THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

Convolutional Neural Networks for Speaker-Independent Speech Recognition

by

Eugene Belilovsky

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Engineering

May 2, 2011

Advisor

Dr. Carl Sable

THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.

Dr. Simon Ben-Avi
Acting Dean, School of Engineering

Dr. Carl Sable
Candidate's Thesis Advisor

Abstract

In this work we analyze a neural network structure capable of achieving a degree of invariance to speaker vocal tracts for speech recognition applications. It will be shown that invariance to a speaker's pitch can be built into the classification stage of the speech recognition process using convolutional neural networks, whereas in the past attempts have been made to achieve invariance on the feature set used in the classification stage. We conduct experiments for the segment-level phoneme classification task using convolutional neural networks and compare them to neural network structures previously used in speech recognition, primarily the time-delayed neural network and the standard multilayer perceptron. The results show that convolutional neural networks can in many cases achieve superior performance than the classical structures.

Acknowledgments

I wish to thank Professor Carl Sable for his guidance and encouragement in advising this work as well as professors Fred Fontaine, Hamid Ahmad, Kausik Chatterjee, and the rest of the Cooper Union faculty for providing me with an excellent foundation in my undergraduate and graduate studies.

I would like to thank Florian Mueller of Luebeck University for his guidance and advice during my stay at the University of Luebeck. I would like to thank the RISE program sponsored by the German Academic Exchance Services (DAAD) for arranging and providing for my stay at Luebeck University, which gave me my first exposure to the field of speech recognition.

I would also like to thank my peers from the Cooper Union who have helped me develop this thesis. Brian Cheung for his help in determining optimal training parameters for the neural networks studied. Christopher Mitchell and Deian Stefan for their help in developing the ideas of this thesis. Sherry Young for her help in revising the thesis.

I wish to thank my parents and sisters for all their great support and encouragement over the years.

Contents

List of Figures

Chapter 1

Introduction

Speech recognition has been a popular field of research since the first speech recognizer was created in 1952 at Bell Labs [10]. There are many variants of speech recognition systems depending on the application. There exist many systems for recognizing only a limited vocabulary; an example of this is the common automatic answering machine which asks users to speak one of several words that are built into a database. Another limitation often made on speech recognition systems is the speaker; many systems, for example, are trained to recognize a particular user's voice, which increases the recognition performance.

After the results of [10] in recognizing isolated spoken digits, it was believed that the methods could easily be extended to recognizing any continous speech, and in fact it was stated for many years that the solution was only 5 years away [11]. Although great accuracy can be accomplished in the limited cases, such as the small vocabulary and particular speaker setups described above, creating a robust automatic-speech recognition (ASR) system is still an ongoing area of research. These systems, intended to recognize large-vocabulary speech from any user, have had mixed success. There are a great deal of applications of such systems including dictation software, easier searches, automatic transcription of video and audio files for easy search and reference, along with a host of other applications. Unfortunately these systems are currently still not robust enough for many real-life applications. The shape of the vocal tract of the user as well as their accent and noise from the environment and/or noise introduced through the processing of their speech, such as in digital phone lines, creates a great deal of problems for the automatic speech recognizer.

One of the primary differences in the shape of the vocal tract is the length of the vocal tract, which directly affects the pitch of a person's voice. Adult vocal tract lengths (VTLs) may differ by up to 25 percent. Thus vocal tract length is one of the key factors determining differences between the same utterances spoken by different speakers. Generally speakers with a longer vocal tract will produce lower pitched sounds [12]. A common approximation is that in the linear spectral domain, the short-time spectra of two speakers A and B, creating the same utterance, are approximately related by a constant factor α with the scaling relation $S_A(w) = S_B(\alpha w)$, where $S_A(w)$ and $S_B(w)$ are the short-time spectra of similar utterances spoken by speaker A and speaker B. Without further processing within the ASR system this spectral scaling causes degradation in the system performance [13].

Various methods have been proposed that attempt to counteract this scaling effect. Some methods attempt to adapt the acoustic models to the features of each utterance, these methods are known as MLLR techniques. Another set of methods attempts to normalize the features obtained. The most popular method in this group is called VTL normalization. These methods have in common the need for an additional adaptation step within the recognition process of the ASR system. Another group of methods attempts to extract features that are invariant to the spectral effects of VTL changes. Though not as mature these methods have a low computation cost and do not require any adaptation stage [13].

In this work we attempt to apply a classifier that has the properties of invariance built into the classification in order to achieve invariance to VTL without any additional steps of adaptation or transformation of the feature set. This classifier is a variant of the convolutional neural network (CNN) which has been successfully applied in the field of image recognition.

Previous work has been done in applying a similar classifer method to Chinese syllable recognition [14]. However, this work did not focus on exploiting the invariance properties of the network for achieving the invariance to VTL. Another work attempted to use a block-windowed neural network(BWNN) [15] to achieve invariance in the frequency domain. However this network lacked several structural advantages of the CNN, in particular the subsampling layer was not used in this work as well as the multiple feature maps generally employed in the CNN.

Chapter 2

Preliminaries

In this section we present the background material required for understanding the work undergone in this thesis. The production of speech, the machine learning approach to speech recognition, hidden Markov models (HMM), and neural networks will be discussed.

2.1 The Speech Signal

In this section we describe the basics of speech production and perception. The phoneme is the basic unit of language and is heavily used when discussing speech recognition. Speech utterances that form words in a language are generally grouped into phonemes, which are the smallest segmentable units of sounds that can form meaningful contrasts between utterances. Even though two speech sounds, or phones, might be different, if they are close enough to be indistinguishable to a native language speaker, then they are grouped together into one phoneme. For example the 'k' sound in "kit" and "skill" is actually pronounced differently in most dialects; however, this sound is generally indistinguishable to most speakers [16]. We will begin our look at the speech production process starting from when the speaker formulates a message that he wants to transmit to the listener via speech. At this point, we assume the speaker has the text of the message in mind. The message is then converted into language code, a set of phoneme sequences corresponding to the sounds that make up the words, along with information about the duration of sounds, loudness of the sounds and pitch accent associated with the sounds [17]. Once this language code is constructed the person executes a series of neuromuscular commands to cause the vocal cords to vibrate when appropriate and to shape the vocal tract such that the proper sequence of speech sounds is created and spoken. Figure 2.1 shows a basic diagram of the position of the vocal chords and tract. The actual speech production occurs as air enters the lungs through the trachea, the tensed vocal cords within the larnyx vibrate due to the air flow. The air flow is chopped into quasi-periodic pulses which are then modulated in frequency in passing through the throat and mouth cavaties. Depending on the position of the various articulators (jaw, tongue, lips, and mouth) different sounds are produced.

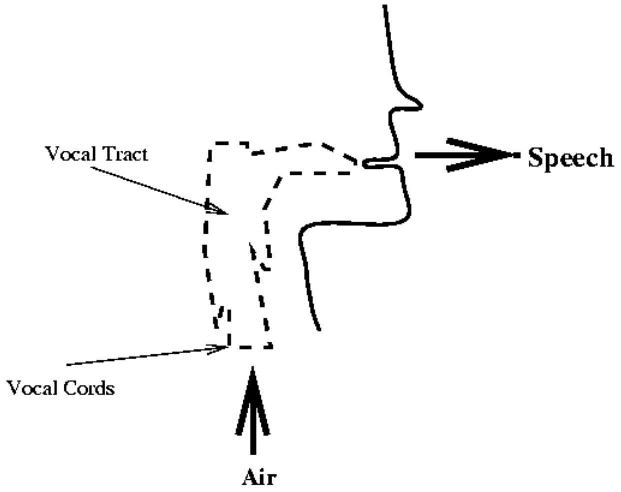

Figure 2.1 *Human speech production from [1]*

At the other end of this process is the listener. The acoustic wave enters the ear and pushes against the eardrum. Using the ossicles as levers, the force of this push is transmitted to the oval window. A large motion of the eardrum leads to smaller but more forceful motion at the oval window. This pushes on the fluid of the inner ear and waves begin moving down the basilar membrane of the cochlea [2].

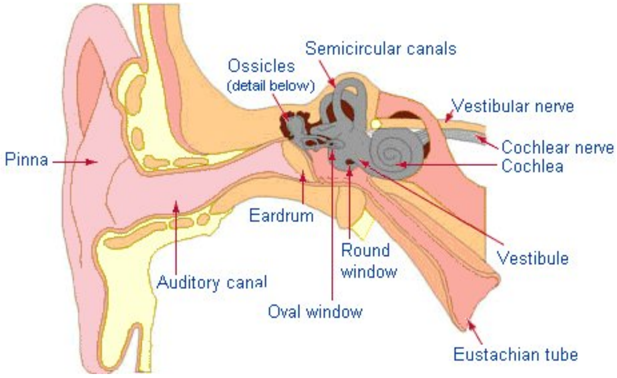

Figure 2.2 *Middle ear as depicted in [2]*

This basilar membrane provides a critical function which we attempt to mimic in many speech recognition feature extraction methods. This membrane mechanically calculates a runnning spectrum of the incoming signal. The membrane is tapered and it is stiffer at one end than at the other. The dispersion of fluid waves causes sound input of a certain frequency to vibrate some locations of the membrane more than other locations. Figure 2.2 shows the basilar membrane is located at the circular organ in the ear. If we unroll the cochlea as in Figure 2.3, one can see an example of how different stretches of the membrane's response provide a physical version of a spectrum for the acoustic signal [2]. The frequency categories created by this membrane are not linear, inspiring the feature extraction method used in this work.

After the speech signal has been processed by the basilar membrane, a neural transduction process then converts the spectral signal at the output of the basilar membrane into activity signals on the auditory nerve. This activity in the nerves is converted into a language code at the higher processing centers of the brain. Finally a

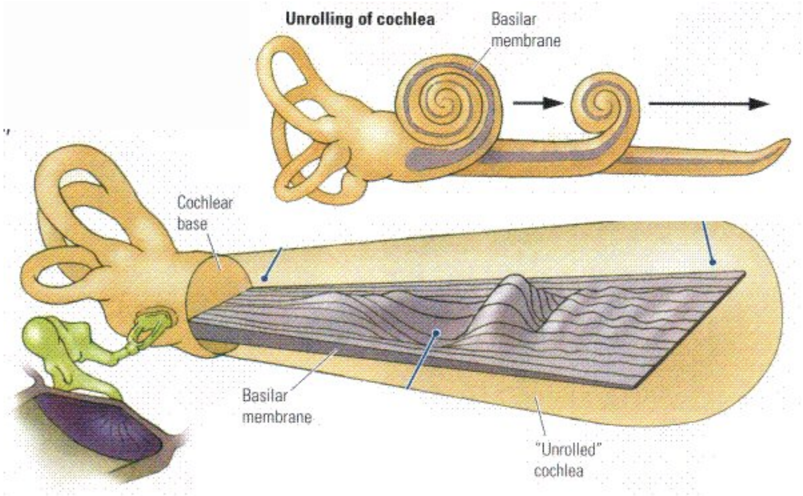

Figure 2.3 *Basilar membrane as depicted in [2]*

semantic analysis of this language code by the brain allows for message comprehension [17].

2.2 Categories of Phoneme Speech

There are several primary groupings for the basic phoneme utterances. Voiced speech sounds include the categories of utterances vowel, diphthong, nasal, and stop consonant. These are generated by chopping the air flow coming from the lungs into puffs of air, causing a semi-periodic waveform. The other broad class of sounds is unvoiced speech, including unvoiced fricatives and unvoiced stop consonants. These are generated without vocal chord vibrations. They are caused by a turbulent air flow which creates a waveform with noisey characteristics [18].

To further group phonemes we must consider the location of the narrowest constriction along the vocal tract during sound production. For example, different consonants are generally associated with different points of maximum constriction. The consonants */b/,/p/* are associated with a maximum point of constriction at the lips, while */t/, /d/* have points of maximum constriction behind the teeth. The categories of vowel, diphthong, and semivowel are caused by different constrictions of the tongue. For example the sound */iy/* is produced by the high position of a tongue hump, while the sound */ae/* is formed by a low tongue hump.

2.3 The Pattern Recognition Approach to Speech Recognition

Continous speech recognition systems today have some of the same basic characteristics of systems developed for other pattern recognition problems, such as feature extraction, an optional feature space reduction, and a trainable recognizer. The recognition stage generally involves combinations of several models, for sub-word units which are used to determine the most likely sub-word unit, word level models which combine the results of the previous model to generate the most likely sequences of words. Finally a sentence model can be used to determine the most likely sequences of words.

Raw speech is generally sampled at a high frequency, most commonly 16 kHz or 8 kHz; this produces a very large number of data points. Furthermore this raw speech, essentially giving the power levels over time, provides little distinguishing information about the utterance [11]. The first stage of the speech recognition process, the feature extraction, takes the raw digital speech signal, which consists of a large number of values per second and converts it into what is described as frames. This compresses the data by factors of 10 or more and extracts the useful information about the speech signal. A diagram of this process is shown in Figure 2.4. Some of the most relevant feature extraction techniques will be discussed.

Larger feature vectors can often greatly increase training and recognition times

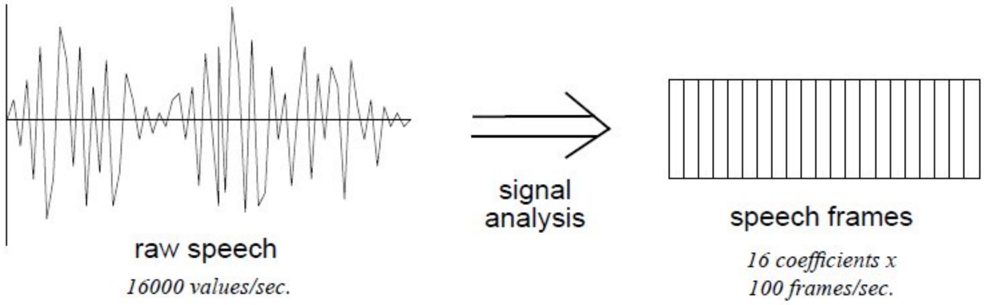

Figure 2.4 *Feature extraction process as shown in [3]*

for typical recognizers, such as the HMM, making them impractical. Furthermore redundancies within the data can cause worse performance in the recognizer. For this reason feature space reduction, used in many systems, attempts to reduce the feature vectors to a smaller set of features, while losing the least information possible. Common techniques for performing this reduction are linear discriminant analysis (LDA), Fisher's discriminant analysis (FDA), and principal component analysis (PCA). These methods generally attempt to find a transformation of the feature space which optimizes the separation of the classes.

