The University of Derby

College of Engineering and Technology

Department of Electronics, Computing and Mathematics

A Comparison of Feature Extraction Methods for Automatic Speech Recognition

by

Dan Roth

September 19, 2019

A thesis submitted in partial fulfilment of the requirements for the MSc in Audio Engineering

Abstract.

The following dissertation is a collection of research and experimentation related to fundamental processing techniques for automatic speech recognition (ASR) with a focus on various methods of feature extraction ASR. These feature extraction methods were compared in order to observe the effect of altering the acoustic analysis and processing involved in ASR as opposed to the more traditional approach of emphasizing different methods of statistical analyses in order to enhance an ASR system. Perceptual Linear Prediction (PLP), Gammatone Frequency Cepstral Coefficients (GFCCs), and Power Normalized Cepstral Coefficients (PNCCs) were chosen as the most promising alternatives to the current reigning paradigm in feature extraction, Mel Frequency Cepstral Coefficients (MFCCs). Kaldi, an open source C++ ASR library, was used to create a total of 72 models using different datasets and feature extraction methods. The Voxforge and TIMIT datasets were used for training and testing models, and a set of noise-corrupted TIMIT databases were created using AURORA, a C library made for convolving speech databases with noise. The resultant models were scored using the Word Error Rate (WER) metric. It was found that MFCCs and PLP techniques were mostly able to outcompete alternative methods, particularly in contexts that used more relevant modelling techniques and when testing with noisy speech databases. It was concluded that it is best to focus on enhancing statistical methodologies for ASR as opposed to focusing on acoustic analysis techniques for ASR at this time.

Table of Contents

1 Introduction

Automatic speech recognition (ASR) has recently emerged as a promising technology that has created new applications for command and control systems, transcription and dictation, and dialogue with artificial intelligence (AI). Although some of the statistical framework behind ASR functionality has long existed, this technology has just recently begun to realize its potential in a variety of different environments. One of the first prominent ASR systems was a digit recognizer constructed by Bell Labs in 1952 (Wu, 2018). ASR technology was one of the first pursuits in AI research as speech has typically been heralded as one of the most natural forms of communication for humans; it only makes sense that one of the first priorities in implementing AI would be to equip technology with the ability to also utilize speech (Markowitz, 1996; Gales and Young, 2007; Wu, 2018). Recent iterations of ASR have transformed a handful of different contexts: stock traders at Lehman Brothers observed a 100% increase in trade volume and a 23% increase in trade accuracy in a test in 1993; "Parrotron," a speech-to-speech model that can effectively normalize speech, led to an error rate decrease of 15.9% when decoding noisy speech and has seen great success in converting heavily accented or atypical speech into decipherable speech for the speech impaired as well as selecting the loudest speaker in a group of overlapping speakers (Markowitz, 1996; Biadsy et al., 2019). It is clear that ASR can be applied to a great many applications in order to promote the ability of AI to aid humans in their daily lives.

While ASR has shown incredible promise, there is still more work to be done in order to create the most effective ASR systems. At the time of writing, the most accurate ASR system is Google's Convolutional Long Short-Term Memory Fully Connected Deep Neural Network, abbreviated as CLDNN. This model architecture can achieve error rates as low as 8% (Chan et al., 2015). It combines a Convolutional Neural Network (CNN) frontend that is effective at reducing frequency variations that is then fed into a Long Short-Term Memory (LSTM) network which normalizes temporal nonlinearities and finally concludes at a Fully Connected Deep Neural Network (DNN) in order to optimize feature vector decorrelation (Sainath et al., 2015). This then feeds into a Hidden Markov Model (HMM), but this topology will be described in greater depth in a later section. This system is able to achieve incredible accuracy when linked and trained as a single model, but it requires a great amount of computational depth in order to optimize and cleanly join all of these moving elements. However, feature extraction, which represents the ASR system's acoustic analysis and interpretation of incoming speech signals into feature vectors, is an often-overlooked segment of the speech recognition process. A great deal of research literature assumes that Mel Frequency Cepstral Coefficients (MFCCs) are used as the reigning feature extraction paradigm, and thus are the standard to be compared against (Al-Noori et al., 2016; Gales and Young, 2007; Kim and Stern, 2016; Povey et al., 2011). There are many potential alternatives to MFCCs that have been cited to show better performance statistics when trained on the same ASR models. It is the interest of this dissertation to compile and verify the findings of prior feature extraction research and to explore the possibility of leveraging the power of feature extraction techniques to improve ASR accuracy rates.

In order to provide a comprehensive analysis of the many moving elements of an ASR system, the fundamental concepts behind designing ASR models will first be examined. This will then be followed by a more in-depth understanding of feature extraction and how it fits into the bigger picture of ASR, as well as an overview of the predominant techniques that are typically compared against MFCCs. Finally, an experiment will be detailed where a multitude of ASR models will be trained with different datasets and feature extraction techniques in order to test the performance of each of these possible alternatives. The findings will be summarized and analyzed in a results section and conclusions will be drawn regarding the nature of feature extraction methods for improving ASR.

2 The Anatomy of ASR Systems

2.1 Overview

In order to properly assess and analyze the differences between varying ASR systems it is necessary to have a familiarity with the building blocks of an ASR model. There exist many permutations to the ASR formula as these technologies have developed in a multitude of ways in recent decades. A plethora of different feature extraction frontends, model and decoder choices, etc. exist and can affect the performance and accuracy of finished ASR models. This research aims to primarily draw upon the techniques of the predominant Hidden Markov Model-Gaussian Mixture Model (HMM-GMM) approach, largely outlined by Mark Gales' and Steve Young's paper, "The Application of Hidden Markov Models in Speech Recognition" (Gales and Young, 2007).

2.2 Hidden Markov Models

Hidden Markov Models, or HMMs, are the foundational element of many predictive systems such as those implemented in weather prediction, text prediction, or automatic speech recognition. Discrete density HMMs were first used in ASR systems by groups at Carnegie Mellon and IBM in the 1970's and this research was later continued by Bell Labs with the introduction of continuous density HMMs (Gales and Young, 2007). Initial technologies were focused on systems that were either speaker-dependent with discrete word large vocabularies or speaker-independent systems with smaller vocabularies. A system is said to be speaker-dependent if it is trained and tested for use by a specific group of speakers, whereas speaker-independence indicates that the opposite is true: speaker-independent systems are meant to accept a variety of speech inputs from any number of speakers. This culminated in the early 1990's with a new focus on continuous speaker-independent recognition tasks, which remains the current focus of today.

An HMM provides a solution to the inherent problem of modelling speech recognition. First, speech audio is recorded by a microphone and transformed into a fixed-size acoustic vector $\boldsymbol{Y}_{1:T} = \boldsymbol{y}_{1},\ldots,\boldsymbol{y}_{t}$; the process that transforms audio into this vector is called feature extraction. This process, as it is the focus of this research, will be discussed in greater depth in later sections. A decoder is then deployed in order to find the most likely sequence of words of length L, $\boldsymbol{w}_{1:L} = \boldsymbol{w}_{1},\ldots,\boldsymbol{w}_{L}$, that is represented by $\boldsymbol{Y}$. The probabilistic model that represents this task is given below:

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}}\{P(\boldsymbol{w}|\boldsymbol{Y})\}.$$

Equation 2.1 - Simplified probabilistic model for the central task of ASR (Gales and Young, 2007).

However, since it is considered quite difficult to model $P(\boldsymbol{w}|\boldsymbol{Y})$ directly, Bayes' Rule, which is perfectly suited for a task where the probability of a current event is to be calculated based on the probability of a prior event, is applied to the equation in order to transform it into the following:

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}}\{p(\boldsymbol{Y}|\boldsymbol{w})P(\boldsymbol{w})\}.$$

Equation 2.2 - The Bayesian transformation of Equation 2.1 (Gales and Young, 2007).

This divides the needs of ASR modelling into two smaller models: the first is an acoustic model which is represented by the likelihood $p(\boldsymbol{Y}|\boldsymbol{w})$ that aims to approximate an acoustic vector based on the available words, or vocabulary, present in the ASR system; the second is a language model symbolized by the likelihood $P(\boldsymbol{w})$ that represents the probability of a word occurring given a dictionary that informs the potential word choices for an ASR model. The acoustic model works on units called phonemes, which are small phonetic units that, when joined together, form pronunciations of speech. For example, there are 40 total phonemes in the English language (Gales and Young, 2007). During training, the acoustic model uses training audio samples and their respective orthographic transcriptions in order to gain a sense of how different phonemes join to create different words, which ultimately creates a pronunciation dictionary. The language model is an *N*-gram model, which models the likelihood of a word based on the previously observed word. This can be presented by the following equation when seeking to output $\boldsymbol{w}_{\mathrm{n}}$:

$$P(\boldsymbol{w}_{\mathrm{n}} \mid \boldsymbol{w}_{\mathrm{n-1}})$$

Equation 2.3 - Probabilistic representation of *N*-gram model operation.

Decoders can then either search through all possible word sequences and remove any sequences deemed unlikely or generate lattices to represent the most likely possibility that is then chosen to become the predicted word sequence.

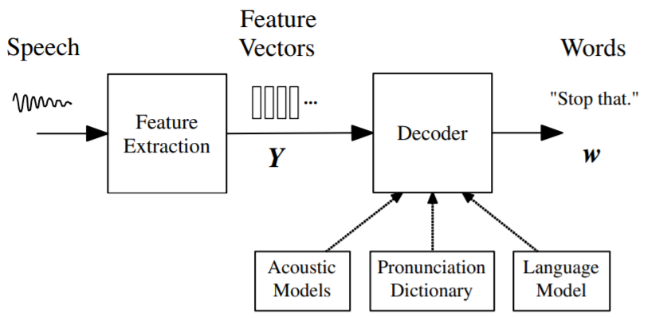


Figure 2.1 - A simplified representation of ASR system anatomy (Gales and Young, 2007).

In order to better prepare the extracted features for input into an HMM, dynamic features are also calculated as it has been found that including dynamic parameters is a key element of phonetic perception (Furui, 1986). These parameters take the form of first order (delta) and second order (delta-delta) regression coefficients. The delta parameter, Δy_t^s, which represents the slope of the time function of the feature parameter y_t^s, can be calculated as follows:

$$\Delta \boldsymbol{y}_t^{\mathsf{s}} = \frac{\sum_{i=1}^{n} w_i \left(\boldsymbol{y}_{t+i}^{\mathsf{s}} - \boldsymbol{y}_{t-i}^{\mathsf{s}}\right)}{2\sum_{i=1}^{n} w_i^2}$$

Equation 2.4 First order (delta) regression coefficient calculation for a feature vector (Gales and Young, 2007).

In this equation *n* represents the window length and w_i are regression coefficients. The second order delta-delta parameters would then be similarly derived but instead using the delta parameters' differences. The final feature vector is a concatenation of these three processes (feature extraction and the subsequent two orders of regression coefficients) and results in the following:

$$\boldsymbol{y}_t = \left[\boldsymbol{y}_t^{\mathsf{s}\mathsf{T}} \quad \Delta \boldsymbol{y}_t^{\mathsf{s}\mathsf{T}} \quad \Delta^2 \boldsymbol{y}_t^{\mathsf{s}\mathsf{T}}\right]^{\mathsf{T}}.$$

Equation 2.5 - Final feature vector (Gales and Young, 2007).

Research has found an error rate decrease of 3% - 4% when combining the original cepstral coefficients output from feature extraction with these dynamic regression coefficients (Furui, 1986).

At this point, the phones are ready to be modelled by HMMs. This is typically modelled as below:

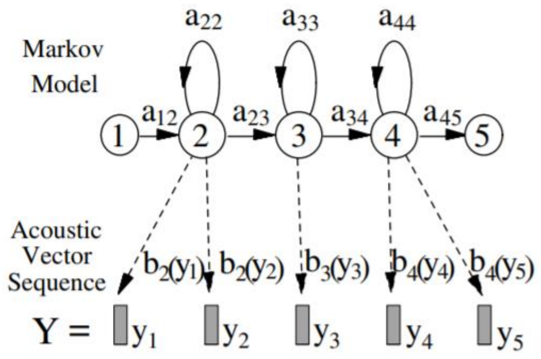


Figure 2.2 - HMM phone model (Gales and Young, 2007).

This continuous density HMM makes a transition from one state to another for every time step, the probability of transitioning from one state to another is given by $\boldsymbol{a}_{ij}$. Once a state has been entered, it outputs a feature vector represented by $\boldsymbol{b}_i()$. The HMM moves through each phoneme unit step by step, using calculated probabilities in order to determine which phoneme is most likely to be next based on the phoneme prior. This style of acoustic modeling can be referred to as a *beads-on-a-string* model because speech prediction outputs are modeled on independent phonemes that are strung together (Gales and Young, 2007). This is referred to as monophonic modeling and it typically assigns a multivariate Gaussian distribution that contains the mean and covariance of each state to each respective output. Unfortunately, this raises some immediate issues as we often rely on more than just single phoneme units in order to identify words; oftentimes we depend on knowing and understanding the context of these phonemes.

This gives rise to context dependent phone modeling, or triphonic modeling. Triphone modeling does not just attempt to identify the phoneme present at the time step itself, but also the neighboring phonemes preceding and following the current phonetic unit. This creates new models that represent three phones at a time, demonstrated by the graphic below:

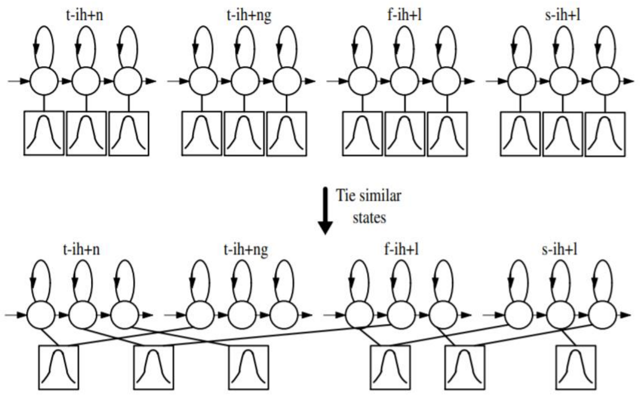


Figure 2.3 - HMM triphone models (Gales and Young, 2007).

Similar states within these models are tied in order to condense the phonetic representation. While a monophonic model simply relies on *N* base phones, a triphone model now requires N^3 base phones (Gales and Young, 2007).

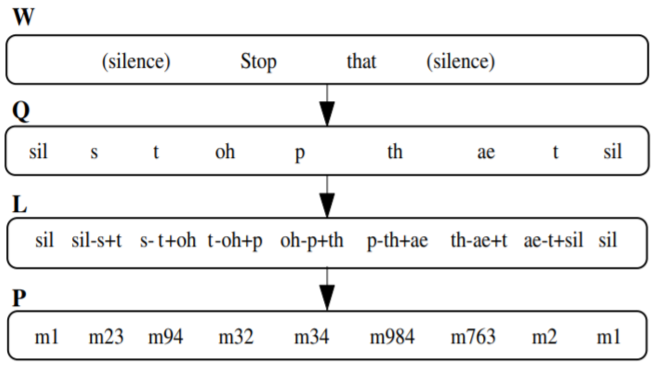

W
(silence) Stop that (silence)
Q
sil s t oh p th ae t sil
L
sil sil-s+t s-t+oh t-oh+p oh-p+th p-th+ae th-ae+t ae-t+sil sil
P
m1 m23 m94 m32 m34 m984 m763 m2 m1

Figure 2.4 - Model condensation in triphonic modeling (Gales and Young, 2007).

The diagram above illustrates the processing levels implemented in the creation of triphone modeling. The top level, *W*, represents the observed or target words. *Q* is a series of corresponding pronunciations drawn from the pronunciation dictionary created for the acoustic model, and then these are grouped together in sets of logical triphones on level *L* that consist of x – q + y phonemes. q represents the central phoneme and x and y are the prior and following phonemes respectively. Finally, level *P* assigns model IDs to these triphone units so that the recognizer can easily sort and distinguish between the potential phoneme groups (Gales and Young, 2007). These act a series of states which can then be re-formatted into a decision tree, which allows for great flexibility in deploying new models from the same collection of states. It is greater streamlined by the tied-state approach described above, which groups the Gaussian output distributions of a state and its nearest neighbors in order to create a Gaussian Mixture Model, or GMM.

In summary, an acoustic model is first created by extracted monophones from training data and outputting Gaussian distributions for these monophones that consist of a series of means and covariances that mirror the means and covariances of the raw training data. Expectation Maximization (EM) is then employed in the form of a forward-backward algorithm, which calculates the best values for increasing the posterior probability of the hidden states in an HMM using a forward and backward pass. This is done over several iterations in order to increase the probability that these hidden states correlate strongly with their associated observed states. Each monophone q is then copied and applied to a triphone model x – q + y and EM is applied to these triphones over several iterations to compute their Maximum Likelihood (ML).

ML would be computed for model parameters **λ** using a feature vector ***Y*** in reference to a word sequence $\boldsymbol{w}_{ref}^{(r)}$ using the following equation:

$$\mathcal{F}_{\mathtt{ml}}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\log p(\boldsymbol{Y}^{(r)}|\boldsymbol{w}_{\mathbf{ref}}^{(r)};\boldsymbol{\lambda})$$

Equation 2.6 - ML estimation for an ASR model (Gales and Young, 2007).

Finally, the triphone states are mapped onto a decision tree and their Gaussian distributions are mapped into tied-state form and EM is applied again. This represents a basic tied-state triphonic HMM acoustic model that forms the basis of the acoustic component of an ASR system (Gales and Young, 2007).

2.3 Language Models and Decoding

This model must now be paired with a language model in order to give an ASR system a greater sense of grammar and context within a specified language. Language can be predicted based on the probability of a word sequence depicted by the following for a word sequence $\boldsymbol{w} = w_1,\ldots,w_k$:

$$P(\boldsymbol{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1},\ldots,w_1).$$

Equation 2.7 - Probability of a word sequence (Gales and Young, 2007).

An *N*-gram representation is then deployed that is better suited for large vocabulary recognition, truncating the word history to $N - 1$ words (typically *N* is within the range of 2-4) in the following model representation:

$$P(\boldsymbol{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1},w_{k-2},\ldots,w_{k-N+1}),$$

Equation 2.8 - *N*-gram Language Model (Gales and Young, 2007).

