1 Goals & Import of Dissertation

The main goal of this dissertation is to develop training methods for neural networks which lead to better classification of data from unseen conditions. This dissertation falls into the category of research on model generalization and model robustness. In particular, this work focuses on small data sets, where traditional neural net training lead to overfitting.

Using the framework of Multi-Task Learning, I train a single neural net to classify data with several labels, encouraging the hidden layers to learn generic, useful represetations of the data. This is accomplished without any explicit adaptation of model parameters or data transformations.

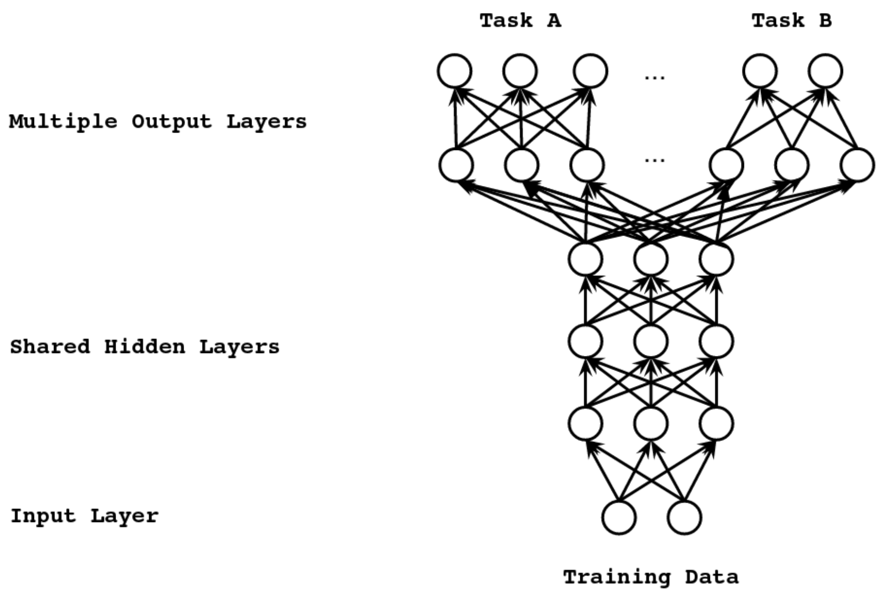
In the past people have dealt with new data by (1) adapting an existing model to the new data, or (2) normalizing the new data to look more like the old training data. More recently, Multi-Task Learning (MTL) has been found to produce models which are better at handling new data, because these models inherently learn more robust features which generalize to unseen domains.

In the following, the neural nets of interest are acoustic models for automatic speech recognition, and the unseen data conditions are (1) `new noise`, (2) a `new speaker`, or (3) a `new language`. For each of these three cases, I have two conditions of data. I train the model on one data condition, and then test the on other.

1. For the `Noise` condition, I train on clean audio and test on noisy audio.
2. For the `Speaker` condition, I train on a set of speakers, and I test on an unseen speaker.
3. For the `Language` condition, I train on (mostly) English, and I test exclusively on Kyrgyz. I must include *some* Kyrgyz in training because the DNN+HMM pipeline requires the testing phonemes defined in the model.

My approach is innovative because it doesn't require massive data sets, as some other popular reseach labs have done (Google, Firefox, Baidu, Microsoft, etc). Further more, my approach is not specific to any one dataset, and it can be used to train any neural net (not just for speech recognition).

MTL training for neural nets works by training a set of hidden layers to perform multiple tasks (multiple output layers). Here is an example of MTL architecture for an acoustic model from Heigold 2013.



During training all `Tasks` are trained in unison, but during testing I only use `Task A` to decode new audio. The only benefit from these extra tasks comes from the training phase, and their influence on the shared hidden layers.

This is the same architecture that will be used in my dissertation, but obviously I will have more layers and more nodes per layer. The number of input nodes on my neural nets corresponds to the dimensionality of audio features, and the number of nodes on each output layer corresponds to the number of phonemes (i.e. monophones or triphones) I've defined for the language.