Generally the recognizer attempts to segment the input speech and recognize each input feature as part of a phoneme, the smallest phonetic unit. However, many variants of this exist such as syllable and word-level classification. The output of the recognizer will generally be a string of phonemes or in some cases strings of words or syllables. For phoneme and likely for syllable recognition the next phase of the system involves taking these outputs and using a model of connections between phonemes to determine the most likely sequence of words. A language model can further help refine the results at the word level. It should be noted that this view is simplistic and some systems can be structured quite differently.

2.4 Feature Extraction

This section describes several popular feature extraction methods including the ones used for this work. For each of these methods the speech signal is divided into frames of length 10-30ms depending on the method. The technique is then applied to each slice of speech signal resulting in an output as demonstrated in Figure 2.4. Three of the techiniques: Mel frequency cepstral coefficients (MFCC), short time fourier transform (STFT), and the gammatone filterbank with ERB scaling are of particular interest when discussing invariance in the frequency domain. They are all time-frequency analysis methods that are closely related.

2.4.1 Mel Frequency Cepstral Coeffients (MFCC)

This feature extraction method is based on auditory models which approximate human auditory perception. It is the most widely used feature extraction method. This method is summarized in Figure 2.5. The DFT is applied to the speech signal and the phase information is removed because pereceptual studies have shown that the amplitude is much more important. The logarithm of the amplitude spectrum is computed because the perceived loudness of a signal has been found to be approximately logarithmic. If the inverse fourier transform is now taken, the resulting components are the cepstral coefficients which have important applications in different areas of signal processing. However perceptual modeling has found that some frequencies are more important based on the mel scale. Thus the next step is to smooth the signal based on the mel scale.

A simple example demonstrates how to rescale the spectrum on the mel scale. Rescaling the signal in terms of other auditory frequency scales is analogous. Let's assume we start with 64 values for the log of the amplitude DFT coefficients. We

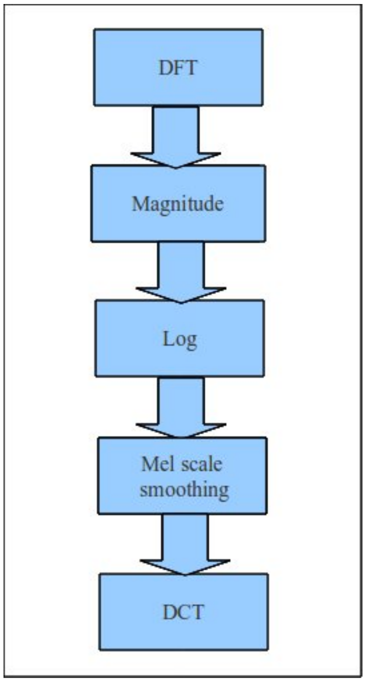

Figure 2.5 *MFCC feature extraction*

subdivide these into 8 bins. Let's assume the mel scale range is between 0-8000 mel. The bins on a linear scale will then range 0-1000,1000-2000,...,7000-8000 mel. We now convert these bins to Hz to determine the start and end of the frequency bin, this is done based on the mel scale shown in Figure 2.6. For example 0-1000 mel becomes 0-1000 Hz whereas 1000-2000 mel becomes 1000-3400 Hz. As we can see using this binning process the lower frequency coefficients become more important as the frequency bins at higher frequencies become larger. Generally to compute a single representative component for each bin the values in each bin are averaged using triangular filters, which emphasize the central frequencies in the bin more than the frequencies at the edge of the bin. Eventually we will have 8 values for each bin. We take the discrete cosine transform (DCT) to decorrelate the signal components, this produces what is known as the Mel frequency cepstral coefficients (MFCC) [19].

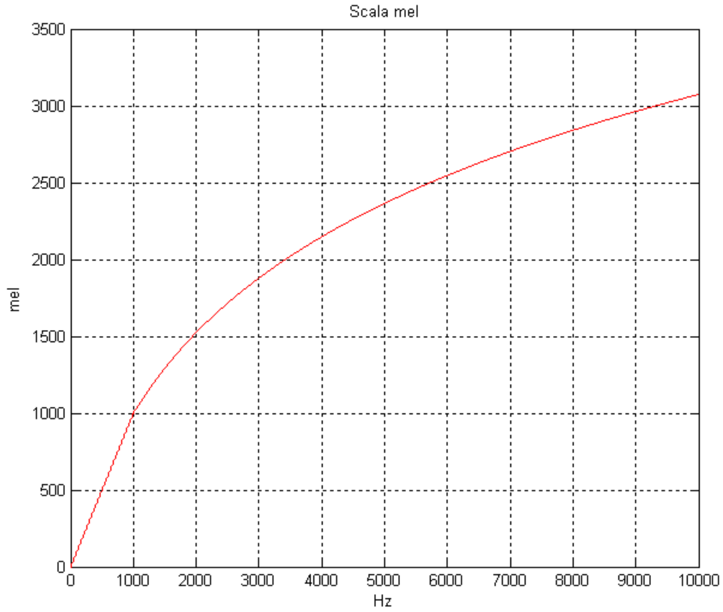

Figure 2.6 *Mel Scale [4]*

2.4.2 Linear Predictive Coding (LPC)

The method of LPC provides a model the action of the vocal tract upon the excitation source. As seen in Figure 2.7, the model involves the excitation source, $u(n)$, either periodic as in the case of voiced utterances or noise as in the case of unvoice utterances. We model the vocal tract filtering as an all-pole filter $H(z)$ with coefficients $a_1..a_p$. Thus given the speech sample at time n, $s(n)$, can be approximated as a linear combination of the past speech signals plus the excitation.

$$s(n) = -(a_1 s(n-1) + a_2 s(n-2) + ... + a_p s(n-p)) + u(n) \tag{2.1}$$

Here the coefficients $a_1, ..., a_p$ can be seen as the coefficients of a digital filter. This means that the speech signal can be seen as a digital filter applied to the source sound coming from the vocal chords and scaled by a gain as seen in Figure 2.7. The parameters of this model are commonly solved for using the Levinson-Durbin algorithm [20]. These coefficients give a condensed representation of the speech signal.

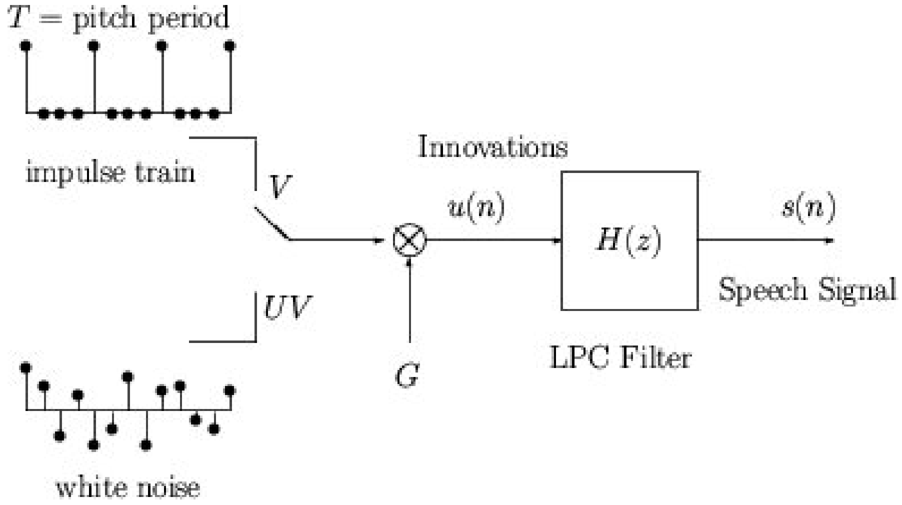

Figure 2.7 *A diagram of the linear predictive coding (LPC) model. Here the branch V refers to the voiced utterances, while branch UV refers to unvoiced utterances from the vocal chords. The filter we model, $H(z)$, represents the vocal tract [1].*

The frequency domain of the $H(z)$ filter often has peaks, at representative frequencies, called formants which encode key speech information. This makes this feature extraction technique a popular one for applications in speech encoding and compression [1].

2.4.3 Short-Time Fourier Transform (STFT)

The classic method of obtaining a time-frequency representation of a speech signal is called the short-time fourier transform (STFT). The STFT is defined as

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.2}$$

where $X(m, \omega)$ is the STFT and $x[n]$ is the speech signal. Here w is the rectangular windowing function, m is the time index of the STFT frame, n is the sample time, and ω is the frequency bin. A more intuitive way of looking at this technique is noting that this method applies a DFT on successive, generally overlapping, frames

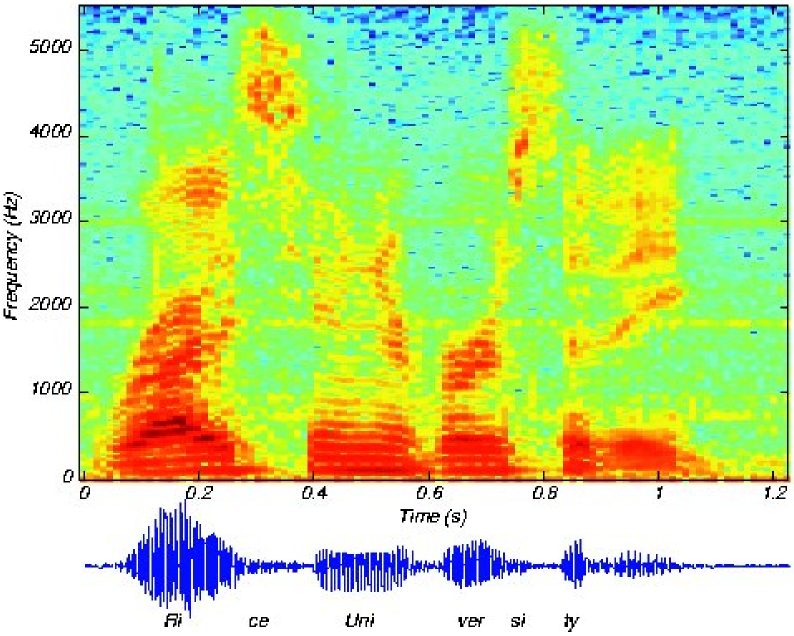

Figure 2.8 *Spectrogram of a speech signal [5]*

of the speech signal. The result of applying this transformation can be seen using what is called a spectrogram as shown in Figure 2.8 [21]. In the figure we can see the articulated speech in the time domain

The STFT has a great deal of limitations in speech recognition applications. The primary issue is that the STFT linearly distributes the frequency bins. This is in contrast to the mel scale described previously or the ERB scale which will be described, the STFT gives the same amount of resolution and weight to high frequencies and low frequencies. This causes a lot of information to be of little consequence, since the primary parts of the speech signal are contained within the lower frequencies. For this reason the STFT is not popularly used; however it is a good example of a time-frequency representation, and this method can help us understand the advantages of the gammatone filter bank model with the ERB scale.

Another way to view the STFT is as a set of ideal bandpass filters applied to the speech signal. This interpertation can help us better understand the gammatone filterbank model which is the primary feature extraction method used in this work. We can obtain this interpertation by regrouping the terms in the STFT as follows,

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} [x[n]e^{-j\omega n}]w[n-m] \tag{2.3}$$

if we define $x[n]_k = [x[n]e^{-j\omega_k n}]$ then the STFT becomes

$$X(m,\omega) = [x_k * Flip(w)](m) \tag{2.4}$$

We can interpert this equation as a filter bank. For each frequency bin, k, the signal is shifted down in the frequency domain so that the frequencies at w_k are at baseband, this gives $x[n]_k$. Then the signal is convolved with the low-pass filter defined by the reverse of the windowing function, $Flip(w)$, thereby producing a series of filter bank outputs for various values of k. A graphical interpertation is shown below in Figure 2.9 [6].

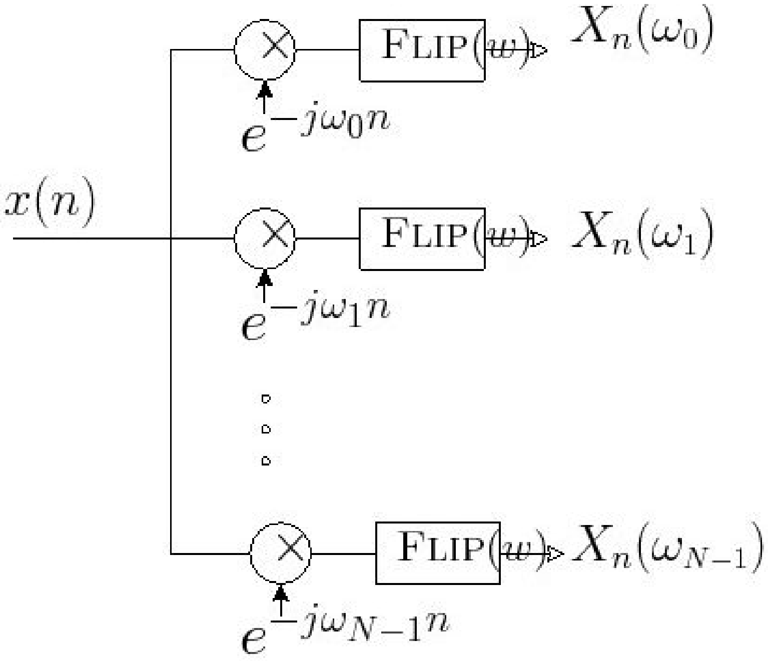

Figure 2.9 *Filter bank model of the STFT from [6]*

2.4.4 Gammatone Filter Bank and the ERB Scale

The gammatone filter is a popular approximation to the filtering performed by the human ear. In some works from physiologists the following expression is used to approximate the impulse response of a primary auditory fibre.

$$g(t) = t^{n-1}e^{-2\pi bt}cos(2\pi f_0 t + \phi) \tag{2.5}$$

where n is the order, b is a bandwith parameter, f_0 is the filter centre frequency and ϕ is the phase of the impulse response [22]. One way of thinking about this function is by noting that the first part is the gamma function from statistics and the cosine term is a tone when the frequency is in the auditory range. Thus this can be thought of as a burst of the centre frequency of the filter enclosed in a gamma shaped envelope.