The probabilities for these *N*-gram models are estimated from training data *N*-gram occurrences using ML estimates. This estimation takes place for the number of occurrences $C(w_{k-2}w_{k-1}w_k)$ of the words $w_{k-2}w_{k-1}w_k$ as:

$$P(w_k|w_{k-1},w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})}.$$

Equation 2.9 - Maximum Likelihood (ML) estimates for *N*-gram models (Gales and Young, 2007).

If data is sparse, this approach can be flawed but compensated for using either Katz or Kneser-Ney smoothing in order to discount the estimation of a model that is lacking in data, which takes the form of weights that are applied to all unseen *N*-grams in order to avoid overfitting an insufficient dataset (Gales and Young, 2007). Class-based trigrams can also be used by associating words with particular classes, or larger groupings of words. These classes are already assumed to be far less numerous than the total number of words, thus negating the effects of dramatic variance in dataset size; words can be sorted into each class by observing the grouping where their recognition likelihood would be highest.

Once acoustic and language models are in place, there needs to be a system for combining the two models in order to properly decipher or decode the observed speech signals. Returning to the problem of finding the most probable word sequence $\boldsymbol{w}$ that fits a feature vector $\boldsymbol{Y}_{1:t}$, it has been found that the Viterbi algorithm is the most robust method for computing these probabilities. The Viterbi algorithm is a dynamic programming method, meaning it breaks a problem down into both a mathematical and computationally efficient function. This typically translates to the dilution of a problem into simpler sub-problems and recursively solving these sub-units in order to arrive at the final solution. A Viterbi algorithm is most potent for ASR applications as it seeks to find the most likely set of hidden states that corresponds to a set of observed states. This algorithm is conveyed by the following equation:

$$\phi_t^{(j)} = \max_i \left\{ \phi_{t-1}^{(i)} a_{ij} \right\} b_j(\boldsymbol{y}_t).$$

Equation 2.10 - The Viterbi Algorithm for ASR (Gales and Young, 2007).

$\phi_0^{(j)}$ is typically set to 1 as an entry state and all other states are set to 0; the maximum probability is thus conveyed by $\max_j\{\phi_0^{(j)}\}$ in respect to a transitional probability (representing a hidden state) and the Gaussian distribution of a feature vector (representing an observed state) as conveyed by the acoustic model. Each maximization is recorded for each possible word sequence and the highest probability word sequence is chosen as the output. Unfortunately, the complexity of sorting through high-dimensional feature vectors and pairing them with potentially large vocabulary language models leads to computational difficulties for a normal Viterbi approach. There have been several methods introduced to combat this, such as processing several Viterbi computations in parallel or dynamically expanding a search space as the algorithm sees fit in order to limit the probabilities to be calculated (Gales and Young, 2007).

In order to further enhance the process of finding the most probable word sequence to fit an observed speech command, decoders are often built to produce the *N*-best set of hypotheses, where N usually ranges from 100-1000 (Gales and Young, 2007). The hypotheses are stored in a word lattice that takes the form of a series of nodes representing time that contain word IDs and their associated acoustic and language model scores. This is a desirable decoding technique because it allows many possible interpretations of the data to be calculated at once while a model has already been created. It can also account for any minor inconsistencies that may have led to confusion or misinterpretation of an input signal. These word lattices can be used as input into another recognition network or potentially reassessed by other language models, which allows for a great deal of expandability. Confusion networks can also be formed by merging the nodes of a word lattice into the simpler representation of a summation of potential word sequences.

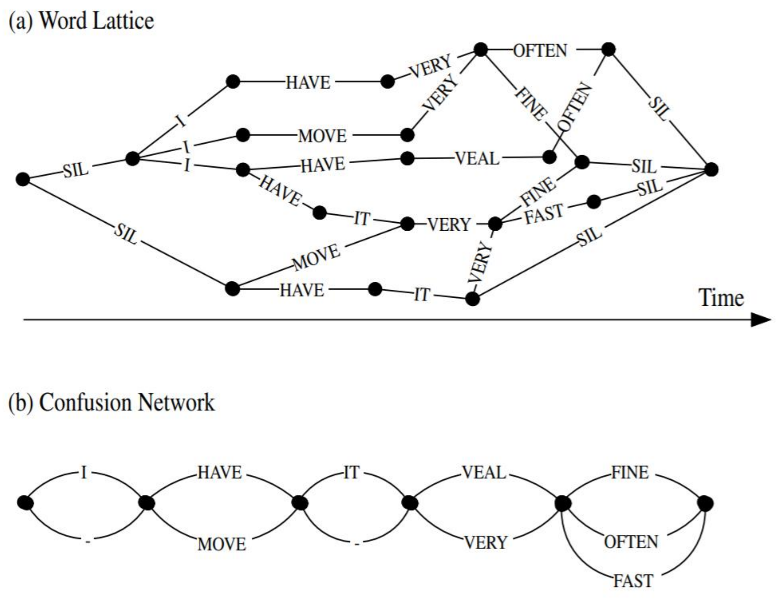


Figure 2.5 - (a) Example word lattice. (b) Example confusion network (Gales and Young, 2007).

This forms a simpler representation of all the paths present in the original word lattice but in a format most efficient for analyzing potential hypotheses. The arcs of a confusion network contain the posterior probability of their corresponding words. This is calculated by summing all probabilities of the word within the original word lattice through a forward-backward algorithm and normalizing these instances to sum these probabilities into one arc. The confusion network representation is considered to be an effective method of Minimum Bayes Risk (MBR) decoding, which seeks to minimize the loss function within a machine learning context.

In order to most effectively compile and decode interlocked acoustic and language models, finite-state automata are often employed. These automata can typically take one of two forms: the first is a Finite-State Acceptor (FSA), which is a set of finite states that are meant to represent and verify a string input; the second is a Finite-State Transducer (FST), which is a series of finite states that represent binary relations between strings (Gorman, 2019). Each state is associated with an input string and its corresponding output string. Weights can then be added in order to create a Weighted Finite-State Acceptor and Transducer (WFSA and WFST, respectively). These weights are meant to signify transition costs, but in ASR these weights correlate to the probabilities associated with entering each state. When moving through an WFSA or WFST, a decoder begins at a designated starting state 0 and proceeds through the graph based on the input observations and probabilistic weights.

An ASR decoding graph is a compilation of several different finite-state automata that combine all the previously created elements of the ASR model (edX, 2019). The first automata is a grammar WFSA, G, that assigns weights to a flat set of phrases in the grammar model.

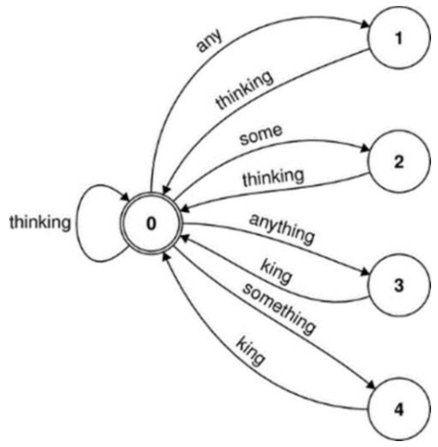


Figure 2.6 - Grammar WFSA, G (edX, 2019).

Next is a WFST that produces the pronunciation lexicon, L, which maps phoneme sequences to word sequences.

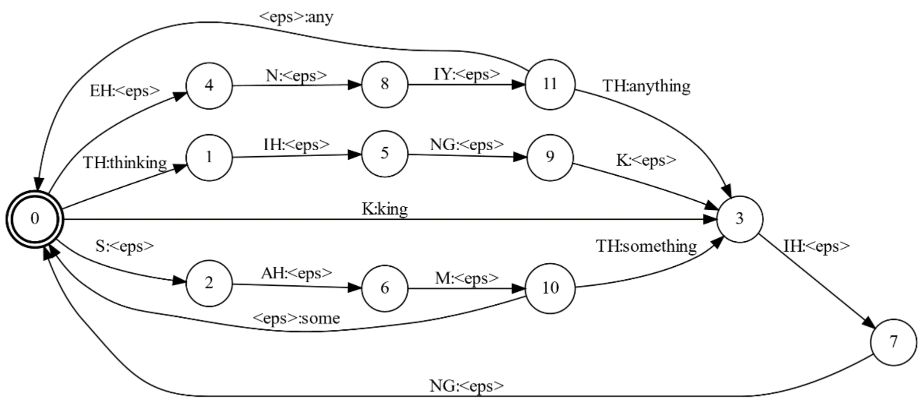


Figure 2.7 - Pronunciation Lexicon, L (edX, 2019).

There is then an HMM FST, H, that maps sequences of HMM states to HMM model phones (either monophonic or triphonic).

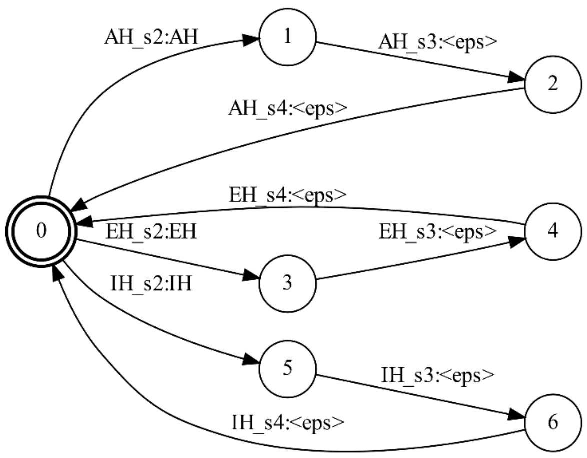


Figure 2.8 - HMM transducer, H (edX, 2019).

Finally, when employing triphonic models there is a context WFST, C, which links sequences of triphones to sequences of single phonemes.

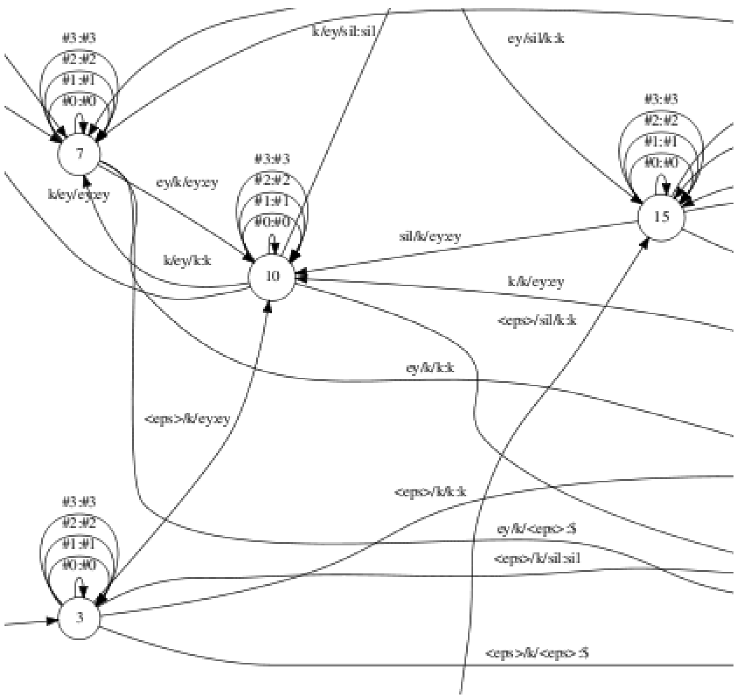


Figure 2.9 - Excerpt of a context WFST, C (Panayotov, 2012).

The final product appears as below, although without the more advanced C component (note that “eps” represents an epsilon, which is an ignored tag and has no effect on the state or its transition):

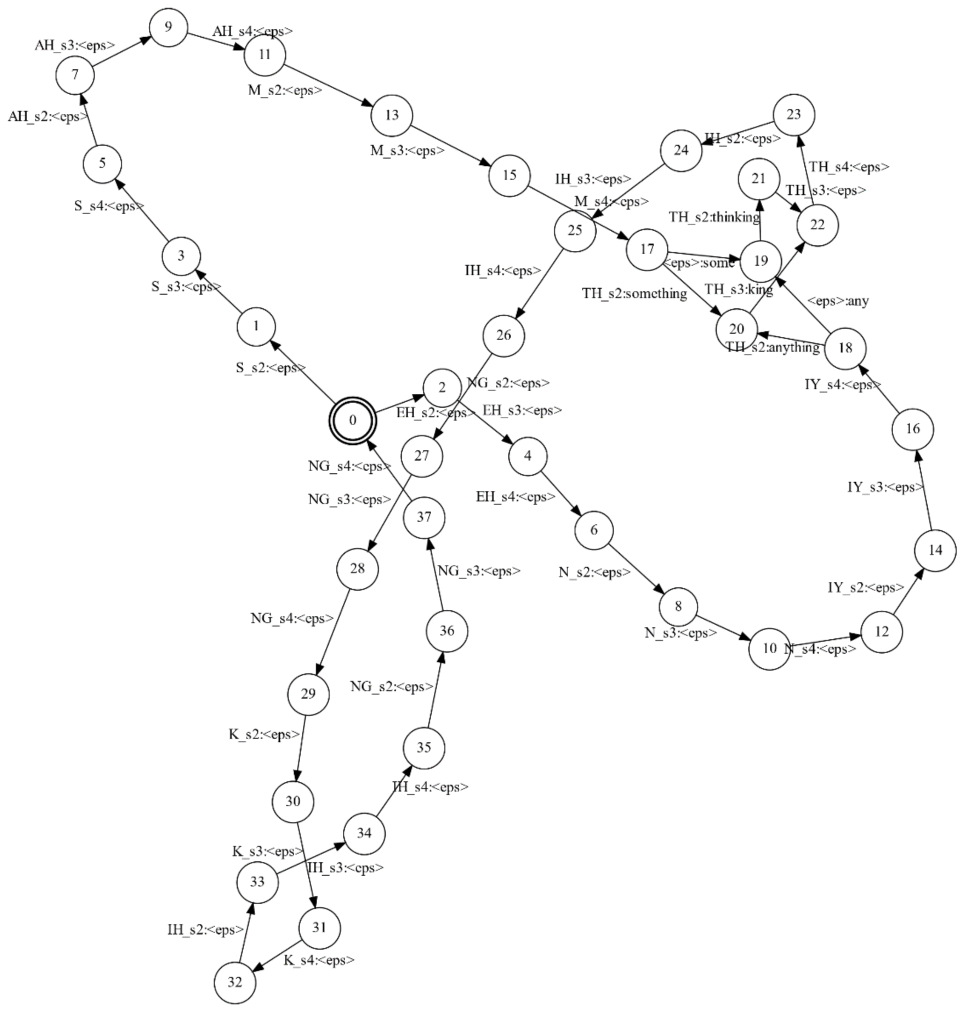


Figure 2.10 - Example of an HLG WFST decoding graph for ASR (edX, 2019).

The collection of these finite-state automata yields an HCLG decoding graph that is created when the model is trained. This graph can then use the processes of determinization, which combines paths that use the same input label and removes epsilons at the input, and minimization, which removes redundant paths and creates the most compact representation that still remains accurate to the permutations present in an original FST (Mohri et al., 2002). When determining the correct output transcription, an ASR model will use a decoding graph like the one pictured above and attempt to piece together the sequence that has the highest scores in respect to both the acoustic and language models present in the HCLG graph. The highest scoring transcription will be chosen by the model and then compared to the true transcription in a test set in order to produce error rates.

2.4 ASR System Enhancements

While the basic system for generating acoustic models using feature vectors serves as an effective foundation for creating competent models, there are greater improvements that have been observed to further increase the distinguishability between feature vectors and to reduce feature vector dimensionality without great loss. It has been shown that the deployment of class labels, similarly to their use in *N*-gram language models, permits the use of more advanced transforms in order to condense and enhance feature vectors. This is in contrast to an unsupervised approach to feature analysis where only overall attributes, such as the variances of the observations, may be used to the same effect (Gales and Young, 2007). Furthermore, studies have found that sorting extracted feature vectors into component classes was more effective than sorting by phone or word units. Organization by component class allows ASR models to utilize the technique of Linear Discriminate Analysis (LDA). This process increases the ratio between the variances between different classes and the average variance per dimension within classes. This ratio can be expressed using the variance between classes $\tilde{\Sigma}_b$ and variances within classes $\tilde{\Sigma}_w$ as:

$$\mathcal{F}_{\mathtt{lda}}(\boldsymbol{\lambda}) = \log\left(\frac{|\mathbf{A}_{[p]}\tilde{\boldsymbol{\Sigma}}_{\mathtt{b}}\mathbf{A}_{[p]}^{\mathsf{T}}|}{|\mathbf{A}_{[p]}\tilde{\boldsymbol{\Sigma}}_{\mathtt{w}}\mathbf{A}_{[p]}^{\mathsf{T}}|}\right)$$

Equation 2.11 - LDA Ratio (Gales and Young, 2007).

The variances are also paired with Gaussian components here. This yields a diagonalized covariance matrix, thus decorrelating the feature vectors. There exists a Heteroscedastic LDA (HLDA) that outperforms LDA by removing identical elements between feature vectors, further increasing feature vector decorrelation, but this process can be computationally intensive and oftentimes LDA paired with Maximum Likelihood Linear Transformation (MLLT) serves as the most effective method for enhancing the acoustic features of an ASR system. MLLT computes the correlation between feature vectors through the application of a linear transform to the covariance matrix generated by LDA. This process is applied at specified iterations in the model training cycle and creates its own transformed covariance matrix. This matrix then updates the model means and is combined with the LDA matrix on every pass. The combined LDA + MLLT processes create heavily decorrelated feature vectors that can efficiently compete with HLDA.

On top of feature vector optimization, model parameters can also be more finely tuned in order to optimize performance. For example, while many basic ASR models rely on enhancing maximum likelihood calculations in order to strengthen an HMM's ability to arrive at accurate, probabilistic decisions, there are alternatives for building stronger models that rely on parameters referred to as discriminative criteria (Gales and Young, 2007). The criteria do not aim to just create a model that is able to arrive at the best word sequences to match the training data, but they also seek to assure overall accuracy when hypothesizing word sequences as well as tuning model parameters so that they are able to effectively analyze unseen test data. These techniques create alternative methods of creating stronger models that do not just involve fitting expected training inputs to expected training outputs.