In this dissertation (following the MTL tradition), each output softmax layer will be referred to as a `Task`.

2 Overview of Speech Recognition

This section contains an overview of the training and testing procedures for standard automatic speech recognition (ASR) pipelines. The overview will provide the reader with a technical grounding in ASR, so that the rest of the dissertation will have some point of reference.

2.1 From DARPA to Siri: some historical context

These three vains of ASR research all exist in some historical context, and it is insightful to compare them as such.

2.1.1 Isolated Word Systems

In the 50's and 60's, much of the work on ASR focused on isolated word, speaker-dependent speech recognition. The typical use case was a single adult male carefully reading off single digits [0-9] into a microphone, or saying words like "start" or "stop". The research was funded by the US Military, and as such all the target tasks were relevant to military operations. Specifically, the military was looking to control some operations by voice as opposed to buttons or typing, and they were happy to tailor the recognizer to work for just one person, as long as it worked well. Each user had to train the system on his or her own voice by dictating some presented sentences.

Given that command and control was the target application, isolated words were a logical choice for the unit of recognition. The user said one word, and the decoder compared that word to all the words it had stored on disc, and returned the closest match. This assumes the user actually said a word in the machine's vocabulary, and so each user was trained to work with the machine, and the machine was trained to work with each user. By modern standards, these recognizers were not flexible. However, what these recognizers lacked in flexibility they gained in accuracy. There was no guessing which word was spoken, given all words in the English language. There was no guessing how many words were spoken either, because the user was trained to say one word at a time. Furthermore, there was no real issue of noise conditions, because recording and testing would be done in the same place. There was no worry about microphone robustness or sampling-rate issues, because the creators knew exactly beforehand what hardware the recognizer ran on. All these problems current ASR research faces were unknown to these early researchers, and their approach was simple, and efficient. This was pattern-matching at its finest. The computer stored a representation of each word in its vocabulary, and given new input from the user, the input was checked against all exemplars in the system.

This approach worked just fine until users wanted more. 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 every word 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.

Also, speaking slowly and carefully only one word at a time was a pain. Humans don't do that. In our story today, this is the complaint most relevant to us.

2.1.2 Continuous Speech Systems and the phoneme

In order to make a recognizer that could handle a string of words, where the length of that string was unknown to the system, major overhauls of the Isolated Word system were needed.

First of all, the acoustics of the words themselves vary drastically in continuous speech. In isolated systems, every word is preceded by silence and succeded by silence. This padding is great for pattern recognition, because the edges of words will be pretty stable, acoustically. When it comes to continuous speech, however, word beginings can be preceded by the tail end of the previous word. As a result, the acoustics of both words become distorted. This distortion is called co-articulation, and it causes the edge sounds to blend together.

It is clear that storing exemplars of words on disc is not an option with continuous speech systems. A system which can recognize 100 words in isolation never has to worry about variants of sounds. That system merely needs to account for the length of sounds, not whether or not their acoustics change. If we want that same system to handle strings of up to three words (e.g. "one two three"), it should ideally store a version of each word in every possible context. That would mean 100 words with 101 preceding contexts and 101 suceeding contexts (where silence is also a context). So, we've gone from needing to store a mere 100 words to storing over a million. The decoding speed would be horrendous, the space used on disk would be prohibitive, not to mention the time needed by the user to record all those contexts!

Modeling whole words became an obvious bottleneck to recognition of continuous speech. This is where linguistics came on the scene in the engineering world with the phoneme.

The phoneme is really just a fancy Greek word to mean speech sound. Every language has a finite set of phonemes, and with this finite set of phonemes all words are created. This is music to the ears of ASR researchers, because all of a sudden, an infinite number of words and strings of words can be boiled down to set of finite acoustic events. What's more, typically languages don't have more than 50 of these phonemes! Where simple systems had hard limits of 100 or 1000 words, with only 50 discrete phonemes there is no upper limit to the number of words a system can recognize.

