

AFIT/GE/ENG/98M-02

CHANNEL-MISMATCH COMPENSATION
IN SPEAKER IDENTIFICATION:
FEATURE SELECTION AND ADAPTATION
WITH ARTIFICIAL NEURAL NETWORKS

THESIS

Edmund Albert Fitzgerald
Captain, USAF

AFIT/GE/ENG/98M-02

Approved for public release; distribution unlimited

DTIC QUALITY INSPECTED 4

19980408 122

19980408 122

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

CHANNEL-MISMATCH COMPENSATION
IN SPEAKER IDENTIFICATION:
FEATURE SELECTION AND ADAPTATION
WITH ARTIFICIAL NEURAL NETWORKS

Edmund Albert Fitzgerald, B.S.E.E.

Captain, USAF

Approved:



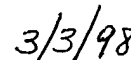
Dr. Martin DeSimio
Thesis Advisor



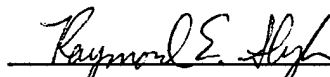
Date



Dr. Timothy Anderson
Committee Member



Date



Dr. Raymond Slyh
Committee Member



Date

AFIT/GE/ENG/98M-02

CHANNEL-MISMATCH COMPENSATION
IN SPEAKER IDENTIFICATION:
FEATURE SELECTION AND ADAPTATION
WITH ARTIFICIAL NEURAL NETWORKS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Edmund Albert Fitzgerald, B.S.E.E.
Captain, USAF

March, 1998

Approved for public release; distribution unlimited

Acknowledgements

First and foremost, I would like to thank my beautiful wife and soulmate, Jill Diana, for supporting me through the trials of AFIT. She was the steady force that kept me focused on what was truly important. Her unselfish love, shown through prayers, encouragement, and constant support serve as a model. She took care of our son almost exclusively while maintaining the finances and household. Jill did this despite our residence being far from friends and family during a serious illness in her family. My success at AFIT serves as a reflection of her successes in our home. And Jack, our son, has been a great blessing, and I appreciate his warmth.

The prayers and encouragement of my loving family, including my father, sister, and in-laws, played no small part in my success.

I thank my advisor, Doctor Marty DeSimio, who taught me several subjects, including Pattern Recognition and Speech Processing with their complex probability theory, in a clear and enjoyable way. He always has a positive, enthusiastic, reasonable, and caring attitude. He is as demanding of his own work and his students, thereby producing a quality student as grounded in the basics as much as the specifics of the research area.

I thank Doctor Tim Anderson and Doctor Ray Slyh for their encouragement, expertise, high standards of work and ethics, and, of course, time despite their busy schedules. They were extremely helpful and reasonable through the entire research process and thesis document development.

I thank my friend Brian Reid who has been a great help in several ways, especially with learning programming techniques in Matlab and C-shell scripts. The templates for Gaussian Mixture Model generation and SID using HTK were developed by Brian. I thank Al Arb, who has helped Brian and I with programming and research ideas in classes and thesis work. I appreciate working with the classes of GE97D, GCS97D, GOR98M, and GE99M, whose esprit de corps, ethics standard, and work ethic make one feel very confident in the personnel of the Armed Forces.

The technical assistance and support from the sponsor of Brian and myself, Captain John Colombi at the National Security Agency, is much appreciated.

I appreciate the teachers who helped me to learn so much and leave AFIT with a great confidence in my knowledge and programming abilities. They include Doctors Peter Hovey, Steve Rogers, Byron Welsh, Vittal Pyati, (Lt Col) Don Gelosh, (Major) Mike Temple, and (Major) Rick Raines.

I also give much credit to the staff in the Engineering and Mission Support section, who personally and professionally cares about the success of students and the strain on their families.

Lastly, I give thanks and praise to the Father, His Son Jesus, and Holy Spirit, the Divine Beings who, fully unified in will, have the name Yahweh. As the God of Abraham has given me life, true purpose, the hope of eternal salvation, a soulmate, and a son, I hope to serve Him faithfully forever.

Edmund Albert Fitzgerald

Table of Contents

	Page
Acknowledgements	iii
List of Figures	viii
List of Tables	ix
Abstract	x
I. Introduction	1-1
1.1 Overview	1-1
1.2 Problem Statement	1-2
1.3 Scope	1-2
1.4 Approach	1-2
1.5 Thesis Organization	1-3
II. Background Theory	2-1
2.1 Introduction	2-1
2.2 Process Overview	2-1
2.3 Feature Selection	2-1
2.4 ANNs for Approximating Functions	2-3
2.5 Mixture Classification Theory	2-4
2.6 Summary	2-5
III. Approach and Methods	3-1
3.1 Introduction	3-1
3.2 TIMIT and NTIMIT Databases	3-1
3.3 Preprocessing of Features	3-1
3.3.1 ESPS Feature Extraction Commands	3-1