One of these discriminative approaches is the use of Maximum Mutual Information (MMI) calculation. MMI seeks to maximize the mutual information shared between a target word sequence and a recognizer output. However, the probability distribution between word sequences and recognizer observations is unknown, so an empirical distribution function is calculated using the training data. This process then uses these distributions to find the acoustic model parameters that provide the greatest average log-posterior probability of generating the correct word sequence. This criterion is maximized as follows for an observed sequence $\boldsymbol{Y}$, target word sequence $\boldsymbol{w}$, and model parameters $\boldsymbol{\lambda}$:

$$\begin{aligned}\mathcal{F}_{\texttt{mmi}}(\boldsymbol{\lambda}) &= \frac{1}{R}\sum_{r=1}^{R}\log(P(\boldsymbol{w}_{\texttt{ref}}^{(r)}|\boldsymbol{Y}^{(r)};\boldsymbol{\lambda})) \\ &= \frac{1}{R}\sum_{r=1}^{R}\log\left(\frac{p(\boldsymbol{Y}^{(r)}|\boldsymbol{w}_{\texttt{ref}}^{(r)};\boldsymbol{\lambda})P(\boldsymbol{w}_{\texttt{ref}}^{(r)})}{\sum_{\boldsymbol{w}} p(\boldsymbol{Y}^{(r)}|\boldsymbol{w};\boldsymbol{\lambda})P(\boldsymbol{w})}\right)\end{aligned}$$

Equation 2.12 - MMI function criteria (Gales and Young, 2007).

The numerator represents the likelihood of the data in respect to the correct word sequence $\boldsymbol{w}_{ref}^{(r)}$ and the denominator is the likelihood of data given all possible word sequences $\boldsymbol{w}$ (Gales and Young, 2007). These both take the form of HMMs and denominator lattice statistics are calculated separately for all utterances in the training set. The numerator is effectively optimized in order to make model outputs that match the correct word sequence as likely as possible while reducing the likelihood of observed data in respect to all other sequences. MMI can also be modified with a boost by multiplying the denominator by $e^{-bA(w,wr)}$. b represents a scaling factor that is usually 0.5 and $A(w,w_r)$ is the accuracy of word sequence w given reference sequence w_r (Povey et al., 2008). This serves the purpose of increasing the likelihood of sequences with more errors, softening the margin of models created using MMI techniques. This helps to prevent overly rigid models from overfitting training data and thus lacking in flexibility when they are exposed to unforeseen speakers and audio inputs.

Another popular discriminatory technique is Minimum Phone Error (MPE) training. MPE training is an MBR technique that seeks to minimize the loss function between observed and target phoneme units when training an ASR model. This is calculated as a Phoneme Error Rate (PER), similarly to Word Error Rate (WER). WER is a common standard for calculating the error loss of an ASR model (edX, 2019). It is equivalent to the sum of all insertions (words added that were not previously present in a transcription), deletions (target words that are missing from a transcription), and substitutions (words substituted in the place of target words) divided by the total number of words in the intended word sequence.

$$WER = \frac{N_{\text{sub}} + N_{\text{ins}} + N_{\text{del}}}{N_{\text{ref}}}$$

Equation 2.13 - WER Formula (edX, 2019).

This algorithm holds true with phonemes in place of words for PER. Either of these metrics as well as a handful of others may be used in MBR optimization, depicted using a loss function $\mathcal{L}(\boldsymbol{w}, \boldsymbol{w}_{ref}^{(r)})$ of observed word sequence $\boldsymbol{w}$ compared to reference sequence $\boldsymbol{w}_{ref}^{(r)}$:

$$\mathcal{F}_{\mathtt{mbr}}(\boldsymbol{\lambda}) = \frac{1}{R}\sum_{r=1}^{R}\sum_{\boldsymbol{w}} P(\boldsymbol{w}|\boldsymbol{Y}^{(r)};\boldsymbol{\lambda})\mathcal{L}(\boldsymbol{w}, \boldsymbol{w}_{\mathtt{ref}}^{(r)})$$

Equation 2.14 - Function for optimizing MBR (Gales and Young, 2007).

When multiple models trained using discriminatory criteria were evaluated using the Wall Street Journal datasets, a group of speech databases containing varying amounts of speech samples, it was found that MPE greatly outperformed ML-optimized systems and marginally improved on MMI-based systems, particularly for small vocabularies (Macherey et al., 2005).

2.5 Adaptation and Normalization Practices for ASR

The fundamental problem with designing effective ASR models is that models will inevitably be introduced to speakers that do not originate from the original training and test sets, as these may be meant to be deployed to a wide range of personnel such as medical professionals or mobile phone users. In order to counteract this, there are systems in place that serve to normalize and adapt to new speakers. This is often implemented by obtaining preliminary samples of new users' speech to allow an ASR model to incorporate their individual speech patterns into the model's training data.

Some of these techniques focus on refactoring acoustic features to increase robustness. One such method is the use of Cepstral Mean and Variance Normalization (CMVN). This process is a combination of Cepstral Mean Normalization (CMN), which removes the mean value from feature vectors, and Cepstral Variance Normalization (CVN), which scales each feature coefficient to have a specified unit variance. The former technique helps to reduce the effect of speaker variation on the model transcription and the latter serves to increase robustness to unwanted noise (Gales and Young, 2007). Vocal Tract Length Normalization (VTLN) is another way to mitigate the effect of exposing trained ASR models to new speakers. This form of processing is inspired by the idea that as the length of the vocal tract changes, the formant frequencies that comprise the voice are linearly transposed. Once a speaker has been identified and sampled, the filter bank center frequencies are linearly scaled to adjust for the individual speaker during feature extraction. This linear transform must often be slightly modified in order to prevent skewing the frequency bandwidths of males and females too heavily. For example, if a basic linear transform is applied then female speakers will lose high frequency content and male speakers will suffer from over-emphasized high frequencies.

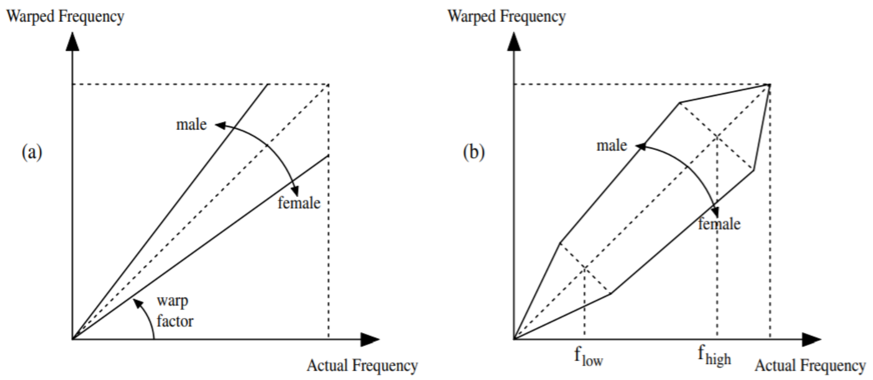


Figure 2.11 - (a) VTLN applied with standard linear transform. (b) VTLN with preferred, piecewise linear transform (Gales and Young, 2007).

Maximum Likelihood Linear Regression (MLLR) methods offer a more complex and robust way to adapt ASR features to unexpected speakers. MLLR uses linear transforms to create a copy of an existing model that optimizes the likelihood of new adaptation data. This is done by applying transforms to the Gaussian components of the data, the mean and variance. MLLR is unconstrained if transforms are calculated and applied separately for the mean and variance. However, it is more typical to use constrained, or feature space, MLLR (CMLLR or FMLLR), which aims to apply a linear transform to all the feature vectors' Gaussian components. This feature transform can be very efficient and allow a model to retain its parameters while flexibly adapting to new speakers and acoustic environments (Gales and Young, 2007). Unsupervised MLLR is the most common application of this process as it is often not known what speaker data an ASR model will be exposed to. In this case, MLLR iteratively estimates transforms based on a new hypothesized transcription for an unknown speech input. Models are then adapted using this transform and the process repeats until the model and the target transform converge, leading to the maximum likelihood of outputting the target transcription (Gales and Young, 2007). This provides an effective way of on-the-fly adaptation that can effectively provide a degree of malleability for an ASR model that may encounter speaker inputs that greatly vary from its original training data.

There also exist approaches that use standard statistical techniques in lieu of calculating transforms that account for speaker differences. Maximum A Posteriori (MAP) adaptation is a popular method of employing statistical analysis to inform better adapted models. MAP adaptation seeks to incorporate a prior over a model's parameters with its adaptation data in order to create models that are fit to new speakers. This is performed by adding a prior calculation of model parameters $\boldsymbol{\lambda}$ as $p(\boldsymbol{\lambda})$ to a previously calculated ML estimation as follows:

$$\mathcal{F}_{\mathtt{map}}(\boldsymbol{\lambda}) = \mathcal{F}_{\mathtt{ml}}(\boldsymbol{\lambda}) + \frac{1}{R}\log(p(\boldsymbol{\lambda}))$$

Equation 2.15 - MAP adaptation for an ASR model (Gales and Young, 2007).

This form of adaptation essentially performs an interpolation between adaptation data and a calculation of priors over the model parameters. As adaptation data increases, the model more closely approximates the adaptation data itself, which makes MAP an effective way over transferring a pre-existing model over to a new domain that is lacking in data of its own (Gales and Young, 2007). However, if there is very little adaptation data MAP computation will be unable to update model parameters, making this process a poor choice for models that require quick adaptation.

While at first it may seem ideal to simply pour more and more training data into the training dataset for an ASR model, this may sometimes complicate the task as the ASR model has to effectively create accurate classes to properly distinguish between different words. In order to compensate for any number of speakers, as the size of a training set increases, the number of speakers present in the dataset also increases. This causes the model to have to calculate and comprehend variances between a multitude of speakers before it can even approach the problem of categorizing different words. Speaker Adaptive Training (SAT) aims to address this by applying adaptation transforms that represent the speakers, removing the need for expensive preliminary analysis of the speakers by the model training parameters. SAT commonly uses FMLLR to estimate transforms that are trained on each individual speaker within a training set. These transforms are then summed into the simplest model representation possible that faithfully incorporates the components of each individual speaker model. The mean for this canonical model is calculated using the following equation:

$$\hat{\boldsymbol{\mu}}^{(jm)} = \frac{\sum_{s=1}^{S}\sum_{t=1}^{T^{(s)}} \gamma_t^{(sjm)}\left(\mathbf{A}^{(s)}\boldsymbol{y}_t^{(s)} + \mathbf{b}^{(s)}\right)}{\sum_{s=1}^{S}\sum_{t=1}^{T^{(s)}} \gamma_t^{(sjm)}}$$

Equation 2.16 - Mean for a canonical model derived using SAT (Gales and Young, 2007).

In this equation, S is the number of speakers, $\gamma_t^{(sjm)}$ is the posterior occupation probability of the component m in HMM model state s_j generated at time t based on data for the speaker s (Gales and Young, 2007). While CMVN and VTLN are completely model independent transformations and FMLLR is a feature transformation that is derived using model estimation, SAT is a model transformation that works purely on optimizing model parameters.

3 Feature Extraction

3.1 Overview

Feature extraction is the process of translating audio information into more digestible data vectors that can be seamlessly interpreted by HMM's in order to form the foundation of the acoustic model. An effective feature extraction stage has several clearly stated goals. The choice of feature extraction algorithm should be capable of easily discerning speech inputs, should not be vulnerable to mimicry by multiple or false speakers, should be sufficiently robust in the contexts of different noise floors arising out of different speaking environments, and should be consistent over time in its use of natural language input (Vimala and Radha, 2012). There is a great amount of debate surrounding different approaches to designing feature extraction to have these qualities.

Several different feature extraction approaches share common processes that are generally regarded as beneficial for the speech signal, such as the use of a pre-emphasis filter as a preliminary step to accentuate the high-frequency spectrum of the input speech as this frequency bandwidth (typically above 1 kHz) is more useful in analyzing the human voice. The signal would then be sampled into frames and a Hamming window applied in order to reduce sample distortion created by the framing process (Madisetti, 2010). Frames are typically computed every 10 ms using an overlapping analysis window of 25 ms. The filter coefficients of a Hamming window of length *n* are computed as follows (*M* is the total number of samples and *n* is the current sample):

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \le n \le M-1$$

Equation 3.1 - Hamming Window (SciPy, 2017).

A Fast Fourier Transform (FFT) is then used to convert the speech signal from the time domain to the frequency domain, which is a much more effective domain in the context of performing an analysis of the frequency content contained in the speech input. It has also been shown that far greater accuracy can be derived from the use of features in the frequency domain as opposed to the time domain (Dave, 2013). A Fourier transform of a signal x_n in *N* total samples is given by the following:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}kn}, \quad k = 0,1,\ldots,N-1.$$

Equation 3.2 Fourier Transform (Jansi et al., 2019).

An FFT would consist of the same transformation above only applied in portions of total samples *N/2* to any requested depth and then processed in parallel in order to speed up computation.

A new problem becomes apparent once the speech input is placed into the frequency domain: the input data intended for the acoustic model cannot include the entire frequency bandwidth of the speech signal in its data vectors. The data resolution would be overwhelming for input into HMM's as it is possible to have nearly 20,000 different frequencies that contribute to the human voice signal (Markowitz, 1996). To complicate matters, a large part of the sound input may also contain extraneous noise that only serves to hinder the identification of the speech signal. However, we do recognize that the human ear must confront the same challenges in understanding speech and yet contains the incredible ability to consistently sort out noise and unwanted inputs in order to target specific human voices. In order to attempt to emulate the uncanny ability of human hearing to effectively interpret speech in a variety of environments, it is typical to use either Linear Predictive Coding (LPC) or filter bank approaches akin to the processing that occurs in the human cochlea/neural auditory pathways (Gerazov and Ivanovski, 2012).

3.2 The Original Feature Extraction Technique: LPC

LPC was one of the earliest feature extraction methods to be widely deployed for speech recognition. It is still recognized as a highly efficient technique that is well suited for input into low or medium bit rate decoders (Dave, 2013). LPC aims to output a power spectrum of the input signal and is typically used for formant analysis and formant estimation. The technique originated in the 1960's and aimed to use the Vocal Tract Transfer Function (VTTF) with all-pole filters applied to time-sampled audio frames, reminiscent of the source-filter model approach to vocal formants. This is ideal as the VTTF provides substantial insight into the foundation of phonation, pairing well with the tendency of ASR acoustic models to depend on phonetic units (Gerazov and Ivanovski, 2012).

In order to create LPC features, the input signal is first pre-emphasized, sampled with a smoothing window, and then a Hamming window is applied to these frames. The frames are then interpreted through VTTF all-pole filters, typically ranging between 8th and 10th order. Uniquely, LPC has several choices of coefficient representation, ranging from linear prediction, reflection or partial correlation (PARCOR) coefficients, to log area ratio or even cepstral coefficients (Gerazov and Ivanovski, 2012). Cepstral coefficients are the preferred output, resulting in Linear Prediction Cepstral Coefficients (LPCCs). The cepstral coefficients are a log compressed magnitude spectrum form of the LPC processed audio samples. Although this technique is still widely used for its computational efficiency and flexibility, it led to more advanced feature representations that were targeted towards creating a simulation of the human cochlea in order to yield the most effective speech signal interpretations. One of the most popular feature extraction methods, MFCCs, was derived from this new conceptual direction for ASR.

3.3 Filter Bank Methodology

The most popular form of filter bank method currently in use is the employment of MFCCs. The foundation for MFCC is the Mel scale, a perceptual scale that was derived from a series of experiments in 1937 (Volkmann et al., 1937). The experiments aimed to create a scale that was entirely subjective as opposed to the more standardized musical and frequency scales. Volkmann (1937) enlisted five observers who reported the perceived magnitude of several frequencies that were all played back at 60 dB. 1000 Hz was used as a center frequency, becoming the equivalent of 1000 mels. Human hearing is now known to be logarithmic in nature along the frequency domain, and so the Mel scale that was derived was naturally a logarithmic scaling of magnitudes from low to high frequency. However, the scale is more linear up to 1000 Hz, after which the amplitudes transition to a more logarithmic response (Madisetti, 2010). These experiments were meant to draw subjective conclusions about the basilar membrane response to a variety of different pitches; this is of course appealing to speech recognition efforts as research commonly revolves around modelling human hearing response in order to create the most effective analysis of speech signal input.

The process for converting audio input into MFCC features serves as the primary precedent for filter bank feature approaches. First, an incoming speech signal is passed through a first-order pre-emphasis filter in order to accentuate higher frequencies. The signal is then sampled into time frames that are dependent on the system sampling rate. This rate can be typically 16 kHz or even 8 kHz as these sample rates are deemed appropriate due to their respective Nyquist frequencies of either 8 kHz or 4 kHz, both of which are permissible for the necessary bandwidth of a speech signal (a large amount of speech formants take place from 1 kHz-5kHz and most information above 3 kHz can be seen as extraneous). The time frames are then smoothed, and a Hamming window is applied to the time frames. The time frames are converted into a frequency domain representation using an FFT. The output frames of the Fourier transform are placed into the Mel scale using the following algorithm:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

Equation 3.3 - Frequency to Mel Scale (O'Shaughnessy, 1987).

The resulting scaled output is logarithmically compressed in order to create wider bandwidths for higher frequencies and greater resolution at lower frequencies. The Mel filter bank consists of 40 bins that each contain a data vector representation of the amplitudes at their respective frequency bandwidths. A plot of the Mel scale is shown below.

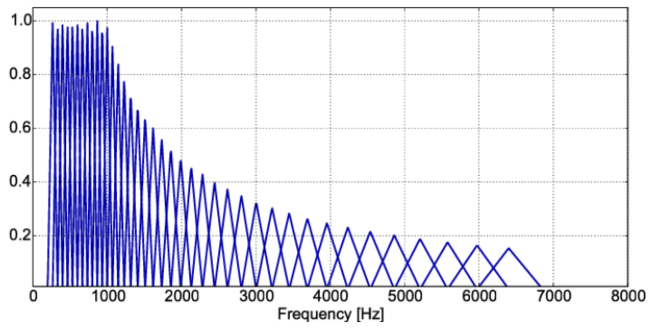


Figure 3.1 - Mel filter banks in frequency domain (Maka, 2015).

The final step of this process is the performance of a discrete cosine transform (DCT) of the Mel scaled audio, performed by the following equation for input signal x_n:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \dots, N-1.$$

Equation 3.4 - Discrete Cosine Transform (Narasimha and Peterson, 1978).

This processing helps to further reduce the amount of extraneous data as this omits the phase information of input audio so that only the frequency information remains, resulting in a cepstral representation. Cepstral transformation has been shown to largely remove speaker excitement from the sound signal, thus cancelling some of the effects of unwanted speaker variation (Madisetti, 2010). This data is then sent on to the speech recognizer for use in creating the acoustic model. See the figure below for an illustration of this process.