2.1.3 Modern Large Vocabulary Continuous Speech Recognition

In the following, I will provide an overiew of three main pipelines in ASR.

The first two pipelines make use of Hidden-Markov Models (HMMs) and are still widely used today in practice. The first HMM-based models use Gaussian Mixture Model (GMMs) as the emission probabilities of states in the HMMs, whereas the second HMM-based approach uses Deep Neural Networks (DNNs) for the emission probabilities. The first method is refered to as the GMM-HMM approach, and the second is the DNN-HMM approach. The pipeline at decoding for these two models is almost identical, but their training is significantly different.

The third approach, which is gaining lots of research interest currently, is end-to-end ASR with DNNs. The training and decoding of these models is completely different from the former HMM-based approaches, as I will explain below.

2.2 Gaussians + HMMs

2.3 Neural Nets + HMMs

2.4 end-to-end Neural Nets

3 Overview of Multi-Task Learning

A task (in classification) is a set of (data,label) pairs.

Most machine learning training uses single-task learning (e.g. classifying an image as a digit [0-9].

In Mutli-Task learning, we learn multiple tasks which share useful information. An example of a non-useful auxillary task is classifying a number as greater or less than 7, when the main task is to identify the identity of a single digit.

4 Background Literature

Here I will cover the literature relevant to working with small (or completely new) datasets. There are two main approaches, (1) adapt a model from one training dataset to a new, smaller dataset; (2) create a model that is robust enough to handle data from multiple domains.

- **Model Adaptation: (e.g. Speaker; Language)**
- **Model Robustness: (e.g. Noise; Channel)**

5 Experiments

This section contains the main contributions of the dissertation research.

This dissertation investigates training methods for acoustic modeling in the Neural Net + HMM ASR pipeline.

I aim to produce acoustic models which perform better (i.e. lower Word Error Rates) on datasets which are not similar the original training dataset.

I investiage the effectiveness of different `Tasks` (eg. linguistic `Tasks` vs machine learning `Tasks`) in a Multi-task Learning framework.

5.1 Data

I am creating acoustic models which generalize well to new data. To measure how well the models generalize, I use a set of speech corpora which exhibit some interesting differences between training and testing data. These differences between corpora exemplify the typical challenges faced in speech recognition generalization.

The training and testing data will differ in either (1) the recording `noise` conditions, (2) who the `speaker` is, or (3) what `language` the speaker is using. The following table shows which data sets are used for each audio condition.

		Corpus	
		Train	**Test**
Audio Condition	**Noise**	TIDIGITS	Aurora 5
	Speaker	LibriSpeech-A	LibriSpeech-B
	Language	LibriSpeech	Kyrgyz Audiobook

Table 1: Speech Corpora

5.2 Model Training Procedure

This dissertation investigates the creation of new tasks for MTL, either using (1) linguist-expert knowledge, (2) ASR Engineer-expert knowledge, or (3) general Machine Learning knowledge.

The former two knowledge sources are useful for buidling acoustic models, but not much else. On the other hand, the final knowledge source (general machine learning concepts) can be applied to *any* classification problem.

The three knowledge sources will be abbreviated as such:

- (LING) **Linguistic Knowledge**
- (ASR) **Traditional Speech Recognition Pipeline**
- (ML) **General Machine Learning**

	Knowledge Source		
	LING	**ASR**	**ML**
Experiments	voicing	monophones	k-means
	place	1/2 triphones	random forests
	manner	3/4 triphones	bootstrapped resamples

Table 2: Experimental Setup

Each of these categories contains a wealth of ideas, but I will consolidate each into three experiments. With three experiments for each knowledge source, my dissertation will contain nine (9) experimental conditions (for each audio condition).