	Page
3.3.2 Matlab Feature Preprocessing	3-2
3.4 HTK Gaussian Mixture Models	3-3
3.5 Feature Mapping Artificial Neural Networks	3-4
3.6 Summary	3-7
IV. Results	4-1
4.1 Introduction	4-1
4.2 Mixture Models with Baseline Statistics	4-1
4.3 Train and Test on TIMIT (T/T)	4-1
4.4 Train on TIMIT and Test on NTIMIT (T/N)	4-2
4.5 Train on TIMIT and Test on Transformed NTIMIT Features (T/ANN)	4-3
4.6 Summary	4-4
V. Conclusions	5-1
5.1 Compensation Technique Development	5-1
5.2 Baseline Testing	5-1
5.3 SID with Transformed Features	5-1
5.4 Final Thoughts	5-2
5.5 Recommendations for Further Study	5-2
VI. Appendix	6-1
6.1 Artificial Neural Networks Theory	6-1
6.1.1 Network Classification Theory	6-1
6.1.2 Multi-class ANN	6-1
6.1.3 Multilayer Perceptron Theory	6-2
6.1.4 ANN parameters	6-3
6.1.5 Backpropagation	6-4
6.2 Gaussian Mixture Model Classification Theory	6-6

	Page
6.2.1 Semiparametric Classification Theory	6-6
6.2.2 Expectation-Maximization Theory for Solving the Mix- ture Model Parameters	6-6
Bibliography	BIB-1
Vita	VITA-1

List of Figures

Figure		Page
2.1.	General Diagram of Channel Mismatch Solution	2-1
2.2.	Time Plot and Corresponding Formant Spectrum from Vowel "a" in "at" [4]	2-2
3.1.	Diagram of Speaker Modeling, Baseline SID, and SID after Compens- ation	3-2
3.2.	The Configuration for Training the Feature Mapping ANNs	3-5
3.3.	The Configuration for Transforming the Features through the ANNs	3-6
4.1.	T/T with Combinations of Formants, Bandwidths, and Pitch	4-3
4.2.	T/N with Combinations of Formants, Bandwidths, and Pitch	4-4
4.3.	T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Three Formants and Pitch	4-7
4.4.	T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants and Pitch	4-7
4.5.	T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch	4-8
4.6.	T/ANN(72 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch	4-8
6.1.	Simple Artificial Neural Network	6-1
6.2.	Single-Layer, Multiple Output Network	6-2
6.3.	Multilayer Perceptron	6-3

List of Tables

Table		Page
4.1.	SID Accuracy on T/T and T/N for Several Combinations of Features	4-2
4.2.	SID Accuracy with Combinations of Compensated and NTIMIT Features for Three Formants and Pitch Using ANNs (45 Hidden Nodes)	4-5
4.3.	SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants and Pitch Using ANNs (45 Hidden Nodes)	4-5
4.4.	SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch Using ANNs (45 H-Nodes)	4-5
4.5.	SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch Using ANNs (72 H-Nodes)	4-6

Abstract

We develop and present results of an artificial neural network (ANN) based compensation technique for mismatched classifier training and testing conditions in a speaker identification (SID) task. One ANN per feature per speaker is trained to perform a mapping of that feature from a corrupted condition to an undistorted condition. Therefore, a classifier trained under one condition may be used to classify data collected under a different condition.

Speech utterances from 168 speakers, collected in a studio, and also re-recorded after transmission over telephone networks, are used for developing and testing the method. Peak formant resonant frequencies, their bandwidths, and pitch are used as features. These features from the studio speech are used to train Gaussian Mixture Model classifiers. Portions of the studio and telephone speech are used to train the compensation ANNs. In mismatched train and test conditions, features from telephone speech are modified by the trained ANNs and applied to the GMMs trained with features from studio speech.

Without compensation, SID accuracy is 6%. The compensation method developed in this work provides mismatch SID accuracy of 58.3%. Previous research on the same data with the commonly used Mel-Frequency Cepstral Coefficients as features and a typical compensation method of Cepstral Mean Subtraction with Band-Limiting gives SID accuracy of 27.4% with the same type of classifiers.

CHANNEL-MISMATCH COMPENSATION
IN SPEAKER IDENTIFICATION:
FEATURE SELECTION AND ADAPTATION
WITH ARTIFICIAL NEURAL NETWORKS

I. Introduction

1.1 Overview

Speaker recognition is the process of identifying a person from the characteristics of their voice. As performed on computers, speaker identification (SID) is the task of choosing one speaker model from a set of models best matching the utterance given. The models are formed from past information and are stored computer memory for eventual SID testing comparisons. A difficulty arises when the environmental conditions under which the models were formed are different from those conditions associated with the new utterance to be tested. Published attempts to solve this channel-mismatch problem have resulted in relatively poor results [13], [20], [22], while SID under common training and testing conditons is generally considered solved [3], [7], [13], [20].