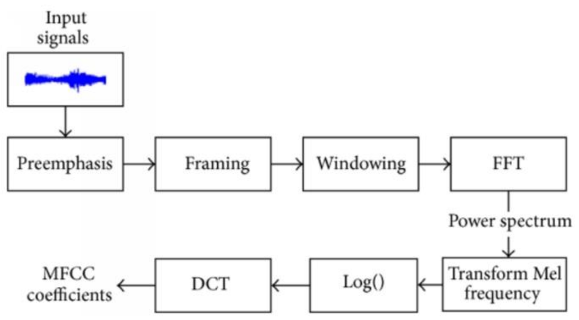


Figure 3.2 - MFCC Processing Pipeline (Gong et al., 2015).

MFCC has seen popular usage in speech recognition system design for its well-regarded compromise between computational efficiency and accurate speech recognition (Chavan and Sable, 2013). However, it is by no means the only option for frequency scaling and in fact has some significant competition by other well-proven methodologies.

3.4 Alternatives to MFCC

3.4.1 GFCCs

Gammatone Frequency Cepstral Coefficients, or GFCCs, are an alternative to MFCCs that have been the subject of much research exploration. GFCCs opt to use Gammatone filter banks rather than Mel scale filter banks (see figure below).

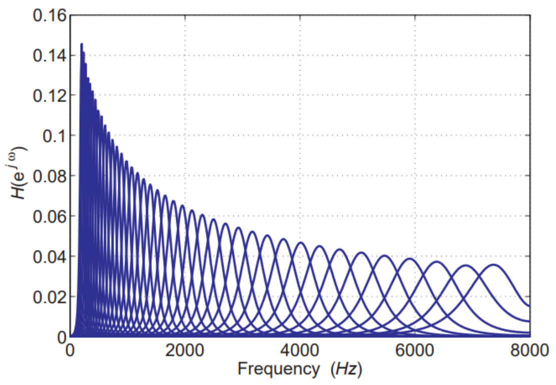


Figure 3.3 - Gammatone filter banks in frequency domain (Tamazin et al., 2019)

The Gammatone scale is based off a model of the auditory filters used in the human auditory system. It is meant to bear a close resemblance to the observed effects of cochlear filtering. A Gammatone filter bank is the equivalent of a time-frequency representation of a cochleagram (Al-Noori et al., 2016).

The recipe for processing GFCCs is reminiscent of the approach for MFCCs with some major differences. All the steps preceding the filter bank transformation are identical (pre-emphasis, time-sampling, and Fourier transform). However, once the samples in the frequency domain are to be placed into filter banks, they are now converted using the algorithm for the Gammatone scale which is as follows:

$$g(f,t) = \begin{cases} t^{(a-1)e^{-\pi bt}\cos(2\pi ft)} & ,t \geq 0 \\ 0 & ,t < 0 \end{cases}$$

Equation 3.5 - Algorithm for Gammatone filtering (Al-Noori et al., 2016).

In this equation, *t* represents time, *a* is the filter order (typically 4th order in this context), and *b* is the rectangular filter bandwidth. The Gammatone filter bank typically consists of 64 bins that range from 80 Hz to 8 kHz and are positioned based on the Equivalent Rectangular Bandwidth (ERB) scale (Al-Noori et al., 2016). The ERB scale is a psychoacoustic measure that attempts to approximate the filters in human hearing as a series of rectangular bandpass filters (Semantic Scholar, 2019). In order to convert from Hz to the ERB scale, the following transformation is used for frequency f:

$$\mathrm{ERBS}(f) = 21.4 \cdot \log_{10}(1 + 0.00437 \cdot f)$$

Equation 3.6 - ERB scale conversion (Smith and Abel, 2007).

Each filter bin is then decimated to 100 Hz in the time domain and an absolute value is taken (Al-Noori et al., 2016). While this decimation may hamper the resolution of the audio signal and thus affect the accuracy of information that can be pulled from these data vectors, this may help to computationally balance the greater number of computations the GFCC process produces. Gammatone conversion is both more complex and more numerous in respect to the amount of filter bins as compared to Mel scale conversion, so this may be a necessary compromise to produce better efficiency.

Typically, logarithmic compression is employed as the final phase of GFCC extraction, but research sometimes opts to use cubic-root compression as a means of differentiation from the standard MFCC technique (Al-Noori et al, 2016; Qi et al., 2013). This cubic root compression is often referred to as an application of an intensity-loudness power law, meaning it aims to emulate the power law of hearing (Missaoui and Lachiri, 2014). A DCT is then applied in order to output cepstral coefficients as in the case for MFCCs.

In order to quantify ASR model performance using different feature extraction techniques, the US National Institute of Standards and Technology Department (NIST) has recommended the use of the Equal Error Rate (EER) method as a potential metric (Al-Noori et al., 2016). This rate is determined by observing the point where the False Acceptance Rate (FAR) and False Rejection Rate (FRR) is equal. The Detection Error Trade-off (DET) is a graph that displays the FAR on the y-axis and the FRR on the x-axis in percentile form which can be used to corroborate the EER point as well as providing a visual aid for the statistical analysis of an ASR system's performance.

Al-Noori (2016) conducted an experiment that utilized speech samples from two separate databases, one that predominantly used American English samples (TIMIT) and another dataset that contained multiple languages (SALU-AC). Both datasets were also corrupted with a variety of different noise floors (street noise, car noise, etc.) in order to test the noise robustness of these feature extraction methods in more realistic contexts. It was concluded that using GFCC was an improvement to using MFCC, particularly in the context of noise rejection. Researchers observed that as noise levels increased (quantified by signal-to-noise ratio or SNR) MFCC began to have much higher EER rates than GFCC, even to the point of a 20% disparity between the two, in favor of GFCC (Al-Noori et al., 2016). However, when using the SALU-AC dataset, MFCC performed better than GFCC, representing an advantage when interpreting multiple languages. MFCCs outperformed GFCCs EER with multilingual samples by as much as 9% in some instances.

It was hypothesized that the mathematical differences between GFCC and MFCC had the largest influence on the performance differences (Al-Noori et al., 2016). Since this GFCC technique used a cubic root rectification method, thus retaining a linear response for a larger frequency bandwidth in the lower frequencies, it was better equipped to reject non-linearities like noise that were present in noisy speech samples better than ASR models using MFCCs. This also put GFCCs at a disadvantage in some situations, as evidenced by GFCC models' inability to handle the non-linearities of multiple languages, which tended to favor MFCC's ability to better accept non-linearities. This serves as a more complex example that illustrates that subtle differences in feature extraction approaches yielded different speech recognition behaviors. It is not immediately clear if all systems would derive a benefit from using GFCC over MFCC methods, although systems that target a distinctly unilingual audience could benefit from the improved noise-rejection that GFCCs display and would minimize the disadvantages of GFCC in terms of its weaker ability to recognize languages outside of its training set.

3.4.2 PLP

Numerous studies have found that Perceptual Linear Predictive (PLP) analysis has some quantifiable advantages over MFCC. PLP is based off the classic LPC approach to ASR, which was mainly centered on calculations of vocal tract transfer functions (Gerazov and Ivanovski, 2012). It is called predictive coding because the analysis method approximates an incoming speech sample as a product of its previous samples, in a way "predicting" what the next sample should be.

However, although LPC is considered to be computationally efficient and effective for conversion into other types of coefficients if necessary, its approach to vocal tract modelling is flawed due to the fact that it displays more frequency information than is necessary and is often not concise enough for the most effective input into a neural network (Madisetti, 2010). Thankfully, PLP is a continuation of this approach but with greater refinement. PLP aims to more closely model the way we hear rather than the way we speak, so it features a roll off of frequencies below 800 Hz and accentuates the presence range of 1 kHz - 6 kHz by incorporating the Bark scale, a frequency scale that is logarithmic after 500 Hz rather than the 1000 Hz of the Mel scale (Gerazov and Ivanovski, 2012). This method employs a similar approach as the Mel scale by modelling human hearing using the Bark scale. It combines this with the advantage of using the autocorrelation processing of LPC which can help to more effectively cancel out acoustic energy present in the source that does not correlate to the original speech signal, such as noise (Gerazov and Ivanovski, 2012). For this reason, it has been found that PLP can be more desirable for feature extraction than MFCC as it uses a similar frequency scaling to model human hearing and it encapsulates computational methods that are meant to increase noise robustness.

Creating PLP features involves a slightly altered process from the processes that have been discussed previously. First an FFT is taken of an input speech signal. Then the power spectrum derived from the frequency domain signal is discretely convolved with a piece-wise approximation of the critical band curve. Critical bands are meant to represent the auditory filters created by the cochlea and are calculated in this context from a conversion of the frequency scale to the Bark scale. The conversion from Hz to Bark values can be computed using the following equation:

$$\mathrm{Bark} = [(26.81f)/(1960 + f)] - 0.53$$

Equation 3.7 - Bark scale conversion (Traunmüller, 1990).

Pre-emphasis is then applied to this rescaled audio and cubic-root loudness compression is used to normalize the samples. The final phase of PLP feature extraction involves autoregressive modelling. An inverse discrete Fourier transform (IDFT) is used to derive autocorrelation values, an approach preferred to an inverse FFT as only a select number of autocorrelation values are required. The first *M+1* autocorrelation values are then used in the Yule-Walker equation to derive autoregressive coefficients for an *Mth* order all-pole model. The resulting autocorrelation coefficients can be transformed into a variety of different formats, such as applying a DCT to derive the cepstral coefficients as in other traditional feature extraction techniques (Hermansky, 1990).

PLP contains great promise due to the combination of several elements that have seen success in other feature extraction methods. It combines a perceptual scale akin to that seen in MFCCs, the linear prediction techniques of LPC, and the cubic root compression that may have contributed to the success of a variant of GFCCs. It can also be combined with other computational methods in order to further increase the accuracy of this technique.

A research study found that when combining Relative Spectra Filtering (RASTA) with PLP, ASR systems were observed to outperform other systems that used MFCC. This is due to PLP's greater smoothing of input signals (since it features a logarithmic response sooner than MFCC) and the reduction of linear distortion and noise thanks to the use of RASTA filtering with PLP (Shrawankar and Thakare 2013). RASTA filtering aims to emulate the ability of human hearing to largely ignore slowly moving stimuli, such as a steady room tone, in order to better filter out noise. When this filtering is applied to PLP feature extraction, it replaces the typical critical-band analysis that forms the spectral characteristics of PLP. This is instead replaced with a series of bandpass filters that contain a sharp spectral zero at the center frequency, which suppresses constant or slow-moving components in each of the filtered frequency bands (Hermansky and Morgan, 1994). This filter can be implemented as an Infinite Impulse Response (IIR) filter with the following transfer function:

$$H(z) = 0.1z^4 * \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}}$$

Equation 3.8 - Transfer function of a RASTA filter (Hermansky and Morgan, 1994).

It is even more informative to view the response of the filter in the frequency domain in order to better grasp how it transforms the typical bandpass implementation in PLP features:

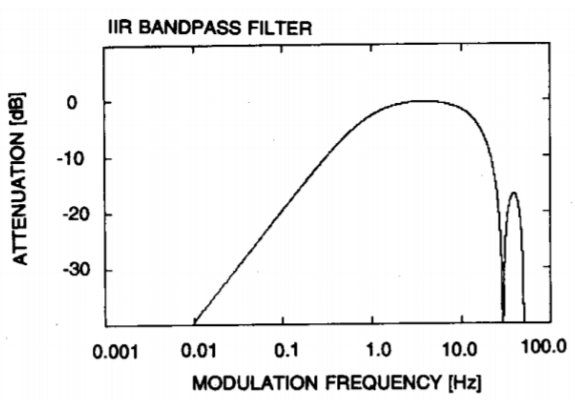


Figure 3.4 - Frequency response of a RASTA IIR bandpass filter (Hermansky and Morgan, 1994).

RASTA filtering was able to improve standard PLP techniques by as much as 40% when testing an ASR model with a dataset of isolated digits with additive noise and filtering. This is a dramatic improvement and proves PLP is not only robust on its own but also has the ability to adapt extremely effective measures to further increase noise robustness.

These findings were confirmed by another study that used a quantitative measure called Word Recognition Accuracy (W_{ACC}) to compare differing feature extraction methods tested on multiple language samples (Sárosi et al., 2011). Word Recognition Accuracy can be obtained by subtracting the WER from 100%. The results showed that PLP outperformed all variations of MFCC, even one variation that used a unique adaptive normalization technique called BEQ (Blind Equalization). This superior performance was once again attributed to PLP's approach to frequency band analysis, which led to greater noise rejection and more accurate recognition (Sárosi et al., 2011). However, there was another approach that led to even better overall results: the inclusion of Power Normalized Cepstral Coefficients, or PNCC.

3.4.3 PNCC

The use of PNCC in ASR systems has been found to offer marked improvement over MFCC in terms of accuracy and usability. In a study that demonstrated the gradual improvement over MFCC as the various parts of a PNCC process were integrated, it was found that the use of a power-law rate-intensity function over the use of logarithmic non-linearity in MFCC had the most influential impact on the observed superior performance of ASR models that employed PNCC (Kim and Stern, 2016).

There are several facets of the power-law approach that help to substantiate this claim. Power-law nonlinearity models are meant to mimic the non-linear relationship between true signal intensity, or power, and auditory nerve firing-rates in humans (Kim and Stern, 2016). Mathematically the process is cleaner as there is an analysis of the magnitude of the signal source, rather than a multiplication of the input signal by a constant in the case of conversion to the Mel scale. Low signal level points are seen as a zero rather than negative infinity and so PNCC is able to handle quieter signal levels with less variation than MFCC, which has to compute frequency scaling using complex numbers due to this mathematical behavior. This helps to reduce the influence of small signal fluctuations when analyzing a sound input. It has also been found that power-law modeling of audio better resembles how we psychoacoustically register physical sound sensation as it is a representation of magnitude (Kim and Stern, 2016).

PNCC is calculated similarly to MFCC in most respects: it still features a traditional pre-emphasis step and seeks to scale sound inputs into a more digestible data vector. PNCC does however incorporate some more sophisticated approaches in order to best model input signals. PNCC uses the Gammatone frequency scale rather than the Mel frequency scale as a filter bank, similarly to GFCC, which helps to bring its modeling closer to that of the response of our cochlea (Sárosi et al., 2011). Another significant addition of PNCC is its use of a medium time frame. MFCC and PLP often use shorter windows of 20-30 ms in order to limit the influence of other input signals as speech input is thought to happen in a much shorter time window than a noise floor or other non-essential audio elements. A window of 50-120 ms is used when employing PNCC as it is actually considered more useful to incorporate environmental noise factors through this larger time frame in order to better analyze and effectively cancel out their effects on the speech signal. Extraneous noise and unwanted sound signals can then be properly treated because this medium time frame incorporates more sound information into the extraction process. Asymmetric noise suppression can be applied in order to find an upper envelope of the sound, representing voice input, and a lower envelope of the sound, which represents the noise floor, which can then be subtracted from the voice input, or upper envelope. Low-pass filtering can also effectively be applied to the non-speech parts of the input signal, further attenuating the influence of non-essential sound.

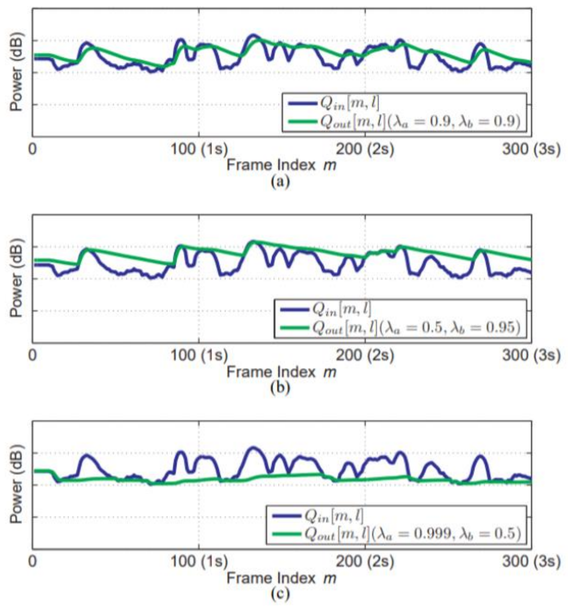


Figure 3.5 – Visualized asymmetric noise suppression: (a) the application of low-pass filtering. (b) the upper envelope. (c) the lower envelope (Kim and Stern, 2016).

The human auditory system has been found to draw more understanding of incoming sound from the attack of a transient than the release and so PNCC can employ temporal masking, a process that attenuates the release of a sound signal (Kim and Stern, 2016).

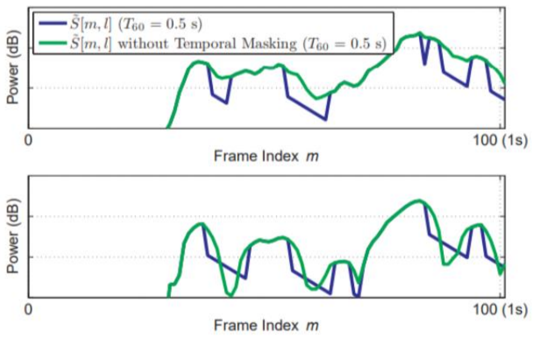


Figure 3.6 – Temporal Masking, the top panel is audio with simulated reverb and the bottom panel is clean audio (Kim and Stern, 2016).

Temporal masking has been found to greatly increase robustness of speech recognition systems against the effects of excessive reverberation, something that can be easily found in most spaces that lack proper acoustic treatment.

PNCC features are calculated by applying pre-emphasis, placing audio samples into overlapped, Hamming windowed frames, performing an FFT, and then scaling the input into the Gammatone scale. Due to the fact that PNCCs employ power function nonlinearities (as processing responses can change due to variations in absolute power) there is a power mean normalization step that takes place in order to prepare the audio samples for cubic root normalization. This is in opposition to the logarithmic nonlinearities that are corrected by MFCC's logarithmic normalization. A DCT is then applied in order to derive cepstral coefficients (Kim and Stern, 2016).

PNCC, thanks to the particularities of its approach to feature extraction, has been shown to greatly improve signal-to-noise performance of sound inputs as compared to MFCC processing. ASR models that employed PNCC feature extraction observed an increase in SNR by up to 13 dB when dealing with white noise and a more modest 3.5 dB in the more challenging situations involving background music and interfering speakers over MFCC (Kim and Stern, 2016).