Going back to our filterbank analysis of the STFT we can think of this gammatone filter as a replacement to the rectangular filters of the STFT. Unlike the triangular filters described for the MFCCs, these filters are based on physiological functions of the ear.

A bank of gammatone filters is commonly used to simulate the motion of the basilar membrane within the cochlea as a function of time. The output of each filter mimics the response of the memberane at a particular place. The filterbank is normally defined in such a way that the filter center frequencies are distributed across frequency in proportion to their bandwith. The bandwith of the filter is determined using the equation for an ERB, also derived using physiological evidence in [23] to be:

$$ERB = 24.7(4.37 \times 10^{-3} \times f + 1) \tag{2.6}$$

Thus the higher the center frequency the bigger the bandwith of the filter.

Similar to the Mel scale in the MFCC the ERB scale is used to accentuate the more relevant frequency bands and suppress the less important ones. In [24] it was shown that the gammatone filterbank set of features has superior performance compared to MFCCs for invariant speech applications.

2.4.5 Time Derivatives

One of the major problems with creating phoneme recognizers is modeling the time dependencies of adjacent frames, phonemes, and words. As we will see the use of HMM addresses this issue and allows for a powerful model of the phonemes connection to the previous phoneme. Another, generally supplementary, common step in ASR systems is to attempt to incorporate information about the time transitions between frames into the feature extraction stage. This is done by calculating the time derivative of the feature set and sometimes the second-order time derivative and attaching these derivatives on to our general feature set [17].

For time-frequency analysis methods these derivatives can be particularly important as was demonstrated in an experiment discussed in [17]. Using isolated syllables truncated at initial and final endpoints, it was shown that the portion of the utterance, where spectral variation was locally maximum, contained the most phonetic information in the syllable.

In ASR applications we will generally estimate the time derivative information using a polynomial approximation. For a sequence of frames $C(n)$, we approximate the signal as $h_1 + h_2n + h_3n^2$. We choose a window of 2M frames so that $n = -M, -M+1, ..., M$. The fitting error becomes $\sum_{-M}^{M} (C(n) - (h_1 + h_2n + h_3n^2))^2$. It can be shown that the following values minimize this error

$$h_2 = \frac{\sum_{n=-M}^{M} nC(n)}{T_m} \tag{2.7}$$

$$h_3 = \frac{T_m \sum_{n=-M}^{M} C(n) - (2M+1) \sum_{n=-M}^{M} n^2C(n)}{T_m^2 - (2M+1) \sum_{n=-M}^{M} n^4} \tag{2.8}$$

$$h_1 = \frac{1}{2M+1}\left[\sum_{n=-M}^{M} C(n) - h_3 T_m\right] \tag{2.9}$$

Where T_m is given by,

$$T_m = \sum_{n=-M}^{M} n^2 \tag{2.10}$$

The polynomial approximation to the second derivative can be obtained similarly as described in [17]. These time derivatives, typically refered to as "delta"(Δ) and "delta delta"($\Delta\Delta$), are a standard feature used to supplement MFCCs and other features in many works [9, 13, 15, 18, 25, 26].

2.5 Recognition

In this section we will discuss various methods of recognition that are used in typical speech recognition systems. First hidden Markov models (HMMs) will be discussed and their limitations will be outlined. Neural networks in general and in the context of speech recognition will be discussed in a separate section.

The most popular method used in speech recognition is that of hidden Markov models. This method, although imposing some of the necessary temporal constraints on the acoustic model, suffers several well known weaknesses which will be discussed. One key weakness is the HMM's discriminating ability. The HMM is generally trained with the maximum likelihood (ML) criterion which maximizes the likelihood that a given observation sequence was generated by a given model. During training this does not propogate any information to competing models about what observations were not generated by the competing model.

For this and other reasons, methods which have shown strong discriminative ability and have proven successful in other fields have become popular. Primarly recent research has focused on neural network based methods although support vector machine (SVM) and kernel based methods are also being explored [27,28]. Newer approaches generally combine neural networks and HMMs or HMMs and SVMs to make use of the HMMs sequential modeling while obtaining better discriminating ability. In addition, some recent works in recurrent neural networks(RNNs) have been able to accomplish state of the art results in phoneme recognition without incorporating HMMs [25].

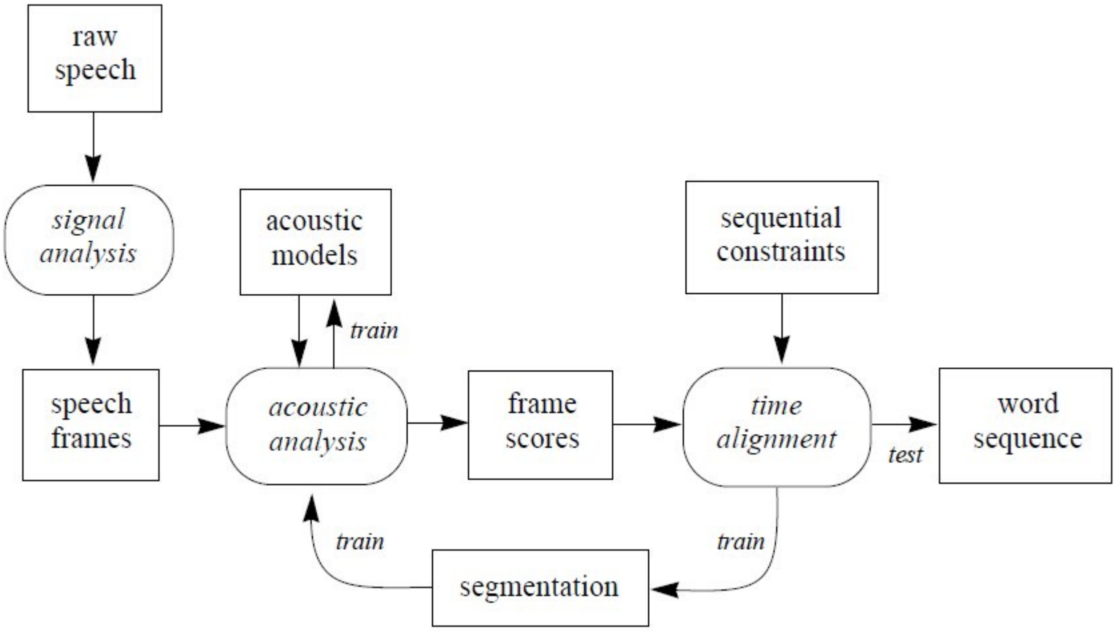

Figure 2.10 *Diagram of a continous speech recognition system from [3]*

Figure 2.10 shows a diagram of a typical speech recognizer. After the input has been processed with the signal analysis and feature space reduction methods described, the next phase can be grouped into a recognizer. The recognizer has been trained prior to the system's use to categorize the input features and perform segmentation. Unlike a classification problem, a continous speech recognizer is more complicated since it needs to segment each sound and determine the most likely sequence.

An acoustic model will usually model speech as a sequence of states. The states can vary in granularity. For example, a state for each word will have the largest granularity and would give the greatest word recognition accuracy. However, the number of examples of each word available will be too small for effective training. On the other end of the granularity scale a monophone model will model a sequence of phonemes. The acoustic analysis will yield scores for each frame of speech. Within HMMs, these scores generally represent emission probabilities, referring to the likelihood that the current state generated the current frame. The time alignment processing then identifies a sequence of the most likely states that produce a word in the vocabulary. For HMMs this time alignment is generally accomplished with the Viterbi algorithm. The time alignment generates a segmentation on the sentence, which can be used during training to train the appropriate acoustic model on corresponding segmented utterances [3].

2.5.1 Hidden Markov Models

Our goal in the recognition stage is generally to determine which category of phoneme, syllable, or word the current frame or observation falls under. We can base the decision purely on properties of the speech within the current time window, but this would preclude us from using the intrinsic temporal properties in speech as well as sequential constrains. Speech utterances have a strong sequential structure. Subword utterances tend to come in specified sequences. Similarly words tend to follow sequential constraints defined by a grammar. In order to recognize speech we must construct an adequate model of these sequential constraints.

A hidden Markov model is a statistical model which is heavily used in speech recognition due to its abilitiy to do just this, model the temporal transitions between states. To understand Markov models, consider a system in which we use all previous observations and the current observation to make our decision on the classification category. Such a system would be impractical. For many applications, in particular speech, the most recent observations are much more important than previous observations. A model which assumes dependencies only on the most recent observations is called a Markov model [29]. In practice, we generally use a first-order Markov model in which the current observation is dependent only on the previous observation. One can represent this as a state model with edges representing transition probabilities between states.

Within a standard Markov model each variable has a probability of transitioning to the next state. The basic Markov model has each state "corresponding to a deterministically observable event" [17]. This model alone is not good enough to be applicable to many problems since we often do not know what state we are in given an observation. Hidden Markov models serve to resolve this discrepancy.

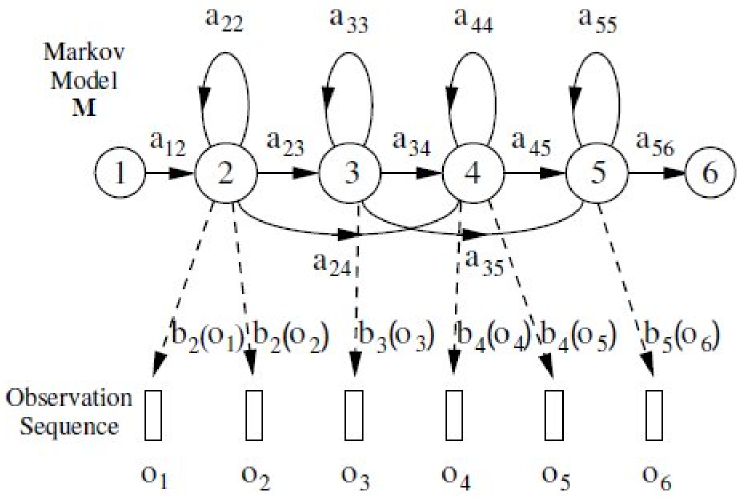

Figure 2.11 *Model of HMM from [7]*

In a hidden Markov model (HMM) there exists another set of parameters, which generally characterize the actual probability of being in a specific state. The HMM model consists of transition variables also called, transition probabilities, and latent variables also called emission probabilities. A diagram of an HMM is shown in Figure 2.11. Here we have the observed feature sequence o_t generated from the probabilities $b_j(o_t)$ or b_{jt} for short, which are the emission probabilities, and also the transition probabilities denoted by a_{ij}. As shown in the diagram, if we are at state 1 there is a certain probability of transitioning to the next state, denoted by a_{1j}.

Mathematically an HMM model, λ, is given by $\lambda = (A, B, \pi)$. Here A is the set of all a_{ij}, the probabilities of transitioning from state i to state j. B is the set of all b_{jt} the probability given state j of generating the observation at time t. Finally π, the initial state distribution, is the set of all π_i which are the probabilities that the initial state is i. It is interesting to note that the HMM is a generative model. Given the parameters of λ, a sequence of utterances can be generated.

Since the HMM is able to impose temporal constraints on a model it is common to construct hierarchial models from smaller models after determining the parameters of the smaller models. As will be discussed, this is critical to large vocabulary continous speech recognition. As shown in Figure 2.12 we can construct phoneme level models trained for each phoneme which can be concatenated to form word level models and sentence level models [3].

A simple example of how the HMM works can be obtained by discussing isolated-word speech recognizers. In this task we know that the input is a word and do not concern ourselves with the boundaries of the classification categories. We attempt to build an HMM for each individual word. Let us assume we have an HMM model constructed for each of M words in the vocabulary. The algorithm that can be used to obtain the HMM parameters will be discussed later. Given a sequence of observations $O = (o_1, o_2, ..., o_T)$ and a vocabulary $w_1, ..., w_m$ the problem of finding the most likely

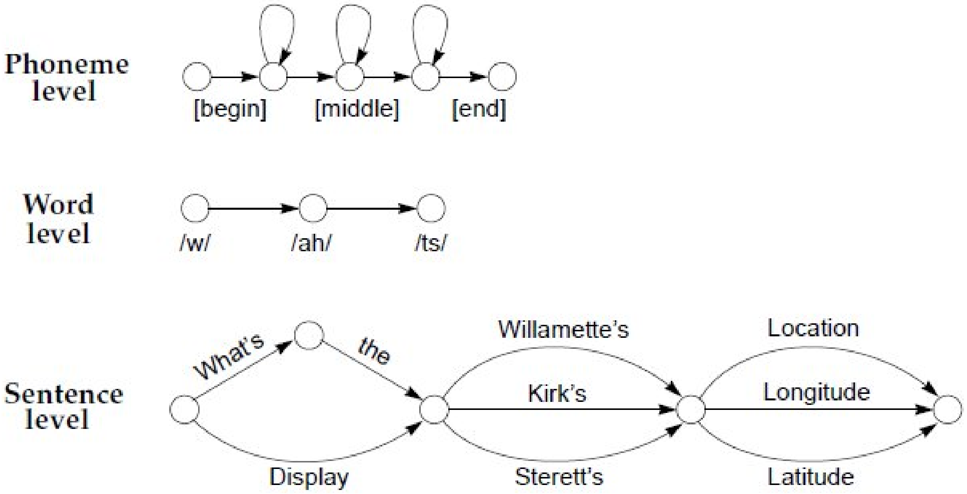

Figure 2.12 *Hierarchial model of HMMs [3]*

word, w_{ml} then reduces to finding:

$$w_{ml} = \underset{t}{\operatorname{argmax}} P(w_t|O) \tag{2.11}$$

which cannot be estimated directly. By applying Baye's rule we obtain:

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)} \tag{2.12}$$

Thus we can reduce the problem to maximizing $P(O|w_i)$. Assuming each word corresponds to an HMM model we attempt to find for each word model $\lambda_1, .., \lambda_M$ the $P(O|\lambda_i)$. Once this set of probabilities is estimated we can determine the most likely word that generated the sequence.