Although ideal theoretically, training speaker reference models under the same degradations as the test features is often not realistic, and the training of models under noisy conditions often still leads to sub-optimum results, whether or not the testing is done under better conditions [3], [6]. Therefore, the goal is to create an identification system robust to mismatched training and testing conditions. In this study, the mismatch involves the difference between studio-quality speech and the same speech distorted over telephone lines and equipment. This simulates conditions for remote access by deployed military personnel to secure electronic equipment, as one example.

1.2 Problem Statement

Investigate features which are robust to channel mismatch and implement a process to address the effects of mismatched training and testing conditions on those features in SID.

1.3 Scope

Features from the 168 test speaker set of the Texas Instruments and Massachusetts Institute of Technology's TIMIT and NYNEX's NTIMIT databases were used to form speaker models and to manipulate through compensation methods for SID testing. The general objectives to investigate the previously stated-problem follows:

- Based on previous research, choose features proven to have reasonable speaker-specific characteristics to achieve high SID accuracy.
- Train models using part of TIMIT, a studio-quality database.
- Perform SID testing using the speaker models against only studio-quality speech to confirm choice of features and maximum achievable results.
- Perform SID testing using the speaker models against some of the distorted speech in NTIMIT utterances to achieve baseline statistics on SID under uncompensated, mismatched conditions.
- Develop channel mismatch compensation technique using portions of TIMIT and NTIMIT databases not used in SID testing.
- Test compensation technique through SID testing with the trained models.

1.4 Approach

Based on Sambur's work [8] and preliminary experiments, formant resonance frequencies, their corresponding bandwidths, and pitch were chosen as reasonable features to pursue. Baseline statistics were obtained for SID for same-channel and cross-channel conditions. That is, speaker models were created by using TIMIT training sentences, and testing was done on TIMIT and NTIMIT utterances. Artificial neural networks (ANNs)

were then used to compensate for the effects of the channel in NTIMIT before obtaining SID rates for comparison to the baseline. The ANNs were used in feature mapping or function approximation, attempting to undo the effects of the telephone channel.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter II provides information on the theory of features, the semi-parametric classifiers, and ANN functional mappers we used. Chapter III explains the methodology we implemented in the techniques and experiments. Chapter IV contains our results, and Chapter V contains our conclusions and recommendations.

II. Background Theory

2.1 Introduction

This chapter provides the necessary theoretical foundation for SID and ANNs. The biological and mathematical basis for the selected features is followed by a discussion on ANN function approximation and semiparametric classification theory.

2.2 Process Overview

Figure 2.1 shows our general SID process developed from the theories that will be discussed in this chapter. Feature selection and extraction was used for both speaker model generation and the creation of mapping ANNs. Original, undistorted features were SID tested against the speaker models for confirmation of feature selection and to establish maximum expectations for compensation results. Uncompensated features were SID tested for baseline statistics to compare the results when we used features mapped by the ANNs.

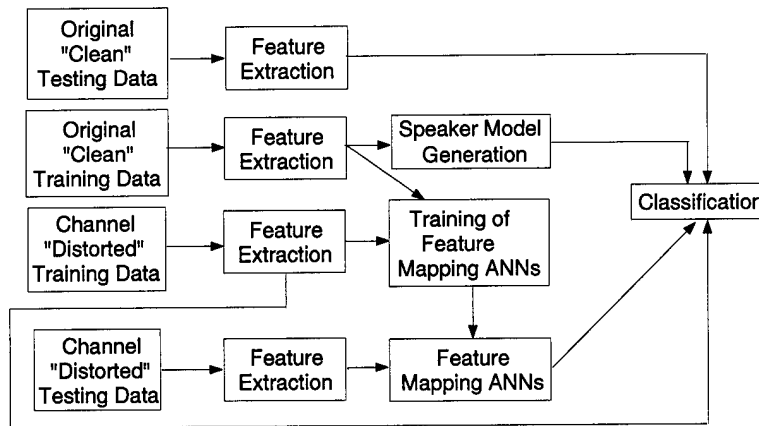


Figure 2.1 General Diagram of Channel Mismatch Solution

2.3 Feature Selection

The most commonly used and accepted features for speaker recognition are Mel-Frequency Cepstral Coefficients (MFCCs) [3], [5], [6], [10], [12], [13]. To obtain MFCCs, a discrete fourier transform (DFT) is performed on the sampled speech segment and processed through a series of triangular filters spaced along a Mel-frequency scale; the Mel-

frequency scale is a nonlinear frequency scale representing characteristics of human hearing. The log of the output magnitudes of the filters is then calculated and processed by a transform, usually a discrete cosine transform, with the corresponding coefficients being the MFCCs for the sampled speech segment.

