenglin Xu, Wei Rao, Eng Siong Chng, and Haizhou Li. A shifted delta coefficient objective for monaural speech separation using multi-task learning. *Proc. Interspeech 2018*, pages 3479–3483, 2018.

[165] Haihua Xu, Van Hai Do, Xiong Xiao, and Eng Siong Chng. A comparative study of BNF and DNN multilingual training on cross-lingual low-resource speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[166] Xuesong Yang, Kartik Audhkhasi, Andrew Rosenberg, Samuel Thomas, Bhuvana Ramabhadran, and Mark Hasegawa-Johnson. Joint modeling of accents and acoustics for multi-accent speech recognition. *arXiv preprint arXiv:1802.02656*, 2018.

[167] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[168] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3:175, 2002.

[169] Steve J Young, Julian J Odell, and Philip C Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics, 1994.

[170] Yu Zhang, Pengyuan Zhang, and Yonghong Yan. Attention-based lstm with multi-task learning for distant speech recognition. *Proc. Interspeech 2017*, pages 3857–3861, 2017.