Here is how $P(O|\lambda)$ is calculated. Let's assume a fixed state sequence q representing one of the possible state sequences of length T. HMMs assume the observations are independent, thus we can say the probability of the observation given the model and the sequence q is related to the individual observations at time t by:

$$P(O|q, \lambda) = \prod_{t=1}^{T} P(o_t|q_t, \lambda) \tag{2.13}$$

$$P(O|q,\lambda) = b_{q_1}(o_1)b_{q_2}(o_2)...b_{q_t}(o_t) \tag{2.14}$$

The probability of this state sequence is given by:

$$P(q|\lambda) = \pi_{q_1}a_{q_1q_2}a_{q_2q_3}...a_{q_{T-1}q_t} \tag{2.15}$$

With these results we can obtain the joint distribution:

$$P(O,q|\lambda) = P(O|q,\lambda)P(q|\lambda) \tag{2.16}$$

Summing over all possible q we obtain:

$$P(O|\lambda) = \sum_{all\ possible\ q} P(O|q,\lambda)P(q|\lambda) = \sum_{all\ possible\ q} \pi_{q_1}b_{q_1}(o_1)a_{q_1q_2}...a_{q_{T-1}q_t}b_{q_t}(o_t) \tag{2.17}$$

[17] gives an intuitive explanation of the above equation. For each possible state q, initially (at time $t = 1$) we are in state q_1 with probability π_{q_1} and generate the symbol o_1 with probability $b_{q1}(o_1)$. The clock moves to $t + 1$ and we make the transition to state q_2 from q_1 with probability $a_{q_1q_2}$ and generate symbol o_2 with probability $b_{q_2}(o_2)$. The process continues in this manner until we reach the last state T.

Unfortunately this direct computation is infeasible due to the large number of calculations involved. Fortunately there exists a recursive algorithm for computing this called the forward procedure. This procedure defines a forward variable:

$$\alpha_t(i) = P(o_1, o_2...o_t, q_t = i|\lambda) \tag{2.18}$$

This gives the probabilitiy of observing the partial sequence $o_1, o_2...o_t$ and ending up in state i at time t. This is related to our desired probability by:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.19}$$

The forward algorithm computes the forward variable at T using a recursive relation:

$$\alpha_1(i) = \pi_i b_i(o_1)\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(o_{t+1}) \tag{2.20}$$

Thus we have a method to obtain the value $P(O|\lambda)$ [17].

We have learned how one can decide which model best fits each word in an isolated word recognition problem. But, how do we determine the parameters of the model in the first place? We will briefly discuss the most popular algorithm used for training. To estimate the model parameters for an HMM, $\lambda = (\pi, A, B)$ the most commonly used technique is called Baum-Welch reestimation (which is also known as the forward-backward algorithm). This is an iterative method which actually makes use of the forward algorithm along with a sister calculation, the backward algorithm as part of its estimation proces. It is a particular case of a generalized expectation-maximization (GEM) algorithm. This method can compute maximum likelihood estimates and posterior mode estimates for the parameters (transition and emission probabilities) of an HMM, when given only emissions as training data [3, 29]. Baum-Welch reestimation, starts with an intial guess for the model parameters. At each iteration we obtain the maximum likelihood (ML) estimate for the model using the efficient backward and forward procedures. We can then use these new estimates for the model parameters in the next iteration. The procedure continues until a stoping criterion is met. For continous speech, the method for isolated word recognition described above is not sufficient because it is not practical to have separate HMM models for each sentence. Another key algorithm with regards to the training of HMMs is used for decoding in the continous case; this is the Viterbi algorithm. This algorithm deals with the problem of finding the best state sequence q given a model λ that best explains an observation sequence O. For isolated word recognition this algorithm can be used to segment each of the word training sequences into states, then study the properties of the spectral vectors that lead to the observations in each state. This allows us to make refinements to the model before it is used. Since Viterbi allows us to find a best sequence we can use this algorithm for the recognition phase in continous speech recognizers. The states in the models are given more physical significance, allowing them to belong to particular words or phonemes. The Viterbi algorithm then attempts to find the best sequence of states in the model. The Viterbi algorithm attempts to find the maximum likelihood state sequence. We define a likelihood variable, $\phi_t(i)$, denoting the likelihood of observing $o_1, o_2 ... o_t$ and being in state i at time t:

$$\phi_{t+1}(i) = \max_i \{\phi_i(t) a_{ij}\} b_j(o_{t+1}) \tag{2.21}$$

The likelihood is computed in a similar fashion to the forward algorithm except that the summation of previous states is replaced by taking the maximum state. The value is initialized with:

$$\phi_1(1) = 1 \tag{2.22}$$

$$\phi_j(1) = a_{1j} b_j(o_1) \tag{2.23}$$

The maximum likelihood then becomes:

$$\phi_N(T) = \max_i \{\phi_i(T) a_{iN}\} \tag{2.24}$$

where T is the last time step and N is the final state. This algorithm is often visualized as finding the best path through a matrix were the vertical dimension represents states of the HMM and the horizontal dimension represents the frames of speech [7].

For large vocabulary continuous speech recognition the HMM will generally be based on phonemes, because the large number of words and their different pronunciations make it highly impractical to train and compare against that many HMM models. For this reason most large-vocabulary continous speech recognition is done at the phoneme level [3].

HMMs have several weaknesses that need to be addressed in order to improve continous automated speech recognition. One problem is, given that we have arrived at a particular state within an HMM, the likelihood that an HMM will generate a certain observation is dependent only on the current state, which is not a valid assumption for speech where the distribution is strongly affected by recent history. Another closely related problem is the independence assumption. HMMs assume that there is no correlation between adjacent input frames, which is false. We have seen one way to deal with this by incorporating time derivatives into the frame so that each state will get a better temporal context.

Models for the HMM states can be discrete or continous. The most commonly used model of continous Gaussian mixtures is not optimal. It assumes a particular form of the distribution. This problem is one of the inspirations for the frame based hybrid HMM and neural network systems. Here neural networks are trained to generate the emission probabilities without making any a priori assumptions about the distribution of the data (unlike a Gaussian mixture model).

Finally, as discussed earlier, the maximum likelihood traing criterion leads to poor discrimination between acoustic models. As we will see in neural network training for each example the network not only learns that example A belongs to class X, but also that example A does not belong to any of the other classes. An alternate training criterion for HMMs has been explored called the maximum mutual information (MMI) criterion; however this has been difficult to implement properly and greatly increases complexity [3].

2.5.2 Scaling of Speech Recognition

It may become confusing to the reader when we discuss the different types of phoneme, word, syllable recognition approaches. Some clarification is needed in this area. As mentioned in the previous section word-level recognition in which we directly attempt to classify the speech into categories of words becomes impractical for large number of words. For example, the average word has many different pronunciations; if we have a vocabulary of even 1000 words, we will need 1000 HMMs or 1000 output states in a neural network. This becomes computationally infeasible very quickly, especially given the amount of training data required for each different pronunciation of the word.

In order to remedy this, phoneme and syllable recognition are studied. Recognition at this level can then be combined into efficient word and sentence recognizers. The problem with phoneme level recognition is that phonemes are very sensitive to context. As mentioned previously the phoneme is only a layover of the most basic speech utterance, the phone. The phones suffer from the same problem as words, there are too many possible phones. However by combining many phones into phonemes which are indistinguishable to the listener, we introduce a great deal of context-variability. Syllables do not suffer from such a problem; however they are not widely used because there is simply too many of them for practical systems [17]. Thus most continous speech recognizers work at the phoneme level. It is also common to construct biphone and triphone models consisting of combinations of two and three phones [3].

There are several ways to get from phonemes to actual word and sentence recognition. A brief example using HMMs is to create HMMs for each phoneme, a standard set of English phonemes contains 39 different categories. These models can then be combined into words using a lexicon of phoneme to word transcriptions. Such a lexicon will have ambiguity since one word can have different pronunciations. To resolve this, language modeling techniques can then be used to better decide which word best fits the current sentence [17].

The field of research in speech recognition is quite varied. The problem being attacked from various scales. Due to this, research in discriminative techniques often begins with a simpler problem then research dealing with language models and how best to incorporate them into large vocabulary continous speech systems. Since there are various techniques which use the phoneme information to generate the actual sentences of the speech recognizer output, many authors focus on the task of phoneme classification and phoneme recognition in order to isolate their methods and make their performance metrics simple and more objective [9, 13, 17].

In the phoneme classification task we focus on discriminating between different phonemes. Here the segmentation of the utterances is known and they are presented to a classifier. In the more sophisticated task of phoneme recognition a sentence is presented and the sequence of phonemes is output by the system. In classification the metric is quite clear to construct; namely, divide the number of correct classifications by the number of total utterances given to get a classification score. In the recognition task the score generally contains a penalty for insertions and deletions of phonemes besides the simple penalty of whether the phoneme was classified correctly.

Since this work focuses on improving the invariance in the recognition stage, we will also simplify our task to that of phoneme classification. Extensions into phoneme recognition will be discussed in section 4.2.

2.5.3 Neural Networks

Neural networks are biologically inspired classifiers. They are heavily used for many machine learning applications due to their abilitiy to learn non-linear functions. They contain potentially large numbers of simple processing units, similar to neurons in the brain. All the units operate simultaneously, allowing for algorithms to be implemented efficiently on specialized hardware supporting parallel processing [3, 30]. In recent years there has been a great deal of research into applying neural networks for speech recognition applications. Although hidden Markov models are very popular and perform well by incorporating the intrinsic time dependencies of speech into the model, their ability to actually distinguish categories based on the current observation is lacking. On the other hand, neural networks have a strong discriminating ability but traditionally lack the ability to model temporal transitions. For this reason neural networks are often used in combination with HMM models, or extra features are built into the networks to help them incorporate time information such as in the time-delay neural network (TDNN) structure and the recurrent neural network (RNN) [8].

There are several basic features that are shared by all types of neural networks. These features are:

- A set of processing units generally refered to as nodes
- A set of connections, or links, between these processing units
- A computing procedure performed by each node
- A training procedure

The nodes in a network are generally labeled as input, hidden, or output. Input nodes receive the input data, hidden nodes internally transform the data, and output nodes represent the decisions made by the network. At each moment in time, each node computes a function of its local input, and broadcasts the results to its neighboring nodes in the succeeding layer [3].

2.5.3.1 Frame, Segment, and Word-Level Training

Speech recognition systems using neural networks can often be categorized by the unit of speech that is used for training. Frame-level training refers to training on a frame by frame basis. An example of this is an HMM and neural network hybrid system in which the neural network is trained to estimate the emission probability used for the HMM. The problem suffered by such systems is the lack of context information. Often a phoneme cannot be categorized by a single frame of short duration [3].

Alternatively segment-level training, which will be the focus of our work here, receives input from an entire segment of speech. Generally this is the whole duration of a phoneme, but can also be a syllable. This allows much more information about the correlation that exists among all the frames to be presented to the network. The difficulty in this method is that the speech must first be segmented before it is evaluated by the neural network. An approach which incorporated a segment-level training in a full continous speech recognition system was demonstrated in [31]. Here an HMM recognizer was used to produce segmented versions of the most likely N sentences, where N was chosen as 20. A segment trained fully connected network was then used on each segment to produce a new set of likelihood scores for each sentence. This was combined with the HMM scores and used to decided on the most appropriate sentence.

In word-level training we segment the input speech into entire words to obtain the maximum context for the neural network input. Unfortunately the problem with this method is the word cannot be easily modeled with one output state. For large-vocabulary systems the number of output words would grow to unreasonable proportions. As has been discussed in the HMM section, training would require far more examples since the number of example words in a given training set is far smaller then the number of phoneme examples.

2.5.3.2 Multi-Layer Perceptrons

The standard neural network called, the multilayer perceptron, consists of a feedforward network of computing units. A diagram of a neural network is shown in Figure 2.13. Each connection has a weight value which is multiplied by the input to the node. The weighted sum of inputs from all the connections is then passed through an activation function. Mathematically we can think of the output of a node in the network, a_i as:

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i}a_j\right) \tag{2.25}$$

where $W_{j,i}$ is the weight of the link between node j to node i. The function, g, is refered to as the activation function. The purpose of the activation function is to allow the network to model non-linearity between input and output. It has been shown that with a single hidden layer, it is possible for a neural network to represent any continuous function of its inputs, given a sufficiently large amount of hidden nodes [30]. One popular choice for the activation function is a sigmoid function. Sigmoids are functions that asymptote at some finite value as the input approaches positive or negative infinity. The most commmonly used sigmoids are the hyperbolic tangent and the standard logistic function [32]. In his work [32], Lecun recommends the use of the hyperbolic tangent to improve speed of training with backpropogation. The advantage of this is that a sigmoid that is symmetric about the origin is more likely to produce outputs that are on average close to zero. This is optimal for the gradient descent algorithm which will be discussed later.

The simplest way to apply a neural network to speech recognition is to present a sequence of acoustic feature vectors at the input to the neural network and detect the most probable speech unit at the output [8]. An example of this taken from [8] for small vocabulary isolated word recognition is shown in Figure 2.13. The desired outputs here are 1 for the nodes representing the correct speech units, and 0 for the incorrect ones. In this way, not only is the correct output reinforced, the wrong outputs can be weakend. Because the wrong outputs are weakened for each example, the multilayer perceptron becomes capable of better discrimination than the hidden Markov model [8]. The same approach can be applied to phoneme recognition for more general automatic speech recognition systems.