SID accuracy plummets when MFCCs are used in conditions with channel mismatch [13], [20]. The results are not promising, even with compensation [13], [20], [22]. Therefore, we sought a different set of features. Sambur [8] demonstrated that the formants and pitch are good features for similar training and testing conditions, although not to the extent MFCCs have been.

Formants are the resonant frequencies the vocal tract imposes onto the signal coming from the diaphragm. For voiced phonemes, speech production generally can be modeled by a quasi-periodic pulse-train generator with spectral modulation occurring through a cavity, the vocal tract. The excitation from the diaphragm is changed by the glottis, a cartilage plate, by manipulating and stretching the adjacent vocal cords as air is passed through. If the vocal cords are vibrating, phonation occurs and the segment of speech is declared voiced. If the waveform is instead aperiodic or random due to the vocal cords not oscillating, it is unvoiced. Modulation is imposed on the glottal waveform by the vocal tract. The vocal tract adds its inherent natural resonances according to the current shape of the tract. These resonant frequencies are formants, and, in this thesis, their peak frequencies will be referred to as formants. The particular formant spectrum structures resulting from the unique shapes of the vocal tract characterize all vowels and some consonants in the English language [5]. The fundamental frequency, the reciprocal of the fundamental period, is called the pitch. An example is given in Figure 2.2, showing a sampled speech segment and its spectrum with the first four formants from the vowel /a/ as in "at".

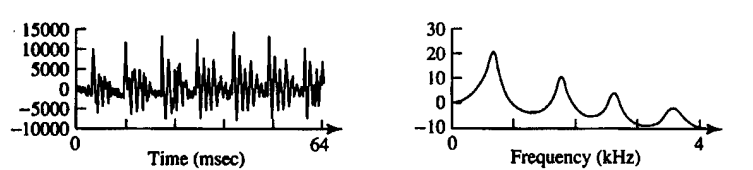


Figure 2.2 Time Plot and Corresponding Formant Spectrum from Vowel "a" in "at" [4]

Sambur [8] performed feature saliency tests on 92 features including formants, bandwidths, pitch, formant contours, glottal-source poles, and nasal pole locations. These general parameters were further subdivided within the 92 by being from particular words; for example, one of the best features he cited was the third formant in the vowel *u*. In his final list of best features, most of the top twenty were the first four formants and pitch from different phonemes; none of these five were particularly dominant over the other four in this list. Besides Sambur, Parsons [5] also recommends using at least three formants for speaker recognition, which includes SID. And since bandwidths are associated with these formants, we felt they should be included along with formants and pitch and potential features for addressing the channel mismatch problem.

2.4 ANNs for Approximating Functions

Although multilayer perceptron (MLP) artificial neural networks (ANNs) are often used for classification of data, they can be used to approximate functions also. Refer to the appendix for background ANN theory. More specifically, MLP-ANNs with two layer of weights and nonlinear activation functions can approximate arbitrarily well any continuous functional mapping from one finite-space to another. This fact is true as long as the number of hidden layer units is sufficient and the number of target nodes does not exceed the number of input nodes [1]. There is a wealth of published papers on this subject [23], [24], [25], [26], [27], [28], [29], [30], [31].

Instead of training the ANNs to connect certain inputs to particular classes, in our application we train ANNs to function like an inverse channel filters by having the output training targets be the values we desire the corresponding training inputs to become. Then we input the corrupted test features to the ANNs with the purpose of "cleaning" those features.

In very general terms, adding additional layers can cause a decrease in typical ANN error criterion. More specifically, with nonlinear function approximations, multiple layers with nonlinear activation functions lead to a much higher probability of lowering the standard error parameter, Sum-Squared-Error (SSE), to an acceptable level [1]. The use of backpropagation, whereby the error derivatives of the network outputs are propagated

back to the hidden layers to be used in their error metric, is a technique for adjusting weights to minimize SSE. While updating the weights based on the errors relating to the weights can lead to convergence, the use of an adaptive learning rate and momentum greatly assists finding a global minimum. The adaptive learning rate is a fractional multiplicative term for adjusting the change in weight updates according to the level of error change. For example, it is desirable to make larger changes in weight values when the SSE is decreasing rapidly. Momentum assists in centering on a global minimum by causing the weight changes to be based on previous weight changes [19].