Unfortunately, this does not come without a cost. The added processing steps can translate to 34.6% more computational power required for PNCC over MFCC extraction (Kim and Stern, 2016). This is only 1.31% more costly than PLP processing, which requires 32.9% more computational power than MFCC, and can be referred to by some as a modest increase in computational needs. However, requiring over 30% more processing than standard MFCC is not something to be completely disregarded and may represent a challenging gap for PNCC to overcome in order to become a mainstay of ASR feature extraction. Speech recognition systems require a lot of efficiency to provide real-time use for consumers and this computational difference may need to be narrowed before PNCC can be successfully employed on a large scale.

4 Feature Extraction Experimentation

4.1 Overview and Motives

This dissertation seeks to confirm the previous findings in the field of feature extraction research for ASR. While it is more typical to generate ASR research that focuses on machine learning modelling and approaches to HMM and DNN architectures, it is of more interest in this research to explore the effect of modifications to the acoustical analysis performed by these systems. It has been stated that oftentimes the use of MFCC and PLP features are assumed and that this portion of the ASR system should not receive more careful thought and attention. However, much of the research that has been assessed above states otherwise: that strong improvements can be observed from subtle differences in processing approaches for the generation of audio features. Many of the feature extraction methods are quite similar with the exception of their choice in frequency scaling, normalization technique, or potential combination with expanded procedures that seek to refine audio feature computation.

4.2 Methodology and Materials

It was decided to test prior research regarding these feature extraction techniques by building several ASR models and quantifying their performance using WER statistics generated when creating and testing models in progress. WER was chosen as it is a standard in quantifying ASR model performance and would provide straightforward insight into how different model performances would compare. Models were trained on a local machine using an NVIDIA Geforce RTX 2080 Ti GPU, Intel i9-9900k CPU clocked at 3.6 GHz, and 32 GB of RAM. Ubuntu 19.04 was used to run all experiments.

Kaldi, an open source C++ library made for prototyping ASR models, was chosen for its relevance to the current ASR industry and its accessibility, support, and ease of use. It is meant to be an updated approach to the Hidden Markov Model Toolkit (HTK) which has long been a standard in open-source ASR prototyping. However, Kaldi is reported to have more modern, robust code and is meant to incorporate many of the newer approaches found in ASR model training (Povey et al., 2011).

Four feature extraction methods were chosen in order to provide a basic test suite of statistics that would quantify techniques that had previously not been gathered together in the same research before. MFCC and PLP were chosen as they are the two default feature extraction techniques that are prepackaged with the Kaldi library. They were thus easy to incorporate and serve as the default methods of choice for the industry. The two methods form a control group for which alternative feature extraction methods can then be tested against as a baseline. PLP was chosen over using the enhanced RASTA-PLP method in order to simplify its employment as well as to measure the performance when using the default approach to the creation of PLP coefficients. RASTA, as well as several other potential processes, are generally regarded as extensions to pre-existing techniques that can definitely improve performance. However, these additional processing steps can be integrated into a variety of other techniques as well, so it served to aim for simplicity in comparing these feature extraction methods on their own and then to later assess potential alterations that can be achieved by the addition of these external processing techniques.

GFCC and PNCC were chosen as two alternative techniques that were meant to confirm the efficacy of using feature extraction methods outside of the normal suite of methodologies. It was hypothesized that the experimental results would follow the precedent set by the previously discussed research; namely that when creating models used clean datasets there would be less of a difference in performance between these methods but that there would be greater differences once ASR models were trained and tested with noisier datasets. This reflects the ability of GFCC and PNCCs to better reject noise, at least in a theoretical basis. However, it was expected that this may come at a loss of computational efficiency as GFCC data vectors are typically larger than in other filter bank methods (64 frequency bins rather than the typical 40) and it had been reported that PNCCs came at a definite computational cost as opposed to techniques that used MFCCs. The chosen techniques are summarized in the chart below, although keep in mind that RASTA-PLP is displayed here and so any of the RASTA processing can be ignored but noted for completeness.

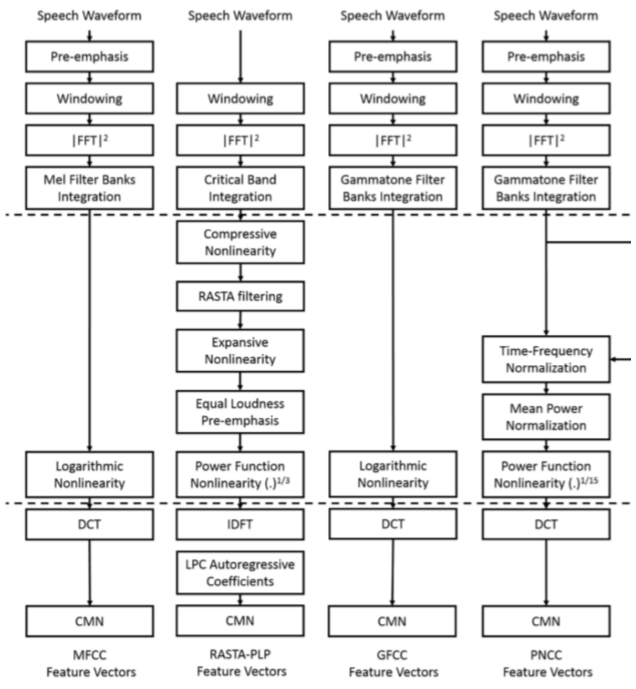

Speech Waveform
Pre-emphasis
Windowing
|FFT|²
Mel Filter Banks Integration
Critical Band Integration
Gammatone Filter Banks Integration
Compressive Nonlinearity
RASTA filtering
Expansive Nonlinearity
Equal Loudness Pre-emphasis
Time-Frequency Normalization
Mean Power Normalization
Logarithmic Nonlinearity
Power Function Nonlinearity (.)^1/3
Power Function Nonlinearity (.)^1/15
DCT
IDFT
LPC Autoregressive Coefficients
CMN
MFCC Feature Vectors
RASTA-PLP Feature Vectors
GFCC Feature Vectors
PNCC Feature Vectors

Figure 4.1 – Summary of feature extraction pipelines chosen for this research; please take note to ignore the RASTA processing for PLP between the critical band integration and the power function nonlinearity in the case of this dissertation research (Tamazin et al., 2019).

4.2.1 Feature Extraction Libraries

In order to generate GFCCs, an open-source library called "featxtra" was used in order to implement GFCC features that were compatible with input as the Kaldi format (Van Segbroeck, 2018). The default GFCC algorithms were used in this library and the decision was made to use the more typical logarithmic compression in order to differentiate from the PNCC approach of employing the power law nonlinearity. The library is primarily focused on providing users with the ability to generate GFCCs but also contains functions to generate Gabor Features (GBF) which serve as another option for alternative feature extraction. It consists of C++ functions and binaries that are meant to easily integrate into the Kaldi C++ format.

A basic Gabor filter is meant to serve as an emulation of the way humans see, it is typically a linear filter that is meant to distinguish visual textures (Fogel and Sagi, 1989). However, when it is used in the spectro-temporal domain it takes the form of a 2D filter. The 2D filter is the result of a convolution between a complex sinusoidal function and a Gaussian envelope. This creates a complex filter that contains both real (Gaussian function) and imaginary (sinusoidal function) components in order to represent an input signal (Meyer et al., 2011). The computation of Gabor features begins similarly to PLP extraction: a Fourier transform is performed, critical band analysis is calculated in order to transform the frequency content into the Bark scale, there is then equal-loudness pre-emphasis applied, and then cubic-root compression is applied in order to emulate the human hearing power law. The resultant signal is then convolved with 59 2D Gabor filters that have been slightly modified for speech feature extraction. The Gabor filters in this instance consist of the convolution of a complex sinusoidal component with a Hamming window. Mathematically, each Gabor filter *g(n, k)*, where *n* is a time index and *k* is a frequency index, is a convolution of sinusoid *s(n, k)* and Hamming window *h(n, k)*, shown below:

$$s(n,k) = \exp\left(i\,\omega_n(n-n_0) + i\,\omega_k(k-k_0)\right)$$

Equation 4.1 - Complex sinusoid component of Gabor filter (Meyer et al., 2011).

$$h(n,k) = 0.5 - 0.5\,\cos\left(\frac{2\pi(n-n_0)}{W_n+1}\right)\,\cos\left(\frac{2\pi(k-k_0)}{W_k+1}\right)$$

Equation 4.2 - Hamming window component of Gabor filter (Meyer et al., 2011).

ω_n and ω_k are the time and spectral modulation frequencies of the sinusoid and W_n and W_k are the time and frequency window lengths of the Hamming window. Experiments that used this technique found that models had higher recognition rates (by about 4-6%) when using Gabor features as opposed to MFCC, PLP, and LPC (Meyer et al., 2011).

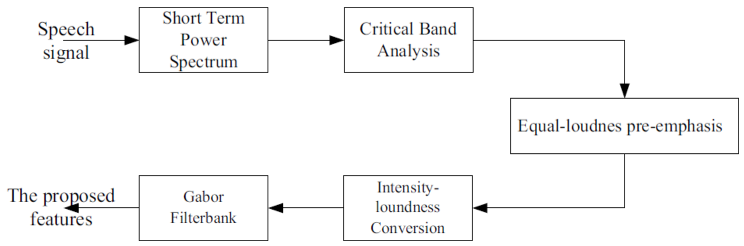

Speech signal
Short Term Power Spectrum
Critical Band Analysis
Equal-loudnes pre-emphasis
Intensity-loundness Conversion
Gabor Filterbank
The proposed features

Figure 4.2 - Block diagram of Gabor feature extraction (Meyer at al., 2011).

"Featxtra" also includes additional processing techniques like Auto-Regressive Moving Average (ARMA) filtering and Long-Term Spectral Variability (LTSV) streaming. ARMA filtering is a combination of auto-regressive modelling that aims to smooth input signals by focusing on high energy components and moving average modelling which serves to enhance signals by suppressing constant level regions using band-pass modulation filtering (Ganapathy, 2015). In the simplest terms, this increases noise robustness by removing constant signals of noise and boosting essential spectral information with the most energy such as the phonemes the ASR system aims to target. LTSV aims to correct for increased error rates for speaker recognition over longer time intervals. Long term variability is collected by measuring the variations in feature vectors over a long period for a target speaker. The averages of these variations can then be collected and used to apply spectral equalization to inputs from the target speaker, thus negating the effects of longer-term fluctuations in speech signals (Furui, 1978). This has been shown to increase accuracy ratings in ASR systems but may be a process that is overly speaker dependent.

It is important to review the tools available in this library in order to assess the variety of affectations that can be made to ASR feature extraction frontends in order to increase their efficacy and accuracy. Although these techniques will not be used in the following experiments, it is useful to know that they exist and that the findings may indicate that one or more of these processes could be employed in order to further improve ASR models.

PNCC features were computed using the "PNCC for Kaldi" library, an open-source collection of bash and Matlab scripts that aims to calculate PNCC features for output into the Kaldi ark feature format (Kwon, 2018). There are several options available to fine tune these features such as the choice between Gammatone and Mel filter banks as well as the ability to use either power nonlinearity compression or logarithmic nonlinearity compression. Settings were kept to default (40 filter banks, Gammatone filters, and power nonlinearity compression) in order to test the traditional approach to PNCC features.

4.2.2 Datasets and Pipelines.

Two different datasets were used in order to test these feature extraction methods on a variety of different speech signals. Choice of dataset often determines a specific processing pipeline that Kaldi has set up as the default method for the creation of ASR models, and these processes will be explained for each dataset in order to provide greater context into how each of the ASR models for this experiment were being created and processed. Each processing pipeline shares general conceptual similarities that follow basic ASR principles but also contains several nuances that differentiate them. Due to the fact that this dissertation focused solely on variations to feature extraction, a function that occurs as a frontend to the pipeline, the following processes were all kept as Kaldi had originally designed them in order to remove any other variables in the ASR model performance.

The first dataset that was chosen for these experiments was the Voxforge dataset (Voxforge, 2019). Voxforge is a collection of transcribed speech in a variety of languages such as English, German, French, Russian, etc. that is intended to be used with open source speech recognition libraries. The project was created to address the difficulty of obtaining high quality speech corpuses that are not locked behind restrictive licenses that require expensive license purchases or private membership. This is also done in the hopes of improving the quality of open source acoustic models, as it they are commonly not operating at the same error rates of commercial ASR models (Voxforge, 2019). If there is easier access to high quality speech samples in a variety of languages, then there is greater potential for open source ASR modelling to improve, as these models are highly dependent on the data they have access to. Voxforge primarily collects its speech audio files from the LibriVox project. LibriVox is a non-profit organization that has created an online repository populated with free audiobooks. The recordings are all created by volunteers that read books in the public domain. This means that the recording quality can potentially vary and that these speech audio samples were not originally intended to be used for the creation of ASR models, which may contribute to their behavior when training models from this data.

This feature extraction research chose to use the English subset of the Voxforge collection. However, although it is possible to subset the data to limit the English samples to only American/British/Australian English as an example, this filtering was not performed in order to retain multiple accents amongst the English speech samples. This is in an attempt to emulate a more multilingual dataset despite the fact that all the audio recordings are still in English. The accented English provides a mixture between a simpler unilingual dataset and a multilingual dataset that could be more useful for testing ASR models that may be exposed to a wider variety of speakers. The audio files are in mono WAV format and sampled at 16 kHz, 16-bit depth.

The Kaldi pipeline typically begins with data preparation; first the Voxforge dataset is organized and sorted into training and test sets. The default setting uses 20 speakers as the test set. At this point, the speakers are associated with IDs and there are transcriptions that are matched with their corresponding audio files for testing and training the ASR model. Once the data is prepped, a language model and vocabulary are created using the SRI Language Modeling toolkit (SRILM). SRILM is a free library that consists of several C++ class repositories that are intended to be used to create language models, typically used for speech recognition, statistical language tagging and segmentation, and machine translation (SRI International, 2016). Next, lexicons are prepared that act as dictionaries, containing potential words and the phonemes that they are matched with. Any words that are out of vocabulary (OOV) have their pronunciations supplied by a Sequiter G2P model that Kaldi has pretrained for use with the Voxforge dataset. Sequiter G2P is a grapheme-to-phoneme converter that is meant to fill in the phonemes for unseen words and is helpful with building English language dictionaries for language models (Bisani and Ney, 2008). Some final data preparation then occurs and the grammar FST is created. It is at this point that feature extraction is performed on the organized data; target alternative techniques are applied here in order to test these different approaches. This is of course set to extract MFCC features by default. Once features are extracted, CMVN statistics are calculated. The Kaldi process then subsets the training directory into a smaller sampling of 1000 speech samples in order to more easily train the models.

Once features are fully prepared, models can be trained. First, a monophone model is trained on the subset training data. A decoding graph is created using this monophone model in the form of an FST and then used in decoding the model, which takes place by testing the model with the test set. At this point WER's are calculated and output so that there is a metric for the performance of the first, most basic model. Alignments are then extracted from the monophonic model so that they can be used to train the first triphone model. The triphone model is trained by 2000 gaussians at a time, concluding at 10000 total gaussians. A decoding graph is then created using this model and decoded, similarly to the process for the monophonic model. This process is repeated once more in order to create a second order triphone model that utilizes delta + delta-deltas. A new triphone model is then trained from the output of the previous model using the LDA + MLLT process. After this model is decoded, denominator lattices are calculated and used to train an MMI model. This model is then decoded and another MMI model is trained with a boost of 0.05. An MPE model is trained next and the alignments from this model are used to train an LDA+MLLT+SAT model. The LDA+MLLT+SAT model is FMLLR decoded and a new set of denominator lattices is generated. The final model uses the alignments from the LDA+MLLT+SAT model and generates an MMI model. There is then a separate FMLLR and standard decoding performed in order to produce the final WER scores for the Voxforge model.

There are additional processes for experimenting with fMMI models, which serve as extensions to the MMI process, but these were deemed unnecessary and it was found that the final MMI model was substantial enough to observe the performance of these feature extraction techniques after many thorough rounds of training. Four different models were trained using this processing pipeline and their resultant WER scores were compiled for discussion in the results section of this dissertation. While this pipeline is largely traditional and does not utilize any of the more advanced contemporary techniques such as the employment of a DNN, it serves as a good representation of a simpler machine learning recipe and allows for the observation of feature extraction methods with more basic ASR models.

In order to supplement the tests performed using the Voxforge database, the Texas Instruments, Inc. (TI) and Massachusetts Institute of Technology (MIT) database, officially referred to as the TIMIT Acoustic-Phonetic Continuous Speech Corpus (or TIMIT), was also used for further ASR model creation. The dataset was specifically created for the purpose of acoustic-phonetic studies, particularly in regard to speech recognition. The 16 kHz sampled, 16-bit speech waveforms were recorded at TI, transcribed by MIT, and then prepared for production by the National Institute of Standards and Technology (NIST) (Linguistic Data Consortium, 2019). There is a total of 630 speakers in the dataset speaking the eight major dialects of American English: New England, Northern, North Midland, South Midland, Southern, New York City, Western, and Army Brat (moved around) (S Garofalo et al., 1992). The map below delineates each of these major regionalities.

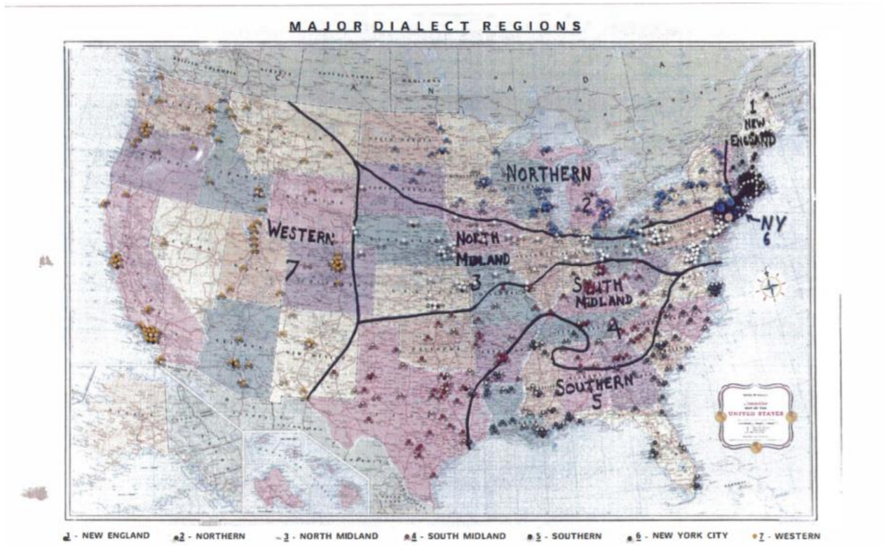


Figure 4.3 - Division of regional dialects in the U.S. (S Garofalo et al., 1992).