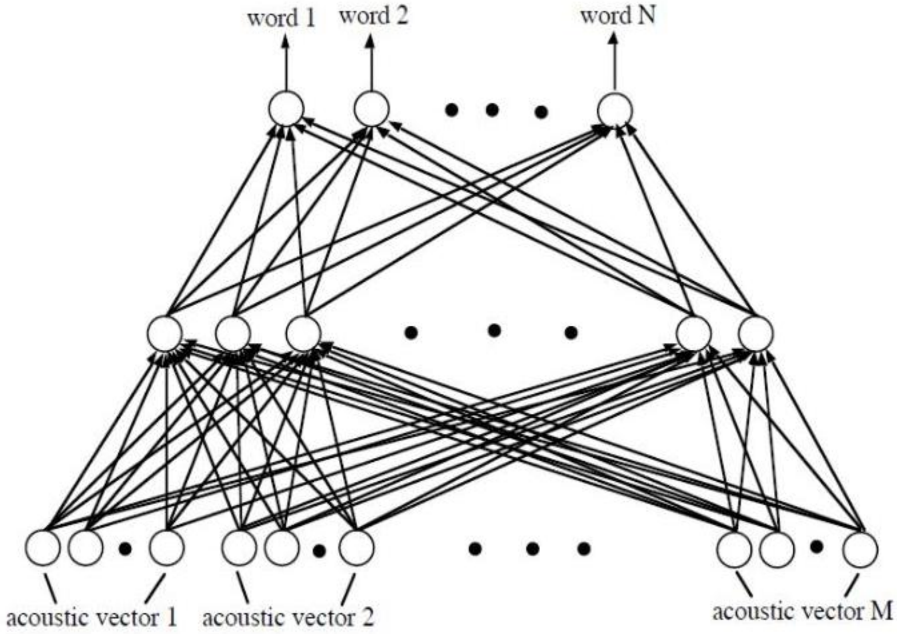

Figure 2.13 *Sequence of M acoustic feature vectors being applied directly to a fully connected neural network [8]*

The problem with this network structure is that it does not incorporate any temporal information into the network. Although in theory the network could learn all the temporal information that is needed if the window M is large enough, the reality is that gradient descent training, which will be discussed does not guarantee this knowledge will be explicitly incorporated. Structuring the network to force temporal relationships between nodes is a more suitable strategy for incorporating temporal information. As we shall see the TDNN provides for an efficient way to incorporate temporal information. Other approaches are also possible such as recurrent neural networks which contain feedback connections, however the theory behind training these networks is still in early stages.

2.5.3.3 Time-Delay Neural Networks

In order to improve the ability of a neural network to deal with the temporal relationships between acoustic vectors, the time-delay neural network (TDNN) was proposed by Waibel [9]. In the time delay neural network each layer gets as input the current input as well as delayed inputs which are then used to calculate the output. One way of looking at this is shown in Figure 2.14 [9]. Here the input features denoted by U_1 through U_j are delayed in time and weighted essentially as a secondary sets of feature vectors.

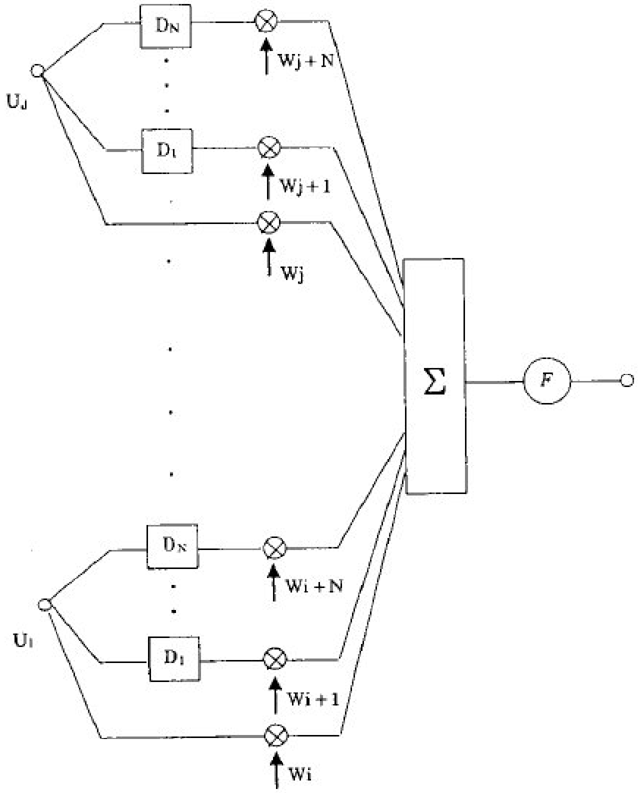

Figure 2.14 *The inputs to a node in a TDNN. Here $D_1, ..., D_N$ denote delayes in time of the input features $U_1, ..., U_j$ [9]*

An important aspect that the TDNN addresses is the proper representation of the temporal relationship between acoustic events. Another key aspect is that the network provides a level of invariance under translation in time. For example the specific movement of a formant (peak frequency in a speech pattern) is important for determining a voiced stop (a stopped consonant made with tone from the larynx while the mouth organs are closed at some point), but it's irrelevant if this event occurs a little sooner or later in the course of time [9]. This translation invariance in the time domain is important as it removes the need for a precise segmentation used to align the input pattern. Even if the current input into the neural network does not completely contain an entire phoneme, if the features of relevance are present at any time within the input stream and the confidence is high enough in the category, we will be able to recognize the existence of a phoneme. This alignment task is otherwise difficult in a continous speech recognition system such as one that uses a basic neural network; wherein it is unclear without alignment the borders between the next phoneme. In [26] it was shown that TDNNs can perform significantly better on misaligned phonemes then standard fully connected multilayer perceptrons.

Figure 2.15 shows Waibel's TDNN implementation. Here the feature vectors, MFCCs, are of length 16, with a total of 15 frames presented to the network. Here each frame is 10ms and the length of the delay of 3 frames is chosen from studies which show that 30ms is sufficient to represent low level acoustic phonetic events for stop consonant recognition [9]. There are 8 nodes in the hidden layer. The next layer uses a time window of 5 frames. The intent in this structure is to have the first hidden-layer learn the the short-time acoustic properties and the second layer learn the long term acoustic properties. The final hidden layer is referred to as an integration layer. This sums the "evidence" across time for the phoneme. This layer has a fixed weight applied to the sum of each row and connected to an output node with an activation function. It is worth noting that another interpertation of this network is that of 8 convolutional windows of length 16×3 applied to the input with

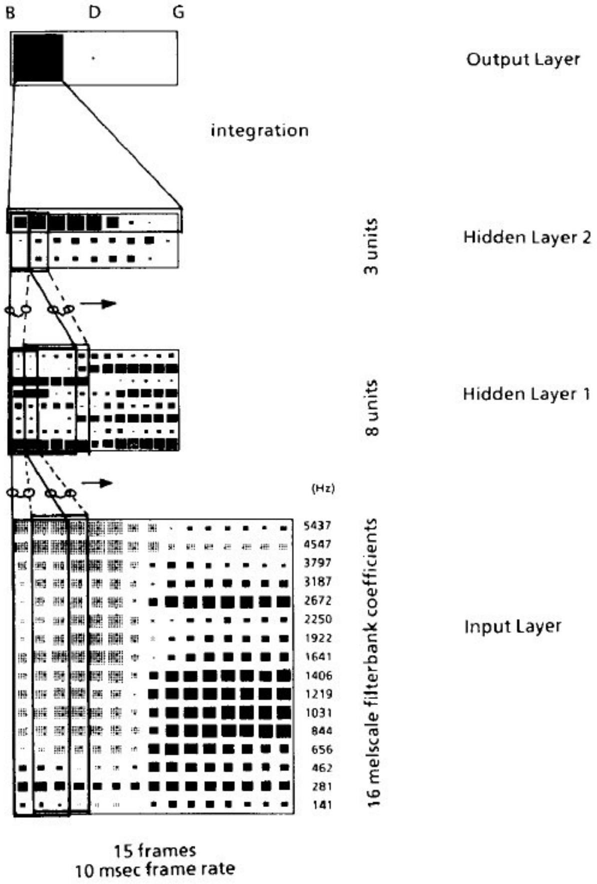

Figure 2.15 *Waibel's Time-Delay Neural Network structure [9]*

each convolutional windows output on a separate row in the second layer. The next layer can then also be interperted as a set of convolutions. This will help us interpert the TDNN within the context of a convolutional neural network.

Similar structures have been widely used in speech recognition since the introduction of TDNNs [14, 15, 18]. The most notable work for our purposes comes from [15], were a structure called block windowed neural networks (BWNN) is described. In this structure a window similar to the TDNN window is applied across time, the difference being that the window does not span the full length of the input in the frequency domain, thus the window is convolved in time and frequency. In this early work it was theorized that this structure would allow for the learning of global features and precise local features about both time and frequency data. This structure is essentially a convolutional neural network without subsampling layers and without the use of multiple convolution kernels and feature maps. Furthermore, the features used were MFCCs which, as described earlier, are not as well suited for visual representation of the speech as gammatone filterbank or the STFT. This work reported improved classificaiton acccuracy for various speech recognition tasks, however this type of network was not used in later work and uses of the TDNN networks have generally not involved windowing in the frequency domain [3, 18]. In this work, we attempt to improve upon this idea by applying all the features of a CNN, particularly the subsampling layer and multiple feature maps, in order to improve the invariance. Furthermore we attempt to exploit the visually distinctive structure obtained by gammatone-filter bank models of speech to allow the CNN to better learn local correlations.

2.5.3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs), designed for image recognition, are specialized types of neural networks which attempt to make use of local structures of an input image. Convolutional neural networks attempt to mimic the function of the visual cortex. In [33], Hubel and Wiesel found that cells in the cat's visual cortex are sensitive to small regions of the input image and are repeated so as to cover the entire visual field. These regions are generally referred to as receptive fields. Several models for image recognition have been created based on these findings; in particular, the work of Yann Lecun studied convolutional neural networks as applied to document recognition [34]. Lecun described convolutional neural networks as an attempt to eliminate the need for feature extraction from images [35].

A key problem with fully connected networks is that they ignore the spatial structure of the input image. The pixels of the input image can be presented in any order without affecting the outcome of the training [35]. In the case of images and spectral representations of speech there exists a local structure. Adjacent pixels in an image as well as adjacent values of a spectral representation have a high correlation. Convolutional networks attempt to force the extraction of these local features by restricting the receptive fields of different hidden nodes to be localized.

Another closely related feature of convolutional neural networks, and the one which is of particular importance in this work, is their invariance properties. A standard fully connected network lacks invariance with respect to translation and distortion of the inputs. Since each node in the hidden layer receives a full connection from the input, it is difficult to account for possible spatial shifts, although in principal a large enough network can learn these variations. This would likely require a large number of training examples to allow the network to observe all possible variations. In a convolutional neural network, a degree of shift invariance is obtained due to several architectural properties.

The convolutional neural network achieves shift and distortion invariance through the use of local receptive fields, shared weights, and spatio-temporal subsampling. The local receptive field allows the recognition to focus on localized structures versus learning only relationships between global structures. Local structures aren't restricted in their position within the input space. The shared weights allow for localized structures to exist in different parts of the image and still trigger neurons to fire in the next layer. Finally, the subsampling of hidden layers improves upon this invariance by further decreasing the resolution of the inputs to the next layer by averaging the result of adjacent nodes from the previous layer [34].

Figure 2.16 shows a modified version of the LeNet-5 network structure [34]. At the input layer, a kernel, which is a fixed size block of 9×3 weights, is applied to each point in the image. This is analogous to applying a 2-D digital filter to an image via a convolution operation, thus the name CNN. In theory, given appropriate training, a convolution kernel can become a well-known image filter such as an edge detection filter. For each kernel applied to the input image there exists a feature map which is the output of the convolution. In subsequent layers, feature maps can be connected in various ways to other feature maps. For example, two distinct feature maps in the first convolutional layer (C1) can be connected to the same feature map in the next layer; this entails applying 2 different kernels at localized points and then combining them at the relevent hidden node in the feature map of C2. A common way to describe this is through the use of a connection table, which is a table of size $N \times M$ of binary values, where N is the number of input feature maps and M is the number of output feature maps. The (i, j) entry indicates the presence or absence of a connection between the jth feature map in the higher layer with the ith feature map in the lower layer.

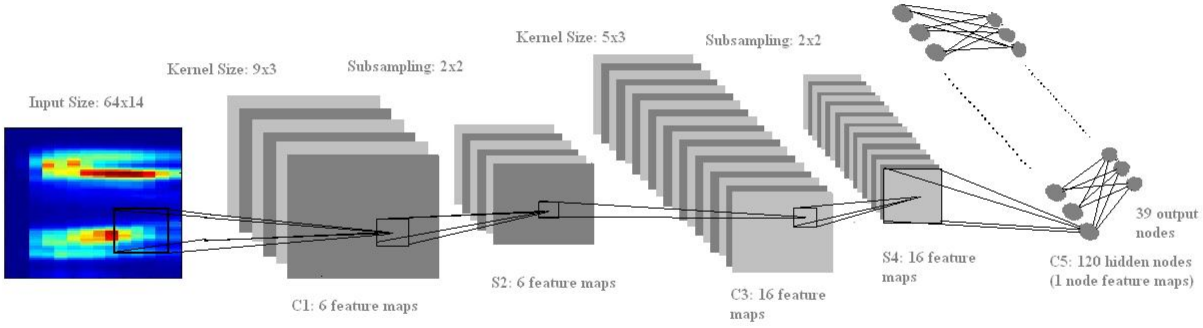

Figure 2.16 *Diagram of a modified LeNet-5 structure*

In this context, we can now formulate a TDNN as a subclass of CNNs. The TDNN can be interperted as a CNN without subsampling layers and convolution kernels spanning the full length of the input in the frequency dimension. Subsequent layers can be interperted as fully connected feature maps.