2.5 Mixture Classification Theory

Statistical classifiers require estimates of class conditional probability density functions (PDFs). Semiparametric methods of classification often prove to be ideal since they can combine good aspects of both nonparametric and parametric approaches. Avoiding the problem of model growth directly with the size of the data set, the model only becomes more sophisticated with data expansion. One type of a semiparametric method is the Gaussian Mixture Model (GMM). GMMs, given the necessary number of components with corresponding appropriate parameters, can approximate any non-disjoint density to a desired accuracy [1]. GMMs have been applied with great success to SID tasks, approximating even multimodal PDFs very well [7].

The GMM is simply a linear, weighted combination of M basis functions, here normal probability density functions:

$$p(x) = \sum_{j=1}^M p(x | j) P(j) \quad (2.1)$$

where $p(x | j)$ is a normal probability density function,

$$p(x | j) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp -\frac{1}{2} \frac{(x - \mu_j)^2}{2\sigma_j^2} \quad (2.2)$$

and $P(j)$ is the mixture weight, μ_j is the mean, and σ_j is the standard deviation for mixture component j .

After the GMM is formed by inputting training data, an error metric, the log-likelihood equation, is used for a set of test observations [1] :

$$E = -\ln \mathcal{L} = -\sum_{n=1}^N \ln p(x_n) = -\sum_{n=1}^N \ln \left\{ \sum_{j=1}^M p(x_n | j) P(j) \right\}. \quad (2.3)$$

The error is minimized by maximizing the likelihood score, *i.e.* the Maximum A posteriori Probability (MAP). Therefore, in SID the GMM speaker model with the highest likelihood score given the utterance would be considered the identified speaker. See the Appendix for more information on classification theory.

2.6 Summary

Based on poor performance of SID under mismatched conditions with MFCCs [13], [20], [22] and work by Sambur [8], formants, bandwidths, and pitch were chosen as features. As good features for modeling the vocal tract, these features must be taken from speech segments declared voiced, since those type of features better model the entire vocal tract than those from unvoiced speech. ANNs were chosen as a transforming compensation technique for the effects of mismatch on the features. The mapped features were used in final testing with Gaussian Mixture Speaker Models, which have been shown [7] to have good performance characteristics in SID applications.

III. Approach and Methods

3.1 Introduction

This chapter describes the databases and expands on feature, ANN, and GMM theories by demonstrating their use for within-channel and cross-channel SID applications.

3.2 TIMIT and NTIMIT Databases

We used the 168 test speaker set of the Texas Instruments and Massachusetts Institute of Technology's TIMIT and NYNEX's NTIMIT databases. NTIMIT is rerecorded TIMIT with the use of a carbon-button telephone handset and artificial mouth sent over various length telephone lines and looped back for recording at a 16 kHz sampling rate [15]. Although originally designed for speech recognition research, TIMIT is a good database for SID under an almost ideal environment given its specifications: eight KHz bandwidth, minimum equipment noise and variability, and depth in phonetic diversity in approximately three second utterance lengths.

The 168 speakers are divided into eight dialect regional subsets, varying size from 11 to 32 speakers. Ten sentences were recorded from each of the 168 speakers. For diversity, the ten sentences per speaker are divided into two sentences with *sa* designations, three with *si*, and five with *sx*. The two *sa* are identical for all speakers, while the *si* and *sx* sentences are not. For testing purposes, the last two *sx*, by using the numerical ordering from UNIX *ls* command, were used for testing as the other eight sentences were used in training. The training set included the two *sa* to insure some completely common conditions for all speakers for GMM generation [7]. The two *sx* were used to simulate more of a real-world text-independent condition.

3.3 Preprocessing of Features

3.3.1 ESPS Feature Extraction Commands. Figure 3.1 helps clarify the following discussion. The pitch values in each frame and the corresponding probabilities of voiced speech were obtained with ESPS 5.1's *getf0* command through a C-shell script. The command *getf0* is similar to an algorithm developed by Secrest and Doddington [14].

Using a correlation function on filtered (one-pole model) linear prediction residuals, potential pitch values are obtained and then used along with spectral consistency penalties over many frames and voicing state information to make final determination of pitch terms [16].

The formant and bandwidths are obtained via ESPS 5.1's *formant* command in a C-Shell script. They are selected from potential values given by the solution for the linear prediction polynomials' roots. Cost penalties are imposed on these candidates, and the final terms are a result of a modified Viterbi algorithm for achieving an optimum mapping regarding consistency over multiple frames of the speech waveform. A preemphasis value of 0.7 was used [4], [6], [16]. With a desire to achieve the multimodal sharpness of the spectra without distortion, we chose a linear prediction coefficient (LPC) order of twelve [6], [16]. And for smoothing of the frame transitions, a Hamming window was applied to each 20 millisecond frame with a ten millisecond step size. The preemphasis value, the LPC order, and the step size were all software defaults. The other parameters were common choices [6]. The choice of these values was not of central importance to this thesis, as was the SID performance when these features are undistorted, corrupted, and transformed.