While this dataset presents several different accents, it can all be consolidated into the American English label and so is more representative of a unilingual dataset. TIMIT was an ideal database for this research since it has been used in many previous ASR experiments and is commonly used as a benchmark for training ASR models. The database is considered to be advantageous to work with as its training set is small enough to easily create many variations for use in experimentation. However, this also means that successful experiments with the TIMIT dataset may not indicate significant improvements for practical ASR deployments as they can often contain much larger vocabularies, varying acoustic conditions, and far more training data. Experiments that use TIMIT are thus considered as a basic starting point for potentially vetting better directions for further research (Hinton et al., 2012). The dataset is distributed by the Linguistic Data Consortium (LDC) and so is a much more legitimate database for strict ASR use as opposed to the more open source approach of the Voxforge dataset.

The Kaldi library uses a different approach to create models using the TIMIT database, providing another useful point of comparison between the use of feature extraction methods between these two databases. Since ASR model creation pipelines can utilize a variety of approaches, it is best to keep these as varied as possible in order to observe how effective the feature extractions techniques are in a wide variety of contexts. It is interesting to note that TIMIT models are decoded using both a test set and a larger development set in order to create two different scores per decoding cycle. A development set is typically used to verify the hyperparameters during model training whereas a test set is intended to be tested on the final model. This indicates there may be greater emphasis on fine tuning model training parameters when creating ASR models using the TIMIT pipeline.

The preliminary data preparation steps are very reminiscent of the Voxforge data pre-processing. The database is first organized, given speaker and utterance IDS, and then a dictionary is sorted using the possible words and phonemes that can be encountered in the speech samples. A language model is prepared using this data; notably silences appear as distinct words in the TIMIT dictionary and so are scored for accuracy similarly to sounded phonemes. After the data is prepared, feature extraction takes place and it is here again that alternative methods can be swapped in for performing the experiments of this research. CMVN stats are calculated for these features as well. A monophonic model is the first model iteration trained and a decoding graph is generated and decoded as in the previous pipeline. The first triphone model is trained with delta + delta-deltas and this is similarly graphed and decoded. An LDA+MLLT model is then trained in order to create a second triphone model and then an LDA+MLLT+SAT model is created to output a third triphone model. These are both respectively graphed and scored. At this junction, the TIMIT process begins to differentiate itself. FMLLR decoding and alignment are performed which then feed into a Universal Background Model (UBM) training phase. A GMM-UBM is created by training a UBM by using many different speakers to represent different speaker models; these models are then adapted into a GMM structure by using MAP adaptation. This type of model is ideal for speaker identification and verification tasks (Sadjadi et al., 2013). This model is then used to train a Subspace Gaussian Mixture Model (SGMM) and this model is graphed and decoded. An SGMM is akin to a GMM but all feature vector output Gaussian components are summed together and different Gaussians are distinguished based on modifications to an external subspace. This format is otherwise identical to a standard GMM. However, it is computationally more efficient as it reduces the number of parameters that need to associated with their respective speech states. It also requires far less training data to create effective models. However, it can be much more complex to create the SGMM subspace while retaining the ability to discern between each of the available speech states (Povey et al., 2011).

Denominator lattices are calculated using the SGMM model and then a combined MMI+SGMM model is created and decoded over 4 iterations. A DNN hybrid model is then trained using a tanh activation function, an initial learning rate of 0.015, a final target learning rate of 0.002, 2 hidden layers, and 20 training epochs. DNN-HMM hybrid systems are a recently popularized technique for enhancing the performance of ASR models. The primary goal of a DNN in these systems is to calculate the posterior probability estimates for HMM states. This is calculated using observation $\mathbf{o}_{ut}$ over time *t* in utterance *u* to yield a DNN output $y_{ut}(s)$ for HMM state *s* as follows:

$$y_{ut}(s) \triangleq P(s|\mathbf{o}_{ut}) = \frac{\exp\{a_{ut}(s)\}}{\sum_{s'} \exp\{a_{ut}(s')\}}$$

Equation 4.3 - Calculation of HMM state posterior probabilities using a DNN (Vesely et al., 2013).

The activation function is represented by $a_{ut}(s)$ for state *s* and is a tanh activation function in this implementation. An activation function within the context of a neural network serves to scale the neural network's output within a desired range, which in the case of tanh is -1 to 1. The DNN aims to achieve a target training function by calculating cross-entropy loss, which is the logarithmically scaled total difference between a target probability and a probability calculated by the DNN. This loss function is minimized using stochastic gradient descent (SGD), a function that seeks to find the minimum error value by adjusting weights and calculating whether or not the gradient of error is positive or negative; in the case of error minimization it continually seeks to output a negative gradient until it reaches a gradient of 0, which is assumed to be the minimum point of error. Weights are updated each training iteration according to a set learning rate and the number of training epochs indicates the number of training iterations. The resultant models are then decoded and scored, and a combination model is created that combines the DNN with the SGMM models and is then also subsequently scored.

An additional DNN is trained using Karel Vasely's training method in Kaldi, referred to as "Karel's Recipe" (Povey et al., 2011). The recipe begins with contrastive divergence training, a very efficient method of creating a restricted Boltzmann machine (RBM). An RBM consists of a layer of "visible" randomized binary units that represent binary input data. These units are connected to a hidden layer that aims to model the most significant dependencies between the "visible" units (Hinton et al., 2012). There exist no connections between any nodes in the same layer (visible to visible or hidden to hidden) but all of their potential connections are established between the visible and hidden layers. An RBM thus acts as a type of Markov Random Field (MRF), a model that acts as a representation of a series of random variables that contain random dependencies that may or may not exist between all of the potential nodes (Kindermann and Snell, 1980). However, an RBM contrasts from an MRF because it consists of two primary groupings (visible and hidden), the different units do not share any of the same weights, and the hidden variables always remain unobserved, even in the training stage (Hinton et al., 2012). RBMs were used as some of the most popular basic units of early deep learning modelling. The energy representation of an RBM, *E(***v**, **h***)*, is given below:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

Equation 4.4 - RBM energy representation (Hinton et al., 2012).

In this equation v_i and h_j are binary states of visible state *i* and hidden state *j*, their biases are a_i and b_j and w_{ij} is a representation of their weights.

Contrastive divergence works by approximating the gradient of a log-likelihood of a present observation generated in the context of a Markov chain starting at the last observed example. The algorithm aims to reduce the gradient as much as possible (typically using gradient descent) in order to increase the probability that a reconstructed observation based on a previous observation accurately represents a target observation. The equation representing the contrastive divergence method of calculating the change of weights (w_{ij}) for an RBM is given below:

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{data} - <v_i h_j>_{recon})$$

Equation 4.5 - Contrastive divergence calculation (Hinton et al., 2012).

In this context, the contrastive divergence consists of 1-step of Markov Chain Monte Carlo sampling (Povey et al., 2011). Markov Chain Monte Carlo sampling can be simply defined as a method to infer the posterior distribution of a parameter (in this case the gradient) by taking random samples within a probabilistic space (Shaver, 2017). Although successful steps of sampling are often recommended to train better models, this recipe aims to create this initial RBM as a frontend feature selector, and so more than one stage of this process is unnecessary (Hinton et al., 2012).

Hinton (2012) states that real-valued data, such as audio reformatted into feature vectors, are better modelled using linear variables with Gaussian noise; this is thus incorporated into the RBM creation process in order to yield a Gaussian-Bernoulli RBM (GRBM). The revised energy function for an GRBM is shown below, with σ_i representing the standard deviation of Gaussian noise for visible unit *i*:

$$E(\mathbf{v},\mathbf{h}) = \sum_{i\in\text{vis}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j\in\text{hid}} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij}$$

Equation 4.6 Revised energy representation of an RBM with Gaussian noise (Hinton et al., 2012).

Additional RBMs can be trained to model the significant dependencies between the first RBM's hidden units and this process can be performed successively in order to produce models with greater and greater complexity. The GRBM and these successive RBMs can be stacked in order to create a deep belief net (DBN), which represents a single, multi-layer generative model (Hinton et al., 2012). As soon as all these weights have been calculated, they can be read in reverse in order to mimic a deterministic feed-forward DNN; this embodies one of the primary advantages of the DBN format: these stacked weights can be used to infer the states of all the hidden layers in a single forward pass. A final output layer consisting of a softmax activation function is then added, concluding in the creation of a DBN-DNN. This DBN-DNN can then be interfaced with an HMM as is increasingly the case with ASR systems.

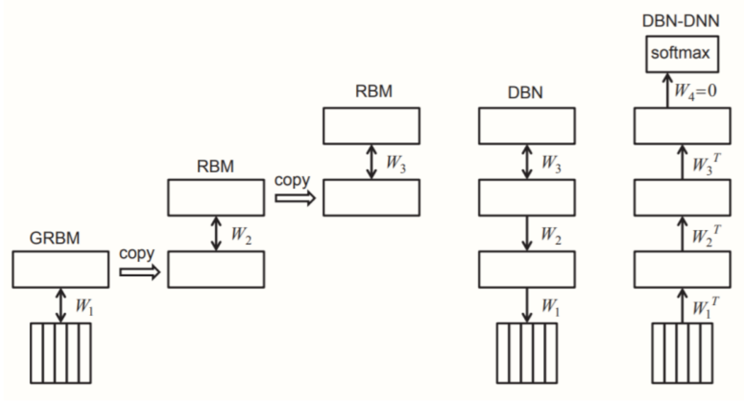


Figure 4.4 - Block diagram of the creation of a DBN-DNN meant for the analysis of a series of windowed acoustic frames (Hinton et al., 2012).

Kaldi has chosen to implement this specific approach to ASR modelling because it has been shown in several research experiments that the DBN-DNN configuration has performed very successfully when paired with the TIMIT dataset (Hinton et al., 2012; Mohamed et al., 2009; Saineth et al., 2009). DBN-DNN models were recorded to have improvements of 2%-7% over other more traditional modelling techniques such as GMM-HMMs. It is notable that DBN-DNNs do not actually require data to be uncorrelated; this is in opposition to conventional ASR approaches that prefer the more independent data vectors provided by techniques such as MFCCs over simpler filter bank coefficients. However, it was found that the best DBN-DNNs that were trained with filter bank coefficients outperformed the same models trained with MFCCs by 1.7% (Mohamed et al., 2009). This indicates that this modelling approach could work best with LPCs rather than DCT-transformed MFCCs; it is also totally possible that the combination of these techniques, PLP, may be one of the better options for input into a DBN-DNN.

At the conclusion of these progressively more complex model approaches, the WERs are collectively output for each processing step in order to allow for comprehensive review of the scoring progression.

Although it is ideal to work with completely clean speech databases and ASR systems can only hope to learn on such straightforward data, it is necessary to test these different ASR systems and feature extraction approaches by training and testing models using noise-corrupted databases. In order to conduct these experiments, the TIMIT database was corrupted with various types of “realistic” noise from a variety of different environments. It is important to establish the robustness of the chosen feature extraction techniques by examining their ability to cope with the presence of background noise and to adapt to the distortion introduced in the frequency spectrum of input speech (Hirsch and Pierce, 2000). The AURORA Experimental Framework, a library in C, was used in order to introduce additive noise and filtering. The stated aim of the library is to effectively evaluate alternative forms of frontend feature extraction, which aligns well with the goals of this research (Hirsch and Pierce, 2000). The filtering component of the database transformation is by default defined by the G.712 frequency characteristic provided by the International Telecommunication Union (ITU). This filtering is meant to emulate the characteristics of pulse code modulated (PCM) channels consisting of 4 wires as they transmit signals consisting of vocal frequencies (ITU, 1996). This frequency response of this filter is pictured below:

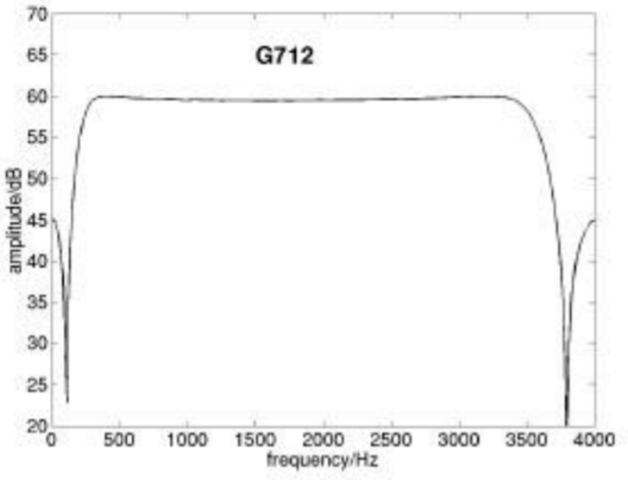


Figure 4.5 - ITU Recommendation G.712 Frequency Response (Hirsch and Pierce, 2000).

The filter is akin to a bandpass with a considerably flat response in the passband from around 300 Hz to 3500 Hz. This is the least destructive filtering available in the AURORA library and serves to represent any minor telecommunications distortion present when employing ASR devices.

The Aurora library then adds noise to the speech samples at target signal-to-noise ratios (SNR). This is defined as the ratio of signal to noise energy after the G.712 filtering is applied. Speech energy is calculated using ITU recommendation P.56, which serves as a method for objectively measuring speech levels in a signal (Hirsch and Pierce, 2000). The noise energy is calculated using the same technique and is then scaled using a root mean square (RMS) calculation. A segment of the noise that is equal in length to the speech sample is chosen for this RMS level measurement. The library is meant to work with a selection of noise examples that represent the most probable telecommunication settings for ASR: the settings are suburban train, crowd of people (babble), car, exhibition hall, restaurant, street, airport, and train station. A smaller selection consisting of babble, car, suburban train, and street noise were chosen for this experiment in order to provide a practical benchmark for observing ASR models in these noise contexts while reducing the total number of overall models that had to be manufactured for experimentation. These also contain a good mixture of stationary noise, as in the case of car noise, and non-stationary noise, such as in the case of street noise (Hirsch and Pierce, 2000). The long-term spectra of each of the chosen noise samples is displayed below over a dynamic range of 40 dB; note that much of their low frequency energy appears to be fairly similar, although there are some distinctive differences between the noise floors and they do contain audibly different noise patterns:

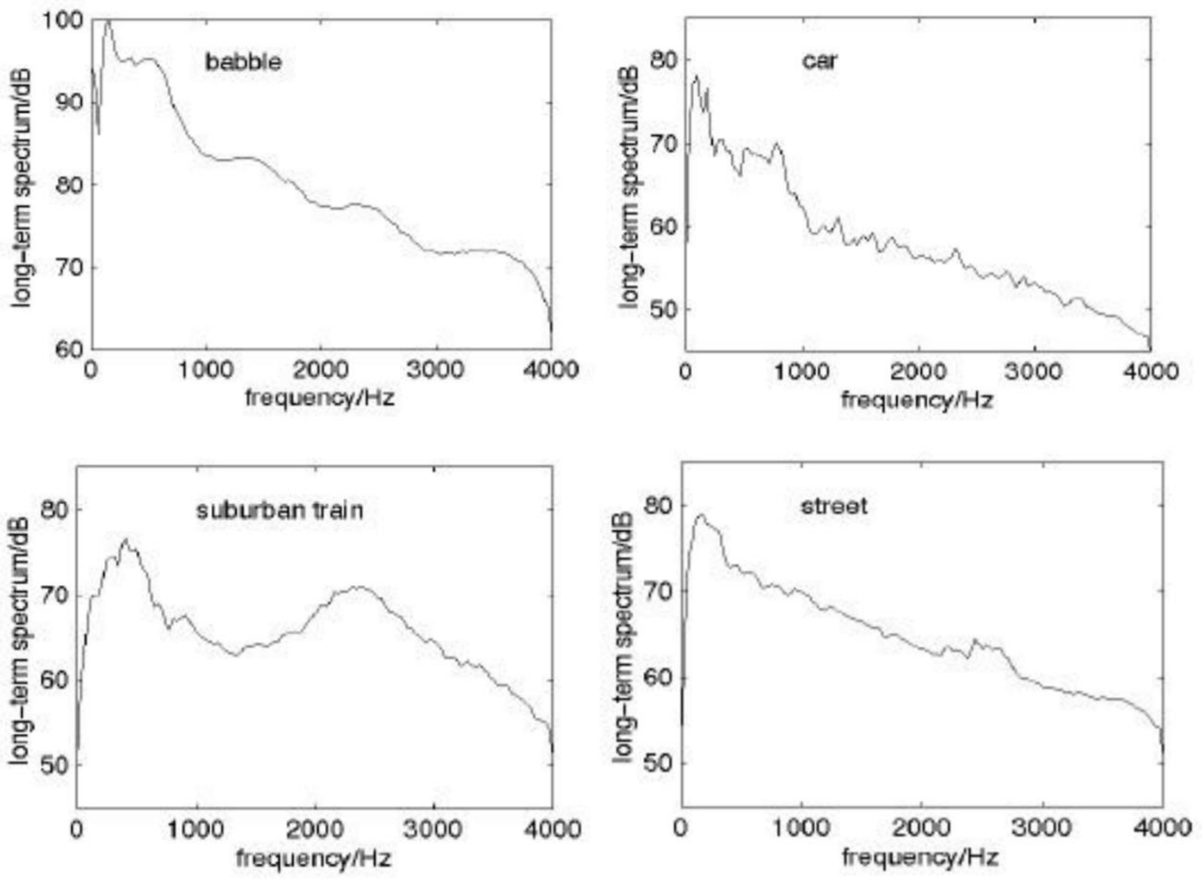


Figure 4.6 - Long-term spectra of the 4 noise samples chosen for noise robustness experimentation (Hirsch and Pierce, 2000).

It can be noted that each of these sound signatures is somewhat disparate; babble noise has the most significant low end and is fairly smooth in response, car noise is quite uneven with a mid-range peak (likely representing engine noise), train noise is bimodal with a peak in the mid-range and in the higher-mid frequencies, and street noise has a low end peak that steadily declines with a slight bump from around 2500 Hz to 2800 Hz. This very different profiles are chosen in the hopes of revealing different qualities of the feature extraction technique performance.