Specifically, I will use the following concepts to create new tasks to be used in MTL training:

Each of these tasks will be added to a Baseline model. More specifically, the Baseline model will be a Neural Net with a single output layer (Task A), and the tasks above will be added as a second task (Task B). You can think of the tasks as simply a new set of labels for the existing data set. For example, when the Ling task of voicing is used, any audio segment labeled `[b]` will be assigned the new label `voiced`.

When these experiments will be applied to each of the three audio conditions, we get the following 30 experiments:

Data Condition	**Train Data**	**Test Data**	**MTL Training Tasks**	**Num. Exps**
Noise	TIDIGITS	Aurora 5	Basline	1
			Baseline + LING	3
			Baseline + ASR	3
			Baseline + ML	3
Speaker	LibriSpeech-A	LibriSpeech-B	Baseline	1
			Baseline + LING	3
			Baseline + ASR	3
			Baseline + ML	3
Language	LibriSpeech + Kyrgyz-A	Kyrgyz-B	Baseline	1
			Baseline + LING	3
			Baseline + ASR	3
			Baseline + ML	3
				30

Table 3: Experimental Setup

5.3 Task Creation Specifics

1. **Baseline**

   All the following architectures will be compared to the performance of the following baseline.

   To account for any advantage mutliple output layers may bring about, the baseline also 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.

   During testing, *only one* of the `Tasks` is used. The additional `Tasks` are dropped and the `Baseline 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. The `Tasks` may in fact not be the best option for final classification; they serve as "training wheels" which are then removed once the net is ready.

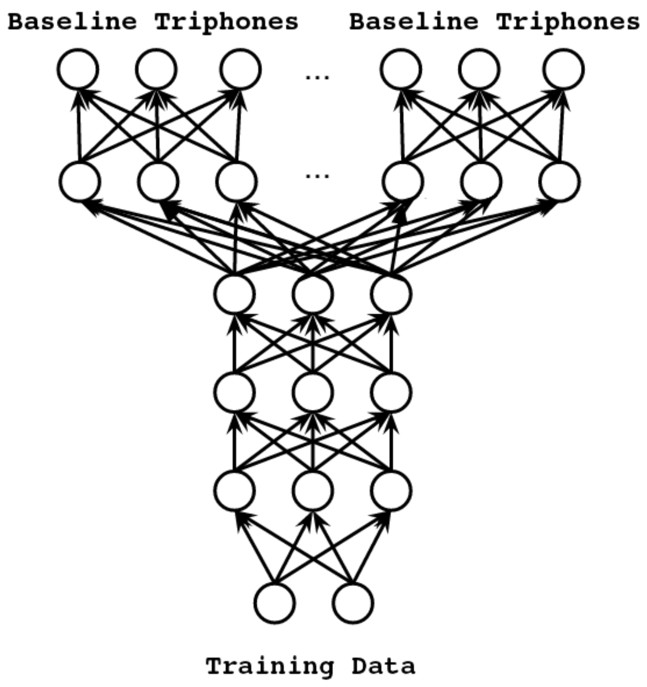


Figure 1: `Baseline`

2. **LING**

All of the linguistic knowledge tasks view the phoneme as a bundle of features.

Using standard features from articulatory phonetics (voicing, place, and manner), the following tasks generate labels for each data point by collapsing the given phoneme labels along one of these three dimensions.

All information from one dimension is removed from the labeled data. This forces the classifier to rely on audio signal features which do not relate to that dimension. The DNN must project the input data into a new space for classification, using only information from the other two dimensions.

(a) VOICING

Voicing information is removed from the data labels.

Speaker Robustness Experiments: The training data is a 4.5 hour subset of Librispeech, with mixed speakers, men and women. The testing data is 30 minutes of speech from two speakers (one man one woman).

First, two separate GMM-HMM models are trained on the training data. The first GMM-HMM model uses the standard CMUDict phoneset (39 phones + stress variants).