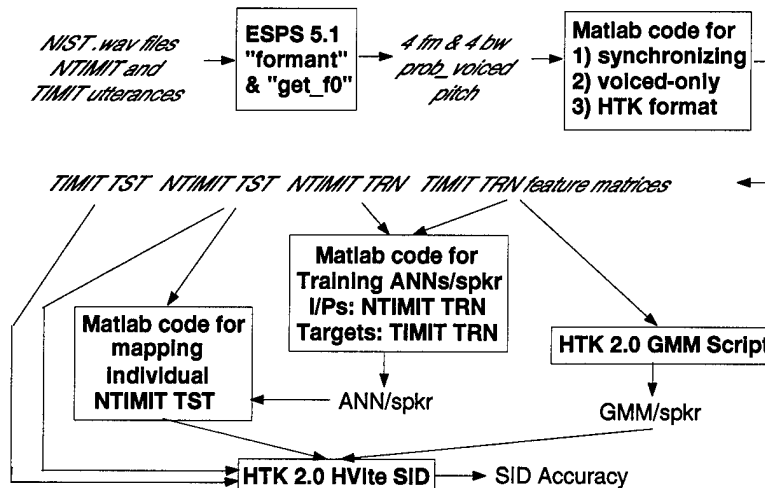


Figure 3.1 Diagram of Speaker Modeling, Baseline SID, and SID after Compensation

3.3.2 Matlab Feature Preprocessing. After the generation of outputs from the *formant* and *getf0* commands, the formants, bandwidths, pitch, and probabilities of voiced speech were input to a data manipulation program written in Mathworks Matlab 4.2c. As

evident by file headers, the outputs of each of the commands have different starting times which caused practically a one-frame shift between the two. Also, *getf0* might have one and occasionally two extra frames, disregarding the shift mentioned. Therefore, correspondence had to be achieved primarily since the ANNs are to be direct feature mappers. Furthermore, accurate values of pitch (being a measurement of periodicity) and formants for proper modeling of the vocal tract can only come from voiced segments. Thus, we needed to eliminate unvoiced frames. We did this by employing the probabilities of voiced speech to eradicate useless data. The use of 0.5 as a decision threshold for voicing probabilities caused no complications as the probabilities from *getf0* were typically zero or one, with occasionally a one-one millionth term or a 0.99 value.

Then, since first formant ranges for vowels, voiced fricatives and voiced stops are typically below 1000 Hz [4], [6] and pitch values in normally read sentences should not generally exceed 160 Hz for males and 400 Hz for females [5], criteria for unlikely values can be developed. Extreme outliers, such as first formants at 2000 Hz, triggered a part of the preprocessing algorithm to eliminate the entire feature vector, since other values might also be impacted. We chose a threshold of plus or minus two standard deviations, as this was found to typically eliminate about three percent of the original feature vectors of each speaker's feature matrices. The use of probabilities of voiced speech eliminated about 50% of the typically 250 original feature vectors per utterance.

3.4 HTK Gaussian Mixture Models

GMMs can be implemented with HTK 2.0, since GMMs are single-state Hidden Markov Models (HMMs). GMMs were created by using feature vectors from the eight training sentences from each speaker in a C-shell script which included HTK commands *HInit*, *HRest*, and *HHEd*. *HInit* provides initial estimates for the means and variances of the component densities in a GMM. *HInit* functions by repeatedly segmenting training data by Viterbi alignment and recalculating the means and variances using a K-Means clustering algorithm. *HRest* uses an EM/Baum-Welch algorithm for re-estimation of the GMM parameters to best model the feature vectors' probability density for individual speakers; this theory is further discussed in the Appendix. We set the variance floor at

0.01 for comparison to previous research [7], [13]. After initially modeling with one, the amount of non-defunct mixture components was grown by an option with a mixture editor, *HHEd*. The mixture with largest weight is then duplicated. Each twin mixture then has their weight halved and is offset from the original mean by plus or minus two standard deviations [13], [17].

For SID, a C-Shell script is implemented which uses the HTK 2.0 *HVite* command. This command uses Viterbi algorithmic techniques to test trained models with a set of feature vectors from a speaker. The object is to find the model with the greatest log-likelihood score given a set of feature vectors from a particular utterance.