It is typically more prudent to employ ASR models that are trained with both clean and noisy speech data (often referred to as multi-conditional) in order to exploit both the lower error rates and more accurate representation of speech lexicons found in training with clean data as well as the potential noise robustness and familiarity introduced by using noisy data in training sets (Hirsch and Pierce, 2000; Madisetti, 2010). However, it is also acceptable to test feature extraction methods using models trained and tested with the same noise floors (ranging from clean to various other SNRs) in order to place greater emphasis on observations of various feature extraction methods rather than testing the effects of different noise floors (Al-Noori et al., 2016; Sárosi et al., 2011). Training and test data that have matched noise floors typically have lower error rates than mismatched sets for the obvious reason that ASR models can better recognize conditions they have already been exposed to. For the purposes of this experiment, it was determined that it would be better to observe single condition training and test sets in order to assess the effects of various noise on feature extraction methods. Experiments were conducted using the four different noise floors at four different SNRs (0, 5, 10, and 20 dB) for the four feature extraction methods, creating a total of 64 different models in order to score the feature extraction methods in a variety of different contexts. These would also serve as a useful point of comparison to the originally scored clean TIMIT models.

4.3 Experimental Results

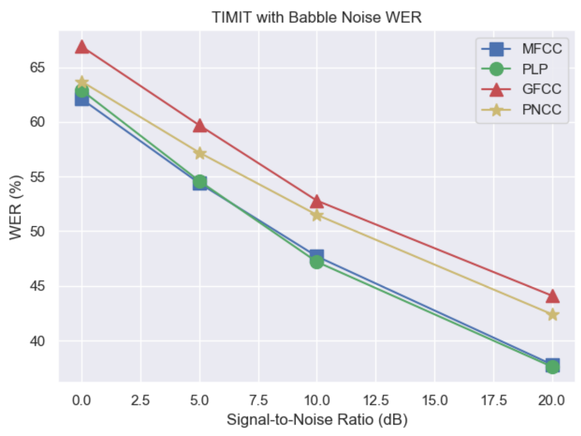
4.3.1 Summary of Results.

The following table includes the WERs for the two datasets, Voxforge and TIMIT. These tests were all conducted using the clean versions of these datasets. Note that the Voxforge error rates were calculated at a higher resolution than the TIMIT WER scores, as they extend to the hundredths of a decimal place. Error rates were recorded for each iteration of the ASR model pipeline, but it is informative to see the error rates of the initial monophone models as compared with final error rates in order to get a sense of the trajectory of progress for each model. Each model is scored over a series of iterations and as the highest scoring iteration varied for each technique the best error rates were chosen for simplicity. The highest scores in each of their respective categories are highlighted in bold.

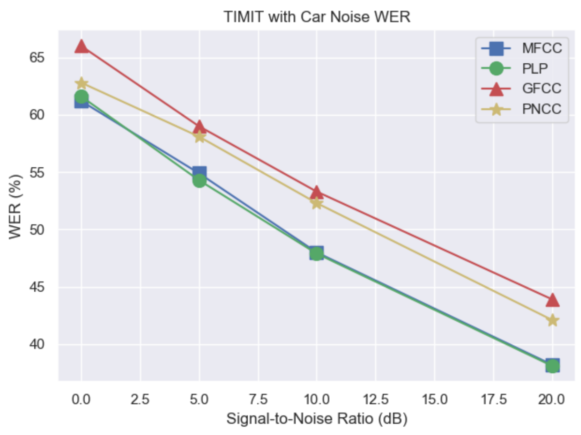
Feature Extraction Technique	**Voxforge WER**		**TIMIT WER**	
	Monophone	Final	Monophone	Final
MFCC	46.96%	14.00%	**32.2%**	**18.2%**
PLP	**35.52%**	11.16%	32.3%	18.5%
GFCC	77.33%	11.60%	48.0%	21.2%
PNCC	37.12%	**10.55%**	36.0%	21.7%

Table 4.1 - Compilation of error rates for clean Voxforge and TIMIT database testing.

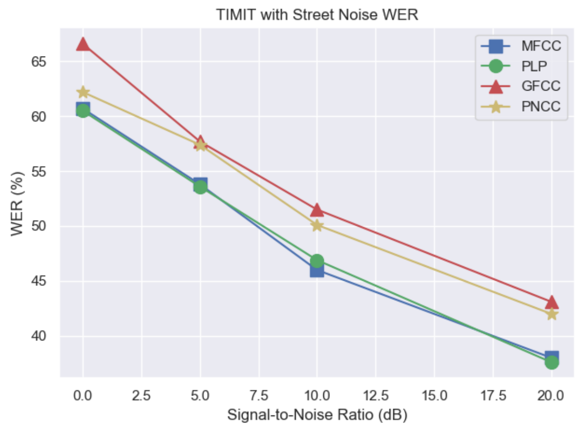
The test results for noisy datasets were graphed in order to provide insight into the trends of each feature extraction technique as SNR varied with dataset. The graphs also provide an overall impression of the competitive performance of the different techniques. Only the final error rates were used to allow for straightforward analysis of the final model performance. Once again, the best error rates of the decoding iterations were chosen for each technique.



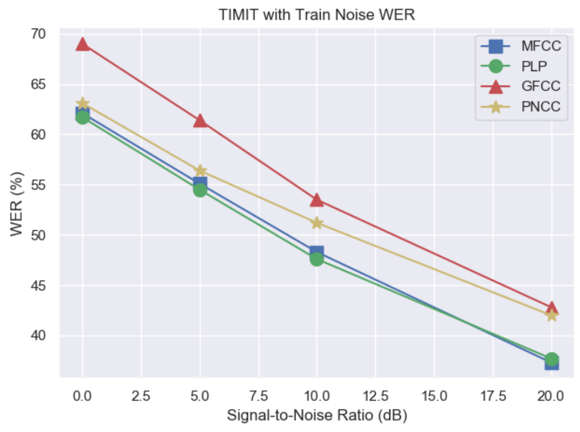
(a)



(b)



(c)



(d)

Figure 4.6 – (a) WER for TIMIT trained with babble noise. (b) WER for TIMIT trained with car noise. (c) WER for TIMIT trained with street noise. (d) WER for TIMIT trained with suburban train noise.

4.3.2 Analysis of Results.

The results for this research were unexpected, at least according to the prior research that had been gathered. It was informative to have used two different datasets, as both datasets revealed different elements of their respective feature extraction experiments. The Voxforge tests behaved somewhat as intended as techniques like PLP and PNCC outdid MFCC in error ratings, but not by much. PLP error rates were the best for the monophonic models, indicating that PLP feature extraction does not require as much intensive training to already be effective. PNCC was close in score though, and this technique resulted in the strongest error rates for the final MMI model which incorporated LDA, MLLT, and SAT enhancement techniques as well as FMLLR decoding. PLP was a close second but it appears PNCC gained the most from using all of the advanced modelling techniques available. The most unexpected results were those from the GFCC technique, which began at by far the worst error rate of 77.33% when decoding the monophone model. Several libraries were experimented with when implementing the GFCC technique in order to verify that this was not an algorithmic problem but it was found that the featxtra library did indeed perform the best when allowed to produce more advanced GFCC models.

Upon further investigation, it was found that the GFCC trained model began to definitively improve once the third triphone model had been trained, leaping from an error rate of 56.27% to 21.90% in one model iteration. Notably, this stage is where the LDA+MLLT optimization is introduced. This may indicate that the GFCC technique gains the most by increasing the decorrelation between feature vectors. Perhaps GFCC does not produce vectors that are decorrelated enough to provide effective means of differentiation for the recognizer. This contrasts with MFCCs which only show an improvement from 24.97% to 23.79% in the same stage, which is a negligible difference yielded from the LDA+MLLT process. It could be that feature vectors in other techniques are already decorrelated enough to be effective and do not stand to gain as much from a processing technique that makes features more distinctive. After this point GFCC models proceeded at a linear pace upon each decoding iteration. Their error rates ended up outcompeting with MFCCs in the final MMI model, but they still were unable to top PLP and PNCC techniques. This technique showed the greatest variance between the first and last models in the Voxforge pipeline and indicate that GFCCs may be more reliant on advanced modelling techniques than other feature extraction methods, implying that they may not be effective enough when used with simplified model architectures.

The results for the TIMIT processing pipeline were also surprising, although they had overall lower variance than the error rates computed for the Voxforge models. MFCCs were the clear winner out of the four techniques as it had the best error rates in both the monophonic and advanced DNN combination models, consistently outperforming the other techniques. However, it must be noted that PLP was not too far behind and can be considered to be nearly identical to MFCCs in performance. Disappointingly, alternative methods GFCC and PNCC had very poor ratings for this task and are also grouped closely together in terms of score. Again, GFCC has performed the worst overall and did not show nearly as much promise as earlier research may have suggested.

The different performance characteristics of these two datasets is quite apparent and definitely warrants discussion. Why did alternative techniques show so much promise in the Voxforge dataset, as in the prior research, but then fail to significantly improve upon the MFCC technique in the TIMIT tests? It could perhaps be attributed to the difference in the databases themselves, as TIMIT is a more legitimate database that is designed to be phonetically rich specifically for the purposes of create robust ASR models. This is in contrast to Voxforge which is a looser, open-source project that simply relies on audiobook recordings as well as other free recordings that speakers have lent to the database for use. This could imply that when these techniques are exposed to the proper variety of phonemes they fall short of the MFCC paradigm, but they may perform better with a dataset with less focus. The Voxforge dataset also contained a variety of different accents in the English language, which for the purposes of this research were not filtered out in any way. The TIMIT dataset is configured to use only a handful of American accents, greatly decreasing the potential speaker variations. MFCCs could perform best in this context with less variation, and the increase in speaker accents could have benefitted other feature extraction techniques which may have proven to be more robust in accepting a greater slew of speaker inputs.

The performance disparities could also be related to differences in the respective database processing pipelines. The Voxforge models are only trained with a random subset of the full database for computational simplicity; in the case of these experiments a subset of 1000 speech samples were used for training and testing models. However, the TIMIT pipeline uses the entire database of 630 speakers, which could contribute a much richer training and test database to work from. There is a total of 6300 different utterances in TIMIT, representing 10 sentences for each of 630 speakers. This can be correlated to vocabulary size as the most speech samples there are, the more potential words and phonemes may comprise this selection. Along that line of logic, MFCCs and PLP features may be better suited to larger vocabulary tasks overall, while GFCCs and PNCCs could struggle beyond a certain boundary of vocabulary complexity. Voxforge models were also much simpler and so they incorporated far less advanced modelling techniques that may serve to rectify issues that could differentiate feature extraction performances. The more robust modelling of TIMIT may have nullified the benefits gained from employing feature extraction techniques that were shown to have provided advantages in previous work.

The hypothesis that PLP could perform well in the context of a DBN-DNN system was however proven to be correct.

The performance of these feature extraction techniques on the datasets convoluted with noise was also unexpected. Prior research indicated that this was perhaps where alternative methods could display their best robustness versus the traditional MFCC technique. However, the performances between the clean and noisy TIMIT datasets remained linear; the same techniques (MFCC, PLP) outperformed alternative techniques GFCC and PNCC just as they had done with the clean datasets. This indicates that the noise may not have had a great impact on the performance of the techniques other than linearly increasing the error rate due to the increase in noise. GFCCs were previously able to outcompete PNCCs in the final clean model scoring, but in the noise testing these models performed the worst overall with little contest. GFCCs may not be as noise robust in these contexts as the other methods as they were detrimentally affected the most out of any technique. While PNCC did not do as well overall, it had some nonlinear behavior which at times was able to yield error rates that were closer to the scores seen by MFCC and PLP. Specifically, PNCC performed best when trained and tested on data corrupted by train noise, as well as by babble noise. Train noise was largely bimodal, featuring peaks in the low-mid frequencies as well as from 2 kHz to 3 kHz. PNCCs may have been able to better mitigate the peak in the presence range than GFCCs due to the difference attributed to the use of cubic root normalization, which may have better rectified this frequency discrepancy than the logarithmic normalization that was used to process GFCCs. This effect can also be related to PNCCs better ability to manage the incredibly high peaks of babble low frequency noise which at some points reached nearly 100 dB. Earlier models in the pipeline that used PNCC had even slightly outperformed MFCC models, but upon the third triphone model iteration, where LDA+MLLT+SAT normalizations are employed, MFCC models start to perform ahead of PNCCs. While GFCCs may have been able to slightly outscore PNCCs on clean models, PNCCs were able to display greater noise robustness than GFCCs as their error rates were not as detrimentally affected overall.

MFCCs and PLP features were incredibly close in performance when tested with noisy speech samples and it is difficult to clarify which is truly better. The Mel scale and critical-band scale may be stronger when skewing the frequency bandwidth against noise and may be less sensitive to these irregularities, which would vindicate its common use in popular ASR systems. There are many more ways that noise experiments could have been conducted. More noise floors could have been used, test sets could have contained a mixture of clean and noisy samples, and it would have been of interest to corrupt the Voxforge database with noise in order to examine the differences in performance between its clean and noisy speech signals. It could have even been an interesting exercise to test systems with slightly level distorted audio in order to examine their ability to cope with additional harmonic content.

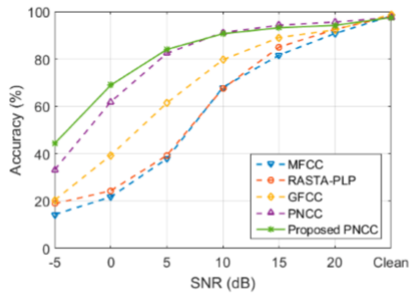
PLP could have also been implemented with RASTA filtering for the purpose of observing how much of a performance boost that would provide models trained with that technique. It is likely that RASTA-PLP would then pull ahead of all other techniques by a clear margin, but that would have to be confirmed with additional experimentation. While it must be said that this does not represent the most comprehensive noise testing suite, overall observations can be made regarding the inability of methods like PNCC and GFCC to outdo MFCC and PLP when exposed to noise as these models were greatly diminished in accuracy when handling noisier databases.

However, what accounts for the differences with previous research when drawing these observations? It could be the variables related to using a different library and thus a different set of tools when creating and experimenting with ASR models. Some experiments with GFCCs have used the older HTK library (Qi et al., 2013). This library employs a simpler implementation of an ASR model, using the traditional HMM-GMM architecture with triphone modelling and can only incorporate the MLLR process and MAP adaptation. Another GFCC exploration opted to use the Microsoft Research (MSR) Identity Toolkit in order to create GMM-UBMs using standard EM to acquire Gaussian parameters and ML decision-making techniques (Al-Noori et al., 2016). Yet another ASR research experiment opted to use models created using CMU-Sphinx, an ASR library that can optionally employ LDA+MLLT as well as MMI model creation; however, these options are sparser than in the Kaldi implementation and the experiment did not cite that these processes were even incorporated into model creation (Kim and Stern, 2016).

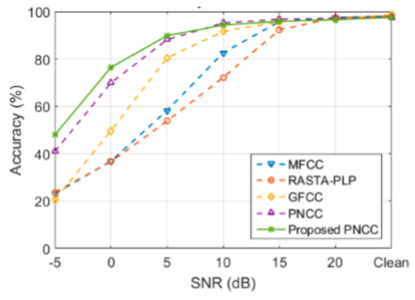
Overall, this accounts for a great deal of variability between different approaches towards creating ASR models between different open-source libraries. On top of this, none of the other processes employed equal the complexity of the model enhancements available in Kaldi. This makes sense as Kaldi is meant to be the most recent and comprehensive collection of ASR modelling techniques, and so it has the advantage of using a great amount of recent ASR research. This is the best explanation for the lack of correlation between the findings of this feature extraction research and the findings of experiments that have been conducted in the past. There have been very few experiments that have examined the differences the error rate performances of these various libraries, but testing performed by the creators of Kaldi did find that the most advanced default Kaldi models were able to achieve lower error rates than the most advanced HTK offerings by up to 2.2% (Povey et al., 2011). The findings of this dissertation have also proven that the introduction of specific modelling techniques, such as employing LDA+MLLT criteria, has had a great effect on model accuracy in specific contexts. The Kaldi toolkit contains a collection of the most relevant and impactful processing tools for creating effective ASR models, and these may have rendered the differences observed between feature extraction techniques in prior experiments obsolete.

Another recent study has tested a similar batch of feature extraction methods: namely MFCCs, RASTA-PLP, GFCCs, as well as PNCCs and an "enhanced" variant of PNCCs (Tamazin et al., 2019). The research employed the TIDIGITS database, a collection of speech samples from 207 speakers that each pronounce 11 digits twice. CMU-Sphinx was used to train GMM-HMM models using only delta and delta-delta feature methods as well as CMN. The dataset was also corrupted using noise samples from the AURORA noise database; all available noise types plus white noise were employed at SNRs of -5 dB to 20 dB in 5 dB increments. Models were trained with clean speech data and subsequently tested with clean and noisy samples.

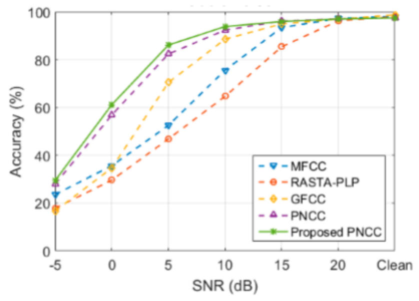
The results of Tamazin et al.'s experiment (2019) implied the inverse of the results of this dissertation. MFCC and RASTA-PLP methods performed the worst, with GFCC in the middle and PNCC having the second-best accuracy scoring. Of course, the enhanced PNCC method proposed by researchers performed the best overall, doing particularly well in street, car, and subway noise in various SNR scenarios.



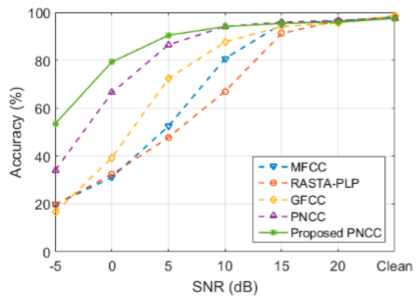
(a) White noise



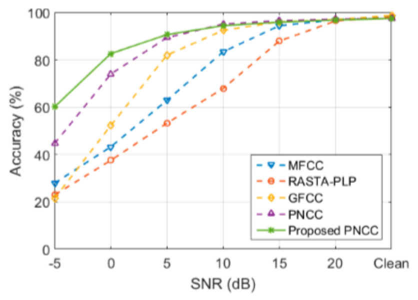
(b) Airport noise



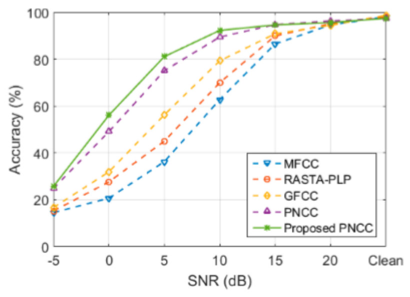
(c) Babble noise



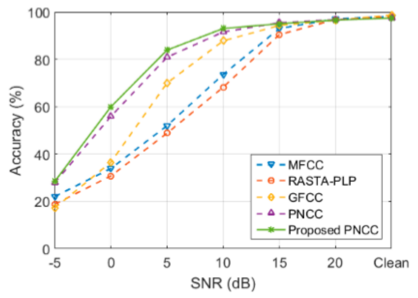
(d) Car noise



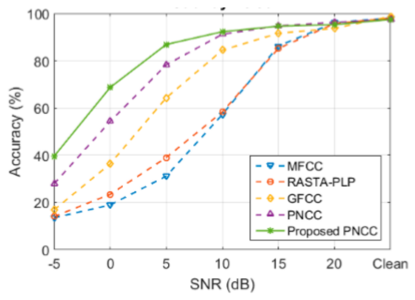
(e) Street noise



(f) Exhibition noise



(g) Restaurant noise



(h) Subway noise

Figure 4.7 – Summary of results from Tamazin et al.'s experiment in a variety of different noise environments and SNRs (2019).