2.5.3.5 Limitations of Neural Networks

Several difficulties are present in this approach versus the HMM approach. One of the large difficulties is that within the hidden Markov model the time alignment can be performed automatically in the recognition phase by the viterbi algorithm. The second difficulty in this approach is time variability, the same word or phoneme from different speakers has different durations. Since the neural network has a fixed number of inputs, either some acoustic vectors have to be cut if the word/phoneme is too long or set to arbitrary values if it is too short [8].

As in past work [9], we will attempt to eliminate the alignment problem from our experiments by restricting ourselves to the phoneme classification task. Within this task it is assumed that the phonemes have been segmented and all that must be found is the category within which to classify the phoneme. The goal is to demonstrate the ability of the convolutional neural network to discriminate between phonemes of different speakers. These networks can then be the basis of larger systems which perform the segmentation task as in [31,36].

The time variability problem has been addressed by various researchers in several ways. Within the original TDNN structure the input size of the segment is fixed to 150ms. This can make recognition difficult for phonemes whose lengths are longer. In a follow up to the seminal paper [9], Waibel's [37] explored combinations of networks trained for different lengths with improved results for categories of different lengths. In her dissertation [18], Hou describes a method of combining a knowledge based categorization system which would first detect the general category of phoneme(fricative, consonant, vowel,etc) giving a much better idea of the phoneme length. This phoneme would then be run through a neural network classifier made for that particular category.

2.5.3.6 Training algorithms

Gradient Descent Back Propogation

Gradient descent is a classical optimization algorithm. Given a set of parameters, W, we seek to optimize a cost function, $E(W)$. Here W can represent the weight vector of a neural network. Gradient descent works by taking a step in the negative direction of the gradient of the function at the current point. More formally, at each time step we find a W that better minimizes $E(W)$ by computing $\frac{\partial E(W)}{\partial W}$ and then updating the W vector to,

$$W(t+1) = W(t) - \mu \frac{\partial E(W(t))}{\partial W(t)} \tag{2.26}$$

where μ is the stepping size, also known as the learning rate, and t denotes the iteration.

In order to implement this update procedure through multiple layers of a neural network, we need to approximate the gradient of the error, $\frac{\partial E(W)}{\partial W}$, with respect to all the weight vectors. This is easy to do for the weights connected directly to output nodes; however computing the components of the error with respect to weights which terminate at hidden nodes requires a procedure known as backpropagation. Backpropogation takes advantage of the chain rule by the following formulation:

$$\frac{\partial E_p}{\partial W_n} = \frac{\partial F}{\partial W}(W_n, X_{n-1})\frac{\partial E_p}{\partial X_n} \tag{2.27}$$

$$\frac{\partial E_p}{\partial X_{n-1}} = \frac{\partial F}{\partial X}(W_n, X_{n-1})\frac{\partial E_p}{\partial X_n} \tag{2.28}$$

Where X_n, is the output at layer n , W_n is the set of parameters used in layer n. The function $F_n(W_n, X_{n-1})$ is applied to the input X_{n-1} to produce X_n [32]. Solving this recursively we can obtain the desired $\frac{\partial E(W(t))}{\partial W(t)}$.

Training of convolutional kernels can be done by computing the error as if each application of the kernel was a separate set of weights. The errors for each application of the kernel can then be summed as in [34] or averaged as in [9] to create the overall weight update for the shared weights.

Batch Training Versus Stochastic Training

Equation 2.26 gives us a procedure for updating the weights once we have computed the gradient with backpropogation. There are two competing ways to use this gradient descent procedure. Batch training involves computing the error on the entire set of training examples, taking the average, and then performing the update procedure. In [9] and the more recent [18], a batch training approach was used to train a TDNN. The networks were trained on increasingly large subsets of the data to increase the speed of convergence.

An alternate approach popularized by LeCun [29, 32] trains on the error from each example as it presented to the network. This is known as stochastic gradient descent. Stochastic training is often preferred because it is usually much faster than batch training and often results in better solutions [32]. The problem with gradient descent lies in the fact that it is only guaranteed to find a local minimum. Stochastic descent introduces a great deal of noise into the training by using the current example as an estimate for the overall error. This noise can actually be advantageous, allowing the descent to venture out of local minimums. There are however ways of improving batch training as discussed in [32]. In general stochastic descent has been the more popular method because it is simply much faster.

Adapting The Learning Rate

In order to improve convergence speed it is common to choose separate learning rates for each weight, unlike the fixed μ in Equation 2.26. Some methods exist for determining this step size in each direction of the weight vector as well as continously adapting this learning rate as outlined in [32,34]. These methods can greatly increase the rate of convergence. A common way to adapt the learning rate, ϵ_k, for a specific weight, w_k, of the weight vector W is by use of the relation,

$$\epsilon_k = \frac{\eta}{\mu + h_{kk}} \tag{2.29}$$

Here μ and η are hand picked parameters. h_{kk} is an estimate of the second derivative of the the error, E, with respect to the weight vector, W. In [32], several approximations about h_{kk} were made to develop an efficient algorithm for computing the parameter during training. The result was a procedure similar to that of backpropogating to compute the first-order derivative of the error. This procedure is referred to as stochastic diagonal levenberg-marquardt.

2.6 Invariant Speech Recognition

Typical early HMM systems have shown that speaker independent systems typically make 2-3 times as many errors as speaker dependent systems. A notable work by Waibel showed that improvement in speaker independent systems can be obtained by training several speaker-dependent TDNNs and combining them [3].

One major source of interspeaker variability in HMM based continous speech systems is the vocal tract length (VTL). The VTL can vary from approximately 13cm for females to over 18cm for males. The formant frequencies (spectral peaks) can vary by as much as 25% between speakers [12]. In [13] invariant transformations on gammatone filterbank based feature vectors were shown to significantly improve recognition in mixed training and testing conditions. Early results in [15] showed that networks with shared weights along the frequency dimension can improve recognition in mixed training and testing conditions. Our goal in this work will be to explore improvements in speaker independent recognition which can be achieved through the use of the CNNs.

Chapter 3

Experiments and Results

3.1 Overview

Experiments have been conducted to study convolutional neural networks for speech recognition. The experiments have been performed using the TIMIT corpus. The phoneme classification task was chosen and results of the CNNs have been compared to the TDNN and a fully connected neural network (FINN). For all experiments the "Eblearn: Energy Based Learning" C++ library has been used to train and test the network. MATLAB has been used to perform the feature extraction.

3.2 TIMIT Corpus and Feature Extraction

The TIMIT corpus is a database of phonetically and lexically transcribed speech from American English speakers of different sexes and dialects. The corpus consists of a total of 6300 spoken sentences; 10 sentences are spoken by 630 different speakers from 8 major dialect regions of the United States. The TIMIT corpus has a variety of sentences selected by researchers at Texas Instruments(TI), MIT, and SRI. The

TIMIT documentation recommends a separation of test and training data [38]. The test set consists of 168 speakers of 1680 sentences and the training set consists of 462 speakers of 4620 sentences.

A subset of the sentences in the test and training set are referred to as "SA" sentences and are read by identical speakers in both the test and training sets; these have been designed to expose the various dialects. Using the same speaker for the test and training set would bias the results for speaker independent experiments. For this reason we have discarded the "SA" sentences as done in other works [18, 25].

The TIMIT database consists of wav files with speech sampled at 16kHz. For each sentence, a phonetically hand-labelled description gives the start and end time for each phoneme in the sentence. The corpus consists of a set of 61 different phonemes. Many of these phonemes are similar with regards to their sounds; confusion amongst them is not typically counted as an error. Typically, the 61 phoneme categories are folded into 39 phonetic categories [18, 25, 36]. Table 3.1 shows how the phonemes are folded. The MATLAB Audio Database Toolbox (ADT) is used to load and parse the description of the data into MATLAB for further analysis.

Phonemes are excised from each sentence in order to allow for training and testing. Extracting phonemes from sentences poses a general problem. Since phoneme length is variable, but the basic neural networks being tested require a fixed length input, we are faced with the problem of how to deal with this variability. For the purposes of demonstrating the abilities of CNNs, a single length was chosen to characterize all phonemes. In section 4.2 we describe some ideas for better dealing with this variability in more sophisticated systems.

In [9], a length of 150ms is used, however this work was conducted only on a subset of consonants. In [39], it is shown that a 150ms duration is superior to using a 200ms duration network on a subset of the TIMIT corpus consisting of vowels.

	Phoneme Category	TIMIT phonemes folded into category	Percent of Database
1	aa	aa,ao	3.46
2	ae	ae	2.30
3	ah	ah,ax,ax-h	3.63
4	aw	aw	0.42
5	ay	ay	1.38
6	b	b	1.26
7	ch	ch	0.47
8	d	d	2.05
9	dh	dh	1.63
10	dx	dx	1.56
11	eh	eh	2.22
12	er	er,axr	3.14
13	ey	ey	1.32
14	f	f	1.28
15	g	g	1.16
16	hh	hh,hv	1.22
17	ih	ih,ix	7.89
18	iy	iy	4.01
19	jh	jh	0.70
20	k	k	2.81
21	l	l,el	3.89
22	m	m,em	2.32
23	n	n,en	5.05
24	ng	ng,eng	0.79
25	ow	ow	1.23
26	oy	oy	0.39
27	p	p	1.49
28	r	r	3.77
29	s	s	4.31
30	sh	sh,zh	1.38
31	sil	pcl,tcl,kcl,bcl,dcl,gcl,h# , pau, epi	20.68
32	t	th	2.52
33	th	th	0.43
34	uh	uh	0.31
35	uw	uw,ux	1.42
36	v	v	1.15
37	w	w	1.81
38	y	y	0.99
39	z	z	2.17

Table 3.1 *List of phonetic categories folded based on [25] and percent of TIMIT training taken by the category*

Waibel speculated that the 200ms duration included extraneous information about adjacent phonemes causing worse results. On the full database there is a greater variability in the lengths of the phonemes. To further investigate the use of this length, distributions of duration for various phonemes were computed. The results are summarized in Figure 3.1.

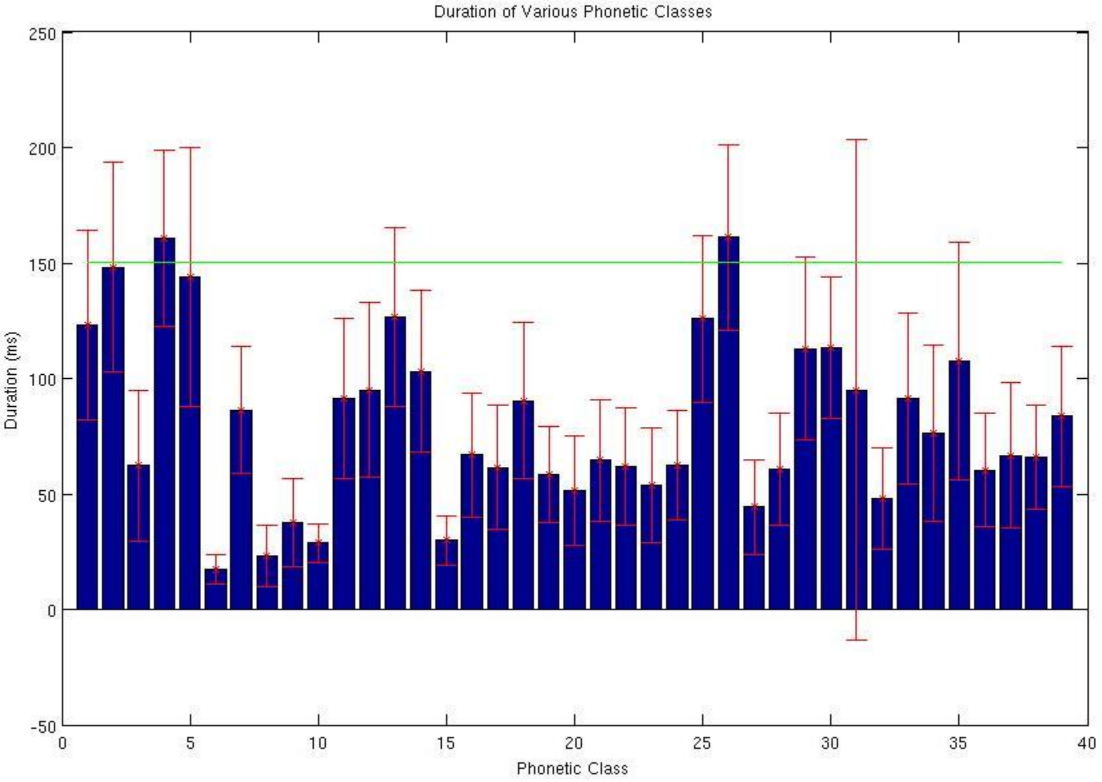

Figure 3.1 *Bar graph of means and standard deviation of phonetic classes corresponding to table 3.1*

It can be seen that 150ms is a long enough duration to incorporate the full length of the majority of phonemes. The phonemes which are slightly longer will still have a majority of their information contained within the segment. Several phonemes are generally significantly shorter (less than 70ms) in duration. It is expected that classification accuracy for these phonemes could be degraded due to extraneous data, but with a sufficient amount of examples, the extraneous data presented outside the boundary of the phoneme should be ignored as noise by the network. A potential improvement for this problem would be to pad smaller length phonemes, thus removing data beyond the boundary of the phoneme; however this might remove key temporal information. Another solution is to downsample or upsample the feature vector sequences to one length in a fashion similar to that discussed in [31]. This will be discussed in the section 4.2.