3.5 Feature Mapping Artificial Neural Networks

The main problem to address is the nonlinear nature of the handset, while accounting for the bandwidth limitations of the telephone networks [3]. This problem was evident from equipment specification, spectrum study, previous research [3], and some attempts at solving the channel mismatch problem based on linear models such as Missing Features [11]. As previously discussed, ANNs became a logical choice to address this nonlinear problem. Since the speaker models were trained on TIMIT, it would be ideal if the channel-distorted test utterances from NTIMIT actually could be made to appear as if no distortion had occurred. We used Mathworks Matlab Neural Networks Toolbox (MM-NNT) 2.0b to construct a training method for function-approximating ANNs. The method inputs all the particular speaker's training feature matrices from NTIMIT and sets a corresponding target output of the TIMIT feature matrices. This is supervised training [1], [9].

The first experiments were done by inputting the parts of the feature matrices with only formants, and setting the target output as only one feature. Therefore, one ANN per feature per speaker had to be created; see Figure 3.2. We used this architecture and set of features due to Sambur's list of best features [8]. An algorithm for insuring convergence and a reasonable SSE was included in the ANN-generation program. The resultant trained ANNs often took longer than originally anticipated since convergence was not automatic given the initial weights of each training iteration. Reaching an acceptable SSE often occurred only after several loop iterations. Each loop iteration caused a reinitialization of

The Training of the Feature Mapping Artificial Neural Networks

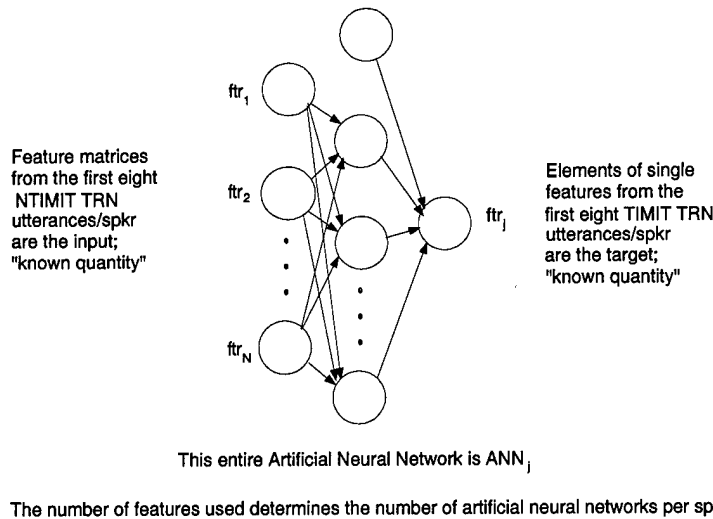


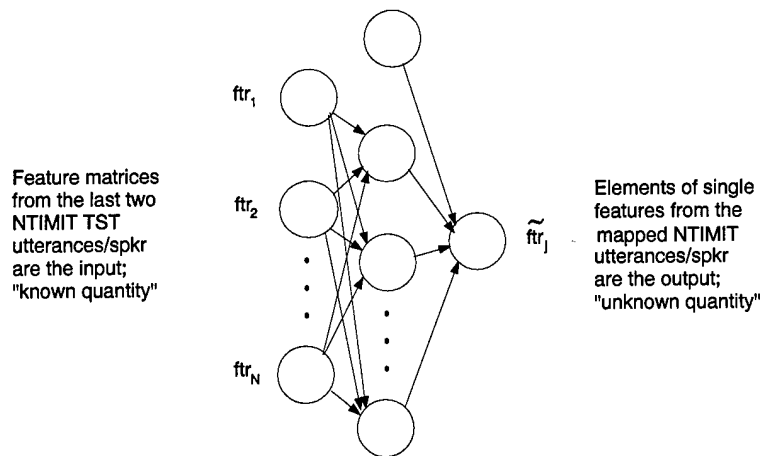
Figure 3.2 The Configuration for Training the Feature Mapping ANNs

ANN weights given by the MM-NNT *initff.m* routine at the beginning. This last step was needed, since experiments were completed on a smaller scale to obtain the correct number of hidden node weights and ANN parameters.

After the complete training of each speaker's ANNs for each feature, the feature vectors from individual test utterances were input to each of the ANNs as shown in Figure 3.3. The transformed feature outputs from each ANN were then recombined into a transformed NTIMIT feature matrix. As evident by Chapter IV of this thesis, some features appeared from sample study to not be transform well to the corresponding TIMIT feature matrix. Therefore, other feature combinations were also tried in SID, such as using one or more original NTIMIT features along with other transformed features.