From this standard phoneset, the normal 3-state monophones are trained from a flat-start via EM training. A total of XXX states are trained with a total of XXX Gaussian components over XXX iterations of EM. These monophones are then expanded into context-dependent triphones via a phonetic decision tree, with a maximum of XXX leaves. The resulting leaves (state clusters) are then trained with XXX Gaussian components over XXX iterations of EM. The final model acheives a WER of XXX on the testing data.

The second GMM-HMM model trained differs from the first model in its set of initial phones. Instead of building monophones (and then triphones) from the standard CMUDict, this `-Voicing` model collapsed all voicing information from the phonetic dictionary (i.e. the lexicon file).

```
B P   --> P
CH JH --> CH
D T   --> T
DH TH --> TH
F V   --> F
G K   --> G
S Z   --> S
SH ZH --> SH
```

(b) PLACE

All place information is removed from the data labels.

```
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 nasal
L R          --> R        voiced laterals
Y W          --> Y        voiced approximants
```

(c) MANNER

All manner information is removed from the data labels.

```
B M V W --> W          voiced labials
P F     --> P          voiceless labials
```

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

Table 4: CMUDict Phoneset

```
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
```

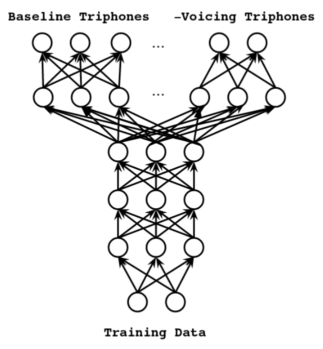


Figure 2: `-Voicing`

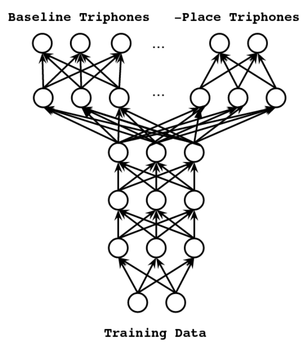


Figure 3: `-Place`

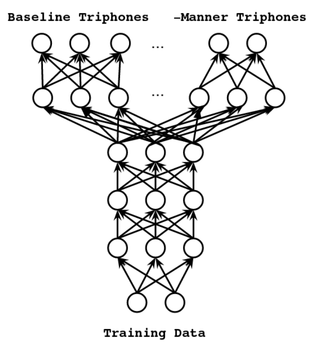


Figure 4: `-Manner`

3. **ASR**

   All the following tasks relate to the structure of the phonetic decision tree used in the traditional ASR pipeline to cluster context-dependent triphones. In GMM training the leaves of the decision tree are then assigned their own Gaussians, and in DNN training the same leaves are used as labels during training via backprop.

   The main intuition behind these experiments is that in using the decision tree labels as targets for the DNN classifier, we are performing model transfer. The decision tree and it's associated Gaussians perform classification, and we are merely training a DNN to perform that same task. So, the decision tree can be thought of as a single task for the DNN to learn.

   However, the DNN only sees the leaves of the decision tree. It doesn't see any of the branches, or any of its wonderful linguistic structure. So, in order to force the DNN to learn the information hidden in the decision tree, the following tasks are like cross-sections of the tree, slicing it from leaves up. The DNN then has to learn how to read these cross-sections, and how to map data onto each layer.

   If we slice the tree at the roots, we have the MONOPHONES. If we slice down half-way (1/2 TRIPHONES), we have more contextual information than monophones but less than full triphones. If go a little farther down (3/4 TRIPHONES), we get even more context, but less general information about the original phoneme.

   (a) monophones
   When we chop the tree at the roots.

   (b) 1/2 triphones
   Chop the tree half-way down.

   (c) 3/4 triphones
   Chopping a little further.