While these results may appear to be the mirror opposite of the tests conducted for this dissertation, a closer examination of the experimental methodology makes sense of the difference in results. First, the TIDIGITS database is a much simpler set of speech samples as it only consists of digit utterances and less than a third of the total number of speakers in the standard TIMIT dataset. This greatly decreases the complexity of speech data and speakers that models must account for, so this may have eased the recognition task. Mass-production scale ASR systems must also be trained using speech data more akin to the Voxforge and TIMIT datasets which feature a variety of sentences and, in the case of the latter, a collection of speech samples that are meant to be phonetically rich and highly relevant for English speech training. The models that were trained for this experiment were also somewhat simplistic as they do not employ any of the methods that have been found to be highly effective above such as employing LDA+MLLT normalization, advanced MMI models, or models with DNN frontends. It was found that the introduction of some of these computational techniques greatly altered the accuracy of ASR models employing different feature extraction techniques. Their absence from a study that is incredibly recent is peculiar as these modelling techniques have become more and more common. Of course, it is also suspicious that all of this leads to results that vindicate the research's proposed method of feature extraction, despite the lack of explanation as to why such simple databases and modelling pipelines were used. It would have been more informative to use a speech database that contains full sentence utterances as well as to test these feature extraction methods with models that used more contemporary approaches to ASR enhancement.

There exist a variety of different findings when handling alternative feature extraction methods due to a lack of centralization in choice of database and ASR modelling library when proceeding with experimentation. These decisions can make a big difference which can be observed when comparing the greatly varying results of this compilation of research. It would be most informative to create a standard for ASR feature extraction comparison, or at least to qualify future findings by explaining or compensating for a particular choice in model pipeline or speech database. It is difficult to qualify the results of this dissertation within the current context of ASR feature experimentation, but it will have to be sufficient to simply be aware of the major variables that can influence experimental outcomes and how they may reflect the results of their experiments. It could be most prudent in the future to mirror a very specific research study when performing ASR testing in order to produce comparable results without great variance in the choice of materials; specifically to the duplicate the exact models, databases, and testing methodologies employed in order to verify prior results.

5 Conclusion

ASR systems have a plethora of configuration options and algorithms for enhancement available. These different processes seek to bring speech recognition accuracy closer and closer to human hearing capabilities. There are many possibilities for altering the methods used for feature extraction, which represent an AI's ability to hear as it is most closely mirrored to how humans process sound. The results of this experiment appear to verify that MFCCs, and the second most popularly used technique PLP, do feature the best performance scoring as either MFCC or PLP had some of the highest scores in phonetically rich or speaker diverse datasets respectively. The addition of RASTA filtering to the PLP technique used in this experiment would have likely increased PLP model accuracy beyond the other models' scores, indicating that this may be the strongest feature extraction method moving forward. However, PNCC had a few moments where it had performed particularly well, specifically in the instance of the clean, diverse Voxforge dataset and in some of the lower SNR noise tests. PNCC has been shown to capably incorporate a variety of different processing techniques within its standard pipeline (Kim and Stern, 2016; Tamazin et al., 2019). It might be advisable to encourage further experimentation into modifications that can be made to PLP and PNCC feature techniques as these explorations could lead to more and more robust extraction methods. The fact that these two approaches are capable of incredibly modular pipelines likely contributes to their current success and continued use in ASR modelling; this should be a quality any feature extraction technique should strive to be able to incorporate. Unfortunately, the findings of this research do not promote much promise in the GFCC technique. While there were some performance improvements to be observed over MFCC in the specific instance of the clean Voxforge dataset, its performance did not overwhelmingly outdo either PLP or PNCC in any contest. GFCCs are also not commonly tested with any additional processing, leading to its characterization as an inflexible feature extraction method. This may not be a technique that can be readily recommended for future experiments but may serve as a benchmark for improvement over MFCCs in certain contexts.

Ultimately the most influential components at the disposal of ASR systems are the variety of modelling techniques and model adaptation, normalization, and enhancement processes. Now standard procedures such as LDA+MLLT lead to strongly decorrelated feature vectors for stronger recognition accuracy, MMI models allow for effective probabilistic improvements to recognizer decision making, and the introduction of DNNs as a frontend to HMMs is forming the backbone for much more robust ASR models. It was found that the differences in these modelling approaches yielded the greatest differences in ASR performance rates, rather than the differences in feature extraction techniques. It is recommended that future ASR research focus on experimentation with new modelling methods as the primary vehicle for improvements in ASR. While feature extraction alternative exist and can be employed, it may be best to make an informed decision on which feature extraction approach may suit the model context and to proceed forward from there. Feature extraction experimentation is currently too rife with variability in modelling methodology as there are currently many potential modelling options. However, this topic may be worth revisiting in the future when ASR models could be more standardized; this would provide a better foundation to examine the possibilities for the acoustic enhancement of ASR models. Until then it is difficult to find a technique that prevails in all contexts; for now, it is far better to investigate how an ASR model thinks through its modelling pipeline than how it hears through its feature extraction.

6 References

1. Al-Noori, A., Li, F. F. and Duncan, P. J. (2016). Robustness of Speaker Recognition from Noisy Speech Samples and Mismatched Languages. In: *Audio Engineering Society Convention 140*. May 2016. [Online]. Available at: http://www.aes.org/e-lib/browse.cfm?elib=18275.

2. Biadsy, F. et al. (2019). Parrotron: An End-to-End Speech-to-Speech Conversion Model and its Applications to Hearing-Impaired Speech and Speech Separation. *arXiv:1904.04169 [cs, eess]*. [Online]. Available at: http://arxiv.org/abs/1904.04169 [Accessed 28 June 2019].

3. Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50 (5), pp.434–451. [Online]. Available at: doi:10.1016/j.specom.2008.01.002 [Accessed 12 August 2019].

4. Chan, W. et al. (2015). Listen, Attend and Spell. *arXiv:1508.01211 [cs, stat]*. [Online]. Available at: http://arxiv.org/abs/1508.01211 [Accessed 28 June 2019].

5. Chavan, R. S. and Sable, D. G. S. (2013). An Overview of Speech Recognition Using HMM. *International Journal of Computer Science and Mobile Computing IJCSMC*, 2 (6), pp.233-238.

6. Dave, N. (2013). Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition. *International Journal for Advance Research in Engineering and Technology*, 1 (6), pp.1-5.

7. edX (2019). Speech Recognition Systems [MOOC] Available at: https://courses.edx.org/courses/course-v1:Microsoft+DEV287x+2T2019/course/ [Accessed: 1 November 2018].

8. Fogel, I. and Sagi, D. (1989). Gabor filters as texture discriminator. *Biological Cybernetics*, 61 (2), pp.103–113. [Online]. Available at: doi:10.1007/BF00204594 [Accessed 9 August 2019].

9. Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34 (1), pp.52–59. [Online]. Available at: doi:10.1109/TASSP.1986.1164788 [Accessed 5 August 2019].

10. Gales, M. and Young, S. (2007). The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends® in Signal Processing*, 1 (3), pp.195–304. [Online]. Available at: doi:10.1561/2000000004 [Accessed 16 June 2019].

11. Ganapathy, S. (2015). Robust speech processing using ARMA spectrogram models. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. April 2015. pp.5029–5033. [Online]. Available at: doi:10.1109/ICASSP.2015.7178928 [Accessed 9 August 2019].

12. Gerazov, B. and Ivanovski, Z. (2012). Overview of Feature Selection for Automatic Speech Recognition. In: *AES Convention Paper 8634*. April 2012. Budapest, Hungary: JAES. p.9.

13. Gong, S. et al. (2015). Emotion Analysis of Telephone Complaints from Customer Based on Affective Computing. *Computational Intelligence and Neuroscience*, 2015, pp.1–9. [Online]. Available at: doi:10.1155/2015/506905 [Accessed 6 August 2019].

14. Gorman, K. (2019). *OpenFst Library*. [Online]. Available at: http://www.openfst.org/twiki/bin/view/FST/WebHome [Accessed 29 August 2019].

15. Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, 87 (4), pp.1738–1752. [Online]. Available at: doi:10.1121/1.399423 [Accessed 8 August 2019].

16. Hermansky, H. and Morgan, N. (1994). RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing*, 2 (4), pp.578–589. [Online]. Available at: doi:10.1109/89.326616 [Accessed 23 August 2019].

17. Hirsch, H. G. and Pearce, D. (2000). The AURORA Experimental Framework for The Performance Evaluation of Speech Recognition Systems Under Noisy Conditions. In: *ISCA ITRW ASR2000*. September 2000. Paris, France.

18. Hinton, G. et al. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29 (6), pp.82–97. [Online]. Available at: doi:10.1109/MSP.2012.2205597 [Accessed 14 August 2019].

19. ITU recommendation G.712, "Transmission performance characteristics of pulse code modulation channels", Nov. 1996

20. Jansi, R., Amutha, R. and Radha, S. (2019). Chapter 8 - Remote Monitoring of Children With Chronic Illness Using Wearable Vest. In: D. Jude, H. and Balas, V. E. (Eds). *Telemedicine Technologies*. Academic Press. pp.121–137. [Online]. Available at: doi:10.1016/B978-0-12-816948-3.00008-8 [Accessed 6 August 2019].

21. Kim, C. and Stern, R. M. (2016). Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24 (7), pp.1315–1329. [Online]. Available at: doi:10.1109/TASLP.2016.2545928 [Accessed 14 October 2018].

22. Kindermann, R. and Snell, J. L. (1980). *Markov random fields and their applications: Ross Kindermann, J. Laurie Snell*, (Contemporary mathematics v. 1). Providence, R.I: American Mathematical Society.

23. Kwon, H. (2018) *pncc_for_kaldi*. [Online] Available at: https://github.com/gogyzzz/pncc_for_kaldi [Accessed 20 June 2019].

24. Linguistic Data Consortium (2019) *TIMIT Acoustic-Phonetic Speech Corpus*. [Online] Available at: https://catalog.ldc.upenn.edu/LDC93S1 [Accessed 12 August 2019].

25. Macherey, W. et al. (2005). Investigations on error minimizing training criteria for discriminative training in automatic speech recognition. In: *The 6th Annual Conference of the International Speech Communication Association INTERSPEECH 2005*. September 2005. Lisbon, Portugal: INTERSPEECH. pp.2133-2137.

26. Madisetti, V. (Ed). (2010). *Video, speech, and audio signal processing and associated standards*, The electrical engineering handbook series. 2nd ed. Boca Raton, FL: CRC Press.

27. Maka, T. (2015). Change Point Determination in Audio Data Using Auditory Features. *International Journal of Electronics and Telecommunications*, 61. [Online]. Available at: doi:10.1515/eletel-2015-0024 [Accessed 4 September 2019].

28. Markowitz, J. A. (1996). *Using speech recognition*. Upper Saddle River, N.J: Prentice Hall PTR.

29. Meyer, B. et al. (2011). Comparing Different Flavors of Spectro-Temporal Features for ASR. In: *The 12th Annual Conference of the International Speech Communication Association INTERSPEECH 2011*. August 2011. Florence, Italy: INTERSPEECH. pp.1269-1272.

30. Missaoui, I. and Lachiri, Z. (2014). Gabor Filterbank Features for Robust Speech Recognition. In: Elmoataz, A. et al. (Eds). *Image and Signal Processing*. Lecture Notes in Computer Science. 2014. Springer International Publishing. pp.665–671.

31. Mohamed, A., Dahl, G. and Hinton, G. (2009). Deep belief networks for phone recognition. In: *in Proceedings of the NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

32. Mohri, M., Pereira, F. and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16 (1), pp.69–88. [Online]. Available at: doi:10.1006/csla.2001.0184 [Accessed 29 August 2019].

33. Narasimha, M. and Peterson, A. (1978). On the Computation of the Discrete Cosine Transform. *IEEE Transactions on Communications*, 26 (6), pp.934–936. [Online]. Available at: doi:10.1109/TCOM.1978.1094144 [Accessed 6 August 2019].

34. O'Shaughnessy, D. (1987). *Speech communication: human and machine*, Addison-Wesley series in electrical engineering. Reading, Mass: Addison-Wesley Pub. Co.

35. Panayotov, V (2012). *Decoding graph construction in Kaldi: A visual walkthrough*. [Online]. Available at: http://vpanayotov.blogspot.com/2012/06/kaldi-decoding-graph-construction.html [Accessed 29 August 2019].

36. Povey, D. et al. (2008). Boosted MMI for model and feature-space discriminative training. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. March 2008. Las Vegas, NV, USA: IEEE. pp.4057–4060. [Online]. Available at: doi:10.1109/ICASSP.2008.4518545 [Accessed 22 August 2019].

37. Povey, D. et al. (2011). The Kaldi Speech Recognition Toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. December 2011. IEEE Signal Processing Society.

38. Povey, D. et al. (2011). The subspace Gaussian mixture model—A structured model for speech recognition. *Computer Speech & Language*, 25 (2), pp.404–439. [Online]. Available at: doi:10.1016/j.csl.2010.06.003 [Accessed 22 August 2019].

39. Qi, J. et al. (2013). Auditory features based on Gammatone filters for robust speech recognition. In: *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. May 2013. pp.305–308. [Online]. Available at: doi:10.1109/ISCAS.2013.6571843 [Accessed 2 July 2019].

40. S Garofolo, J. et al. (1992). TIMIT Acoustic-phonetic Continuous Speech Corpus. *Linguistic Data Consortium*.

41. Sadjadi, S. O., Slaney, M. and Heck, L. P. (2013). MSR Identity Toolbox v1.0: A MATLAB Toolbox for Speaker Recognition Research. In: *IEEE SLTC Newsletter*, November 2013.

42. Sainath, T. N., Ramabhadran, B. and Picheny, M. (2009). An exploration of large vocabulary tools for small vocabulary phonetic recognition. In: *2009 IEEE Workshop on Automatic Speech Recognition Understanding*. November 2009. pp.359–364. [Online]. Available at: doi:10.1109/ASRU.2009.5373263 [Accessed 14 August 2019].

43. Sainath, T. N. et al. (2015). Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. April 2015. South Brisbane, Queensland, Australia: IEEE. pp.4580–4584. [Online]. Available at: doi:10.1109/ICASSP.2015.7178838 [Accessed 28 August 2019].

44. Sárosi, G. et al. (2011). Comparison of feature extraction methods for speech recognition in noise-free and in traffic noise environment. In: *2011 6th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. May 2011. pp.1–8. [Online]. Available at: doi:10.1109/SPED.2011.5940729 [Accessed 23 October 2019].

45. SciPy (2017). *scipy.signal.hamming*. [Online] Available at: https://docs.scipy.org/doc/scipy-1.0.0/reference/generated/scipy.signal.hamming.html [Accessed 8 August 2019].

46. Semantic Scholar (2019) *Equivalent rectangular bandwidth*. [Online] Available at: https://www.semanticscholar.org/topic/Equivalent-rectangular-bandwidth/710666 [Accessed 16 August 2019].

47. Shaver, B. (2017). *A Zero-Math Introduction to Markov Chain Monte Carlo Methods*. [Online]. Available at: https://towardsdatascience.com/a-zero-math-introduction-to-markov-chain-monte-carlo-methods-dcba889e0c50 [Accessed 14 August 2019].

48. Shrawankar, U. and Thakare, V. M. (2013). Techniques for Feature Extraction In Speech Recognition System : A Comparative Study. *arXiv:1305.1145 [cs]*. [Online]. Available at: http://arxiv.org/abs/1305.1145 [Accessed 29 October 2018].

49. Smith, J. O. and Abel, J. (2007) *Equivalent Rectangular Bandwidth*. [Online] Available at: https://ccrma.stanford.edu/~jos/bbt/Equivalent_Rectangular_Bandwidth.html [Accessed 7 August 2019].

50. SRI International (2016) *SRILM – The SRI Language Modeling Toolkit*. [Online] Available at: http://www.speech.sri.com/projects/srilm/ [Accessed 12 August 2019].

51. Tamazin, M., Gouda, A. and Khedr, M. (2019). Enhanced Automatic Speech Recognition System Based on Enhancing Power-Normalized Cepstral Coefficients. *Applied Sciences*, 9 (10), p.2166. [Online]. Available at: doi:10.3390/app9102166 [Accessed 16 August 2019].

52. Traunmüller, H. (1990). Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, 88 (1), pp.97–100. [Online]. Available at: doi:10.1121/1.399849 [Accessed 8 August 2019].

53. Van Segbroeck, M. (2018) *featxtra*. [Online] Available at: https://github.com/mvansegbroeck/featxtra [Accessed 20 June 2019].

54. Veselý, K. et al. (2013). Sequence-discriminative training of deep neural networks. In: *The 14th Annual Conference of the International Speech Communication Association INTERSPEECH 2013*. August 2013. Lyon, France: INTERSPEECH. pp.2345-2349.

55. Vimala, C. and Radha, V. (2012). A Review on Speech Recognition Challenges and Approaches. *World of Computer Science and Information Technology Journal (WCSIT)*, 2 (1), pp. 1-7.

56. Volkmann, J., Stevens, S. S. and Newman, E. B. (1937). A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8 (3), pp.208–208. [Online]. Available at: doi:10.1121/1.1901999 [Accessed 4 August 2019].

57. Voxforge (2019) *About Voxforge*. [Online] Available at: http://www.voxforge.org/home/about [Accessed 20 June 2019].

58. Wu, C. (2018). *Structured deep neural networks for speech recognition*. Ph.D., University of Cambridge. [Online]. Available at: doi:10.17863/CAM.23363 [Accessed 22 October 2018].