The phonemes are extracted from the sentences by finding the middle of the transcription and then capturing the previous and next 75ms and passing this 150ms segment to the feature extraction stage. The feature extraction has been performed using the Gammatone filterbank with ERB scaling as described in section 2.4. This stage produces 64 features per frame for 14 frames, corresponding to 64 bins of ERB-scaled filters between 10Hz and 8kHz. The output of the energy in each band is integrated over 20ms as done in [13], advancing by 10ms for each frame (50% overlap between frames). The final output is a gammatonegram of size 64 × 14. Figure 3.2 shows four examples of the phoneme category */iy/* processed in the manner described above. As we can see there are visually distinctive patterns that exist amongst examples of this category. There are 3 main areas of excitation with respect to the frequency. These can be seen as the reddest points along the frequency axis. We can see that these areas can be offset in time as well as less drastically offset in frequency when comparing different examples of */iy/*.

3.3 Computing Tools

Eblearn is a C++ based library aiming at allowing easy development of energy-based learning models [40]. An energy-based model is one in which an error (or energy)

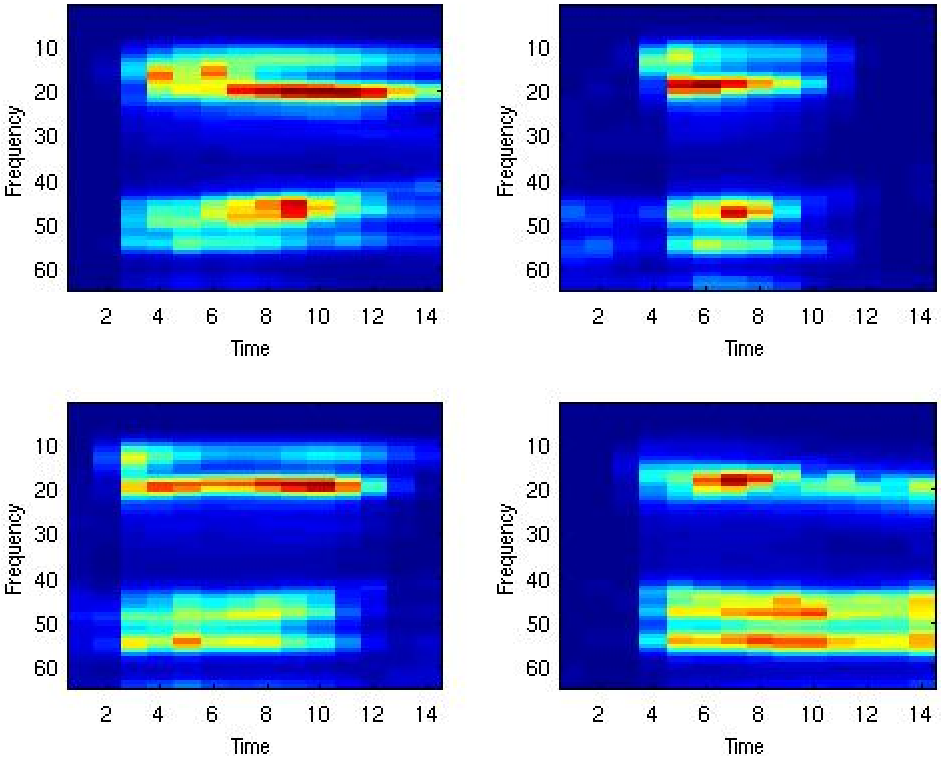

Figure 3.2 *An example of gammatone filterbank features extracted from different examples of the phoneme /iy /*

score is computed given a training example and a label. The library implements a very general approach for stochastic gradient descent backpropogation, allowing modules to be easily added and removed from a network. It implements all the known tricks to make gradient-based learning fast, including the stochastic diagonal Levenberg-Marquardt method, which was described earlier.

Eblearn functions through the use of modules which define how information is forwarded and backpropogated through the modules. Examples of modules available in Eblearn include a convolutional module, which performs convolution on an input image, bias modules, which add a bias to the input, nonlinearity modules, which can apply a sigmoid to the input, as well as a subsampling module which performs weighted subsampling on the input. These modules can be combined into layers, most notably convolutional layers and subsampling layers.

These flexible modules allow the formulation of the CNN, TDNN, and FINN (standard MLP) structures. A formulation of the TDNN using convolutional and subsampling layers is shown in Figure 3.3. We interpret the delays as convolutions in time mapping to various feature maps which correspond to rows in the second layer as seen in Figure 2.15. The next layer performs a convolution along each feature map combining the result into C feature maps, where C is the number of outputs. This layer is a convolutional layer with a full connection table. Finally, the subsampling layer can be used to perform the integration along time with the use of a window of size 1 × N, where N is the length of a feature map. Each feature map, of size 1 × N, will be multiplied by a single weight. Similarly a regular neural network can be constructed as shown in Figure 3.4 by treating the first layer as a convolutional layer with a full connection tables to N feature maps, of size 1 × 1.

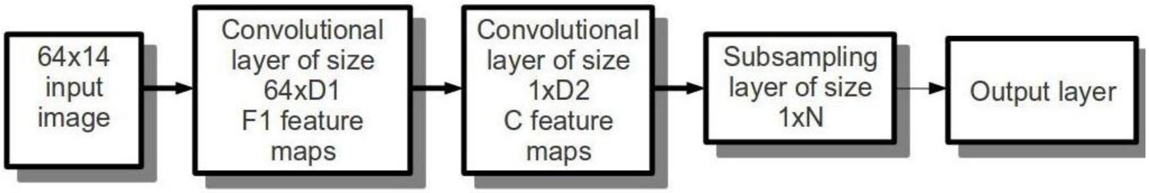
64x14 input image
Convolutional layer of size 64xD1 F1 feature maps
Convolutional layer of size 1xD2 C feature maps
Subsampling layer of size 1xN
Output layer

Figure 3.3 *The Eblearn construction of the TDNN network. D1 and D2 are the delays in the first and second layer, respectively. F1 represents the feature maps in the second layer. C is the number of output classes. N is the length of the feature maps in the final hidden layer.*

Malcolm Slaney's toolbox [41] as well as Dan Ellis' web resource [42] has been used in MATLAB to compute the ERB-scaled gammatone filter bank output for each frame of a phoneme. Ellis' algorithm approximated the gammatone filterbank output

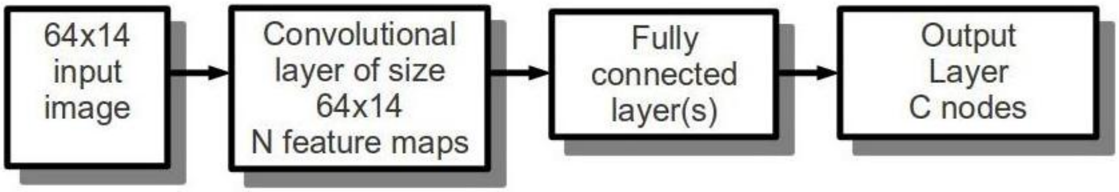

Figure 3.4 *The Eblearn construction of the FINN network. N is the number of nodes in the first hidden layer.*

by "calculating a conventional, fixed-bandwidth spectrogram, then combining the fine frequency resolution of the FFT-based spectra into the coarser, smoother gammatone responses via a weighting function." [42] This was done due to the extremely high computational complexity of the ERB filter bank routines from Slaney's toolbox. This approximation allowed feature extraction to occur 30-40 times faster.

All experiments have been performed on a desktop PC with an Intel Q6600 processor, clocked at 2.4GHz on 4 cores. Each experiment ran on a single core, and the estimated times which will be presented are intended only as rough estimates of the relative speed of training each network. There has been much work done in optimizing the neural network execution and training. The architecture of the network allows for highly parallizable architectures such as GPGPU and FPGAs to be used. Recent work has shown highly efficient and fully parametrized execution and training for CNNs on GPGPUs [43].

3.4 Stop Consonant Classification

In the seminal TDNN paper [9], the Japanese stop consonants */b/*, */d/*, and */g/* were used to conduct experiments. A further restriction on these consonants was that they be followed by a vowel. This type of combined consonant-vowel utterance is referred to as a CV utterance. These stop consonants are known for being difficult to distinguish. In other works the CV consonants */b/*, */d/*, and */g/* from TIMIT were used to conduct experiments with TDNNs [18]. We have chosen the full set of */b/*, */d/*, and */g/* consonants from TIMIT to perform initial experiments. These experiments are cruicial for exploring properties of the network structure as well as the invariance of the networks, due to the length of time needed to train on the full set of classes. To fully converge on the full TIMIT dataset containing 140,000 phonemes can take up to two weeks on a standard desktop computer for some network structures. However, this subset contains approximately 5000 training examples, and the network structure has significantly fewer connections due to the number of nodes at the output layer; thus convergence can be achieved in as little as 10 minutes in some cases. The fully interconnected neural network (FINN), time delayed neural network (TDNN), and the convolutional neural network (CNN) have been created and trained using Eblearn.

The FINN experiments have been conducted with a single-hidden layer as well as a two-hidden layer network. First we attempted to determine the effect of adding extra nodes to the hidden layer. The results can be seen in Table 3.2.

	Training Data	Test Data		Training Time	
Hidden Nodes	Overall Correct (%)	Overall (%)	Class Average (%)	Epochs	Duration
8	94.49	85.45	85.15	2200	50 min.
50	94.97	**86.37**	**85.97**	2200	3.4 hrs.
120	94.70	85.73	85.01	2200	7.3 hrs.

Table 3.2 Classifcation rates for single hidden layer fully connected networks.

There is a significant improvement when increasing the number of nodes from

8 to 50; however increasing to 120 nodes yields no further improvement and in fact seems to yield the same test result as 8 nodes. This might be because overfitting can occur more easily with a larger number of nodes. Adding more hidden nodes slows down the training time per epoch but does not improve the rate of convergence as measured in epochs.

Since the TDNN used by Waibel had multiple hidden layers we attempt to use a two-hidden layer structure for this fully connected network. The FINN constructed consists of two hidden layers, the first layer having 50 nodes and the second layer having 20 nodes. The performance of this network versus the best performing network above is shown in Table 3.3. The two-hidden layer network shows a slight improvement in performance on the test data and a significant improvement on the training data.

	Training Data	Test Data		Training Time	
Layers	Overall Correct (%)	Overall (%)	Class Average (%)	Epochs	Duration
1 (50 nodes)	94.97	86.37	**85.97**	2200	3.4 hrs.
2 (50-20 nodes)	98.27	**86.74**	85.80	2200	3.6 hrs

Table 3.3 Comparison of one and two hidden layer networks

A TDNN has been constructed to compare to the fully connected network. The TDNN uses a similar structure to that used in [9]. It is adjusted to fit the size and time scale of the features we have used, since Waibel used MFCCs. The length of the kernel in time has been selected as 2 for the first layer. Each frame is 20ms with a 50% overlap; thus 2 frames would encompass a 30ms period. [9] suggests this is optimal for learning low-level acoustic information. The second layer consists of a longer window of 4 frames. The structure has been implemented, as shown in Figure 3.3, as a series of 2 convolutional layers with kernels of size 64 × 2 mapping to feature maps with full connections. The second layer was represented as a convolution of kernels sized 1 × 4 mapping to 3 feature maps with a full connection table. The integration layer is represented with a subsampling layer of size 1 × 10 (full length of the final feature map). We have attempted to use 8 feature maps in the first hidden layer as in [9] as well as a larger number of 20 feature maps. Table 3.4 summarizes the results.

Feature Maps in	Training Data	Test Data		Training Time	
in First Hidden Layer	Overall Correct (%)	Overall (%)	Class Average (%)	Epochs	Duration
8	84.60	81.33	80.84	180	15 mins
20	84.58	**81.7**	81.2	180	20 mins

Table 3.4 Results for TDNNs on TIMIT stop consonant classification

The number of feature maps does not appear to greatly affect the result. Furthermore the TDNN performance is actually worse than the FINN performance on this subset of the data. Since the number of classes here is small, we can speculate that the FINN might be generalizing better due to the specific placement of the phoneme in the window. It is possible that the integration layer is reducing the resolution too much in the time domain. Another possible explanation for the discrepancy between past reported results is the larger data set used compared to [18] and [9]. As shown in [43] standard neural networks can perform as well as specialized networks given enough data which demonstrates the variability of the data. The time invariance advantage obtained by TDNN might become irrelevant once enough variability is shown to the rigid FINN. At that point the TDNN could be learning invariance we do not want, such as translation invariance to other consonants and vowels within the segment. It must also be noted that training in [18] and [9] were performed using a staged batch training approach, whereas in this experiment the TDNN is trained with stochastic gradient descent.

To test the capability of the convolutional neural network, a LeNet-5 structure was modified for the task of speech recognition as shown in Figure 2.16. The kernel size at the input layer has been chosen as 9×3. The time dimension has been chosen in a fashion similar to that of the TDNN. The frequency dimensions has been chosen based on results in [13]. Both subsampling layers have a subsampling window of 2×2 in order to introduce extra invariance to the network.

To compare the effect of the number of feature maps in the hidden layers, the feature maps in the first convolutional layer (C1) as well as second convolutional layer (C3) have been varied similarly to the FINN's hidden layers. A random connection table is used between the subsampling layer S2 and the convolutional layer C3, as done in [43]. Table 3.5 shows the results.

	Test	Training Data	Test Data		Training Time	
Network	Energy	Overall Correct (%)	Overall (%)	Class Average (%)	Epochs	Duration
CNN 6-16-120	0.501	99.75	85.64	85.08	60	49 min.
CNN 20-40-50	0.452	99.73	**86.97**	**86.41**	60	2.65 hrs
FINN	0.495	98.35	86.74	85.80	2200	3.6 hrs
TDNN	0.594	84.58	81.7	81.2	180	20 mins

Table 3.5 Results for best scoring FINN, TDNN, and CNN on TIMIT stop consonant classification. For the CNN the three hyphen separated values characterize the number of feature maps in C1 and C2 (as shown in Figure 2.16), and the nodes in the full layer. The energy indicates the average mean squared error across all the output activations in the test set.

As we can see increasing the number of feature maps improves the result on the testing data. The CNN generaly shows much better convergence on the training data, achieving near 100% convegence on the training data. The best performing CNN also obtained significantly lower average mean square error as compared to the best performing TDNN and FINN. Overall the CNN shows a slight improvement over the FINN on the test set. We can make a similar argument as before as to why the CNN performed only slightly better for this scenario, the training data may represent enough variation to the FINN to make the CNN's generalization improvements less significant.