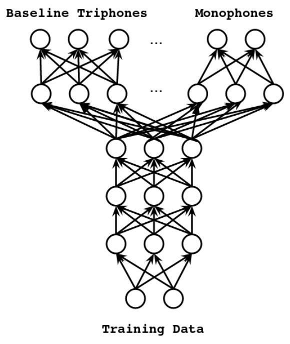


Figure 5: `Monophones`

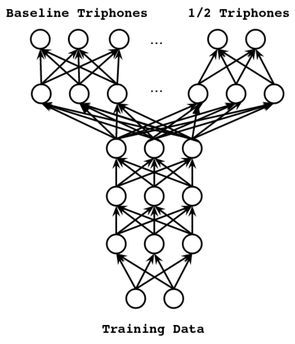


Figure 6: `1/2 Triphones`

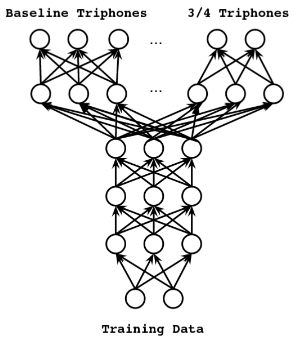


Figure 7: `3/4 Triphones`

4. **ML**

   The following tasks do not make use of any linguistic knowledge or any part of the ASR pipeline. The only things needed to perform these tasks is labeled data.

   The two approaches above use linguistics or the ASR pipeline to force a DNN to learn structure about the data, because that information is useful for classification.

   We typically do not have this kind of *a priori* information about the datasets we use in Machine Learning. Therefore, an interesting problem is how to learn this structure in a data set when we don't have access to that expert knowledge.

   The following tasks force the DNN to learn structure in the data without any knowledge about that structure. In order to do so, I make the assumption that the data does in fact have heirarchical relations. That is, I assume the `(data,label)` pairs were generated by something like a phonetic decision tree, and I try to recover the structure of that tree.

   (a) k-means

   Standard k-means on the data, with the caveat that labels cannot be split across clusters. A first round of clustering is performed, and then all data from the same original label are shifted to the cluster with the most data points from that label. Then, centroids are recalculated, and data is re-clustered. This adapated k-means should find related data points in the same clusters. If k-means is working, we would expect to be able to recover phonemes (monophones) from the labeled triphone data.

   (b) random forest

   In another attempt to cluster triphones along phonetic categories, the random forest procedure works as follows: (1) take a random subset of the labels, (2) train a random forest with all data points associated with those labels, (3) re-classify all the rest of the data with the new random forest. In this way, we will reduce the number of labels (eg. out of 2,000 triphone labels I choose 500), and classify unseen data as its closest seen data point.

   (c) bootstrapped resamples

   In this approach, new labels are not generated at all. The separate tasks for the DNN are just different samples of the data.

   Some sub-samples may exhibit a more useful decision plane than others, and if we randomly subsample for multiple tasks, the different decision planes will all have something in common. The individual peculiarities of one sub-sample will have to be ignored for the DNN to perform well on all tasks.

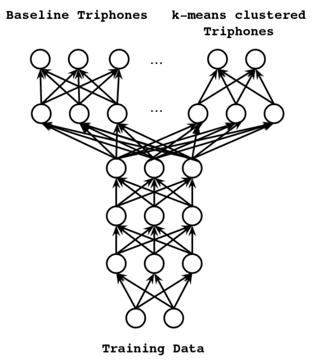


Figure 8: `k-means`

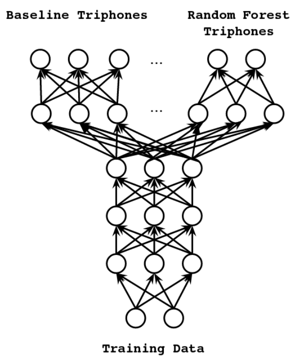


Figure 9: `1/2 Triphones`

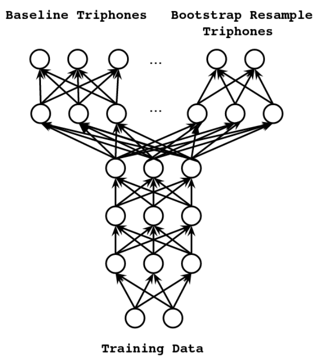


Figure 10: `3/4 Triphones`