We can further examine the performance on a per-class basis. As we can see from Table 3.6 the CNN has a larger performance improvement when averaging the per class performance. The per-class classification is further broken down in Table 3.6. The TDNN performs significantly worse for all classes. The FINN and CNN have similar performance in the */b/* and */d/* categories, with the FINN performing slightly better (about 0.5% and 0.2% improvement). The CNN however performs significantly better in the */g/* category (improving nearly 3%). This category is nearly half the size of the other categories. It is possible this class shows greater variability between the training and test sets, which might be due to a lack of enough training examples to represent all the variability that can be encountered in the test set. It is possible for the CNN to perform better in this scenario due to its invariant properties.

	Class Correct (%)		
Network	*/b /*	*/d/*	*/g/*
CNN 20-40-50	89.1	86.57	83.63
FINN	89.62	86.81	80.98
TDNN	83.18	80.38	78.93

Table 3.6 Error rates per class for stop consonant recognition. Categories */ b/*, */d/*, and */g/* have 886, 841, and 452 examples in the test set, respectively

3.5 Phoneme Classification on Full TIMIT Database

In this section we describe experiments on the full TIMIT corpus. Here all 39 phonetic categories are used. There are 140,000 example utterances used for the training set and 50,000 example utterances used for the test set. Tests are conducted using two CNNs with the same feature map sizes as in the stop consonant recognition experiments. Since this experiment has a larger number of output categories it may require an even larger number of feature maps to properly represent the data. Practical limitations on training such a large database prevent training larger networks. With each feature map there is a large number of extra convolutions and other operations that must be performed, increasing training time significantly. Future work can include hardware optimizations which would allow for larger networks to be trained, to see if performance can be further improved.

We use a larger FINN for this experiment consisting of 150 hidden nodes in the first hidden layer and 75 hidden nodes in the second layer. The TDNN has 75 feature maps in the first layer and 39 feature maps in the second layer. In general, convergence is faster on a per epoch basis, compared to previous experiments, since most classes have a large number of examples. No validation set has been used, as overtraining has not been observed on this large dataset. The results of these experiments are shown in Table 3.7.

As we can see the increase in feature maps from the smaller CNN to the larger CNN significantly improved performance. It has been found through preliminary experiments with the smaller CNN, that the number of hidden nodes in the fully connected layer did not signficantly affect results; thus the number of nodes in this layer has been decreased in the network with more feature maps, so as to speed up training time.

	Test	Train Correct	Test Correct		Training Time	
Network	Energy	Overall Correct (%)	Overall (%)	Class Average (%)	Epochs	Duration
CNN 6F-16F-120N	0.97	72.31	68.89	69.15	50	105 hrs.
CNN 20F-40F-50N	0.90	80.19	**71.76**	**70.48**	30	270 hrs.
FINN	0.99	76.1	68.68	65.48	30	25 hrs.
TDNN	1.09	67.47	63.75	63.63	50	125 hrs.

Table 3.7 Results for the full TIMIT phoneme set and example set.

The CNN has achieved better performance than the other networks in both the class average correct rate and the overall correct rate. Even for the smaller network, we can see that it is able to outperform the FINN on the test set.

3.6 Mixed Gender Training and Testing Conditions

In order to test the ability of the CNN to perform well in mixed training and testing conditions we have constructed a mixed gender experiment as in [13]. As discussed previously, the female vocal tract length is generally shorter than the male vocal tract length, thus training on only one gender and testing on the other would give us an extreme scenario we can use to better isolate the performance in terms of vocal tract length variability. Preliminary experiments have shown that overtraining would occur on the female training set before the training error would converge. This is likely due to the smaller number of examples per class. Thus we create a validation set which would be used to determine the stopping criterion.

We choose the first 30,000 example utterances spoken by females from the TIMIT training set. We choose another 10,000 examples as the validation set. The test set is chosen as the male subset of the regular TIMIT test set. This containes 33,624 utterances. We have used the same networks specified in the full training and test set experiments. The experiments are run to convergence and the network weights that produced the smallest validation energy are chosen for testing. The results are summarized in Table 3.8.

	Validation	Test	Train Correct	Test Correct		
Network	Energy	Energy	Overall (%)	Overall (%)	Class Average (%)	Epoch
CNN 20F-40F-50N	1.092	1.551	82.6	**40.05**	**34.06**	10
CNN no-subsampling	1.059	1.56	91.4	37.87	27.05	5
FINN	1.297	1.76	60.8	28.29	25.64	16
TDNN	1.334	1.728	56.73	29.83	28.70	6

Table 3.8 Results for the mixed gender training and testing conditions for TIMIT. Unlike in previous tables, here the epoch refers to the epoch chosen for early stopping, based on the smallest validation energy.

Silence is a much larger category compared to the other categories, as we can see from Table 3.1. The large gap in the performance between the class average classification rate and overall classification rate for the CNN can be in large attributed to this category. For the CNN, the silence category had a 22.9% error rate, while for the TDNN and FINN it was 53.4% and 50.99%, respectively. Since there might be simpler, knowledge-based methods, for identifying this category, we should give more consideration to the class average performance, as that can tell us more about the networks discriminating ability, without the large bias created by the silence category.

The subsampling layer is one of the significant differences between the BWNN, used in [15], and the CNN we have constructed. In order to study the importance of the subsampling layers for invariance we constructed a CNN with the same structure as the best performing CNN except without the two subsampling layers. It can be seen in Table 3.8 that this network performed similarly to the FINN and TDNN networks in the per class average category. However it outperformed the the TDNN and FINN in the overall category, due largely to its improved performance in the silence category obtaining 27.9% error rate in this category.

The CNN significantly outperforms the other networks in both the overall and class average performance. We can also observe that the TDNN, although performing worse on the training set then the FINN, as we have seen before, has better performance on the test set. We can attribute this to the TDNN's own invariant properties.

Chapter 4

Conclusions and Future Work

4.1 Conclusion

The CNN structure's usefulness with respect to several phonetic classification tasks has been examined. The CNN shows improved performance over the simpler neural network structures studied. For voiced stop consonant recognition the CNN shows better discriminating ability on a per-class basis than the two other networks tested. In recognizing all TIMIT categories the CNN performs significantly better than the competing structures, furthermore there is reason to believe that if training time is shortened, larger network structures could be tested which may obtain further improvements. The CNN showed large improvement over the classical structures in the mixed gender training and testing condition. This suggests a better ability to model speaker variability.

We have found that the TDNN trained with stochastic gradient descent on larger datasets can perform worse than the regular neural network. As discussed previously the integration layer of TDNNs might create too much invariance when presented with larger datasets. The smaller, more localized, subsampling windows of the CNN inspire a better mechanism for achieving similar invariance.

In our experiments it has been shown that the neural networks performance is susceptible to various parameters. The network is sensitive to some parameters more than others. For example some preliminary experiments have shown that the CNNs are not easily affected by changing the kernel size or by changing the number of nodes in the fully connected layer. However, the number of feature maps significantly affects performance. Due to the training time needed to see how well the network performs exhaustive parameter searches have not been implemented. Further work in parameter selection can potentially improve performance.

4.2 Future Work

There is a great deal of extensions and work that would need to be undergone to extend the use of convolutional networks to speech recognition. The next step in determining the possiblities of this method within a continous speech recognition framework is the extension into a phoneme recognizer as has been done for other discriminant methods [27]. This would require development of an appropriate segmentation stage as well as a model of the inter phoneme structure. This could potentially be accomplished with a HMM as in [31]

As we have seen from the stop consonant experiments networks trained to distinguish between specific categories perform better than those trained between all categories. This improvement can be largely attributed to variability in length between phonemes. This suggests an approach similar to [18] can be constructed, where the broad phonetic category is first detected and then various networks are trained for particular phonetic categories to yield a greater overall accuracy.

Another approach for dealing with variability in the length of the phoneme is that described in [31] and [3] as a segmented neural network. Here each segmented phoneme is downsampled to an appropriate number of frames. The segmentation was achieved using a HMM model and viterbi decoding.

Another extension of the CNN's invariance can occur for frame-level training. A one-dimensional convolutional kernel as well as subsampling layers can be constructed to achieve invariance in frame based HMM/NN hybrid system.

Bibliography

[1] (2009) Speech compression. [Online]. Available: http://www.data-compression.com/index.shtml

[2] R. Port. (2005) Audition for linguists. [Online]. Available: http://www.cs.indiana.edu/~port/teach/641/hearing.for.linguists.html

[3] J. Tebelskis, "Speech recognition using neural networks," Tech. Rep., 1995.

[4] Wikipedia, "Mel scale," 2010, [Online; accessed 22-October-2010]. [Online]. Available: http://en.wikipedia.org/wiki/Mel_scale

[5] D. Johnson. (2009, August) Spectrograms. [Online]. Available: http://cnx.org/content/m0505/2.20/

[6] J. O. Smith, *Spectral Audio Signal Processing, October 2008 Draft.* http://ccrma.stanford.edu/ jos/sasp/, 2010, online book.

[7] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4.* Cambridge University Press, 2006.

[8] J. Hennebert, M. Hasler, and H. Dedieu, "Neural networks in speech recognition."

[9] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, 1989.

[10] K. Davis, R. Biddulp, and S. Balashek, "Automatic recognition of spoken digits," *Journal of Acoustic Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[11] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.

[12] L. Lee and R. Rose, "Speaker normalization using efficient frequency warping procedures," *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1, pp. 353–356, 1996.

[13] F. Müller, E. Belilovsky, and A. Mertins, "Generalized cyclic transformations in speaker-independent speech recognition," in *Proc. 2009 IEEE Automatic Speech Recognition and Understanding Workshop*, Merano, Italy, Dec. 13-17 2009.

[14] Y. S. Lin Zhong and R. Liu, "A dyanmic neural network for syllable recognition," *Proc. Int. Joint. Conf. Neural Networks (IJCNN)*, 1999.

[15] H. Sawai, "Frequency-time-shift-invariant time-delay neural networks for robust continuous speech recognition," *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 0, pp. 45–48, 1991.

[16] Wikipedia, "Wikipedia," 2010, [Online; accessed 22-October-2010]. [Online]. Available: http://en.wikipedia.org/wiki/Phoneme

[17] L. Rabiner and H. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[18] J. Hou, "On the use of frame and segment-based methods for the detection and classification of speech sounds and features," Ph.D. dissertation, Rutgers University, New Jersey, United States, 2009.

[19] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *In International Symposium on Music Information Retrieval*, 2000.

[20] K. AidaZade, C. Ardil, and S. Rustamov, "Investigation of combined use of mfcc and lpc features in speech recognition systems," *World Academy of Science, Engineering and Technology*, 2006.

[21] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Prentice–Hall, 1975.

[22] R. Patterson and I. Nimmo-Smith, "An efficient auditory filterbank based on the gammatone function," MRC Applied Psychology Unit, Tech. Rep., 1987.

[23] B. Glasberg and B. Moore, "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, vol. 47, pp. 103–138, 1990.

[24] J. Rademacher and A. Mertins, "Auditory filterbank based frequency-warping invariant features for automatic speech recognition," in *Proc. ITG-Fachtagung Sprachkommunikation*, Kiel, April 2006.

[25] S. Fernndez, A. Graves, J. Schmidhuber, S. Fernndez, A. Graves, and J. Schmidhuber, "Phoneme recognition in timit with blstm-ctc," 2008.

[26] D. G. M. Scordilis, "Tdnn vs. fully interconnected multilayer perceptron: A comparative study on phoneme recognition," *Proceedings of the Fourth Australian International Conference on Speech Science and Technology*, pp. 214–219, 1992.

[27] A. Ganapathiraju, “Support vector machines for speech recognition,” Ph.D. dissertation, Mississipi State University, Mississippi, United States, 2000.

[28] S. E. Krger, M. Schaffner, M. Katz, E. Andelic, and A. Wendemuth, “Speech recognition with support vector machines in a hybrid system,” in *in Proc. EuroSpeech, 2005*, 2005, pp. 993–996.

[29] C. Bishop, *Pattern Recognition and Machine Learning.* Springer Verlag, 2006.

[30] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[31] S. Austin, G. Zavaliagkos, J. Makhoul, and R. Schwartz, “Speech recognition using segmental neural nets,” *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1, pp. 625–628, 1992.

[32] Y. LeCun, L. Bottou, G. Orr, and K. Muller, “Efficient backprop,” in *Neural Networks: Tricks of the trade*, G. Orr and K. Muller, Eds. Springer, 1998.

[33] D. Hubel and T. Wiesel, “Receptive fields and functional architecture of monkey striate cortex,” *Journal of Physiology*, vol. 195, pp. 215–243, 1968.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Intelligent Signal Processing.* IEEE Press, 2001, pp. 306–351.

[35] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time-series,” in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[36] J. R. Glass, “A probabilistic framework for segment-based speech recognition,” *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.

[37] A. Waibel, “Consonant recognition by modular construction of large phonemic time-delay neural networks,” in *NIPS*, 1988, pp. 215–223.

[38] N. I. of Standards and T. (NIST), “The darpa timit acoustic-phonetic continuous speech corpus,” online Documentation.

[39] A. Waibel and H. Nobou, “Speaker-independent phoneme recognition on timit database using integrated time-delay neural networks (tdnn),” in *EUROSPEECH*. ISCA, 1991.

[40] P. Sermanet, K. Kavukcuoglu, and Y. LeCun, “Eblearn: Open-source energy-based learning in c++,” in *Proc. International Conference on Tools with Artificial Intelligence (ICTAI’09)*. IEEE, 2009.

[41] M. Slaney, “Auditory toolbox version 2,” 1998. [Online]. Available: http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/

[42] D. P. W. Ellis, “Gammatone-like spectrograms,” 2009, web resource. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram

[43] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep big simple neural nets excel on handwritten digit recognition,” *CoRR*, vol. abs/1003.0358, 2010.