To elaborate further on the neural network architecture, a number of attempts were made to find the parameters which were optimized in the sense of SSE. When considering four formants as inputs and one as output, the trials with subsets of speakers were performed with hidden nodes ranging from three to 25. ANNs with twelve hidden nodes achieved optimum results, though up to 20 was close. We implemented the idea, partially based on previous research, that the number of hidden layer nodes increasing with input layer complexity could be tied to their ratio. In this case, the ratios giving best results had

The Usage of the Feature Mapping Artificial Neural Networks



This entire Artificial Neural Network is ANN_j

The outputs are then grouped into matrices to be used in HVite SID.

Figure 3.3 The Configuration for Transforming the Features through the ANNs

ratios of nodes between 3:1 and 5:1. This empirical theory was used with other feature combinations for validity. With nine inputs of formants, bandwidths and pitch, 45 hidden nodes, i.e. 5:1 ratio, gave the best results. Decreasing the amount of nodes towards 3:1 ratios caused a rapid degradation in accuracy. Increasing to 72 hidden nodes yielded good, but lower percentages. We also noted the ANNs with 72 nodes took two to three times as long to train as those with 45 hidden nodes.

The ANN parameter search was appropriate for the full speaker set though done on a subset, since the ANNs were trained on individual speakers. We settled on a momentum rate of 0.9 with a learning rate decrease (LRD) of 0.5, versus the cited rates of 0.95 [19] and default of 0.7, after a number of subset trials. The LRD is a multiplying fraction which reduces the momentum term when an error increase is encountered.

The initial parameter search trials also demonstrated the combination of the non-linear activation function logistic sigmoid for the output layer and hyperbolic tangent for the hidden layer proved to be the optimum pair. Before processing, the feature elements were all divided by 10000 to keep them within the range of the activation functions. After

being transformed, they were multiplied by that same constant to achieve useful values for classification.

3.6 Summary

The 168 test speaker set of TIMIT and NTIMIT were used in this study. Although originally intended for use in speech recognition research, the databases' characteristics, including richness in phonemes and dialects, make it suitable for use in SID. The extraction of features from these sampled speech databases was done primarily through the use of ESPS 5.1's tools *formant* and *getf0*. They both use linear prediction techniques for estimation, then assign cost penalties related to consistency to assist in making final value determinations.

Recognizing GMMs are single-state HMMs, HTK 2.0 provided the means for generating GMMs for each speaker in the databases. A C-Shell script with the initializing (*HInit*), growing (*HHed*), and parameter re-estimating (*HRest*) of the GMMs was used.

To address the apparent nonlinear nature of the mismatch problem, ANNs were created to transform corrupted features towards undistorted ones. After small scale trials that determined the general ANN architecture, it became clear features needed to be fed into an individual ANN for each mapped feature desired. Various feature combinations with formants, bandwidths, and pitch were tried with different numbers of hidden nodes and adjustments of ANN training parameters. Through experimentation, we determined fairly optimum ANN architectures before total feature mapping and SID experiments were performed.

IV. Results

4.1 Introduction

This chapter provides baseline statistics on within-channel and cross-channel conditions. These will be followed with the results of SID with features transformed by ANN channel compensation techniques. A discussion of the results includes explanations for certain feature combinations out-performing others.

4.2 Mixture Models with Baseline Statistics

Reynolds [15] showed that the best method for finding the optimum mixture amount was through empirical methods. Bishop [1] alludes to this parameter search for optimization regarding GMMs. Therefore, a search was done to determine the appropriate number of mixture components for maximum accuracy. Some initial results demonstrated there was not enough diversity in the data to support the 32 mixtures Reynolds found to be optimum for his use [15]. So we tested GMMs with maximum mixtures of two to 16 (or more as necessary) until a peak and accuracy descent was found.

4.3 Train and Test on TIMIT (T/T)

As the results of Sambur [8] indicated, the use of four formants and pitch provided respectable SID accuracy rates with a peak of 82.7% with GMMs of ten mixtures while training and testing on TIMIT. In the tables and charts, f represents formant frequency, b stands for bandwidth, and p represents pitch; the adjacent numbers indicate the *particular* formant frequency or bandwidth.

It is reasonable to hypothesize from Sambur's work the elimination of one of these good features would lead to a decrease in accuracy. This is demonstrated by the results when one formant, the fourth, was eliminated as well as when pitch was not included. Observing the results with ten mixture components in Table 4.1 and Figure 4.1, the results dropped to 50.9 % and 69.3 % , respectively, with little improvement with different amounts of mixtures. This demonstrates the value of pitch and, especially, the fourth formant. The bandwidths were included in this study and found to be valuable, increasing the within-

Table 4.1 SID Accuracy on T/T and T/N for Several Combinations of Features

Number of	T/T	T/N	T/T	T/N	T/T	T/N	T/T	T/N
Mixtures	fb1fb2	fb1fb2	f1f2	f1f2	f1f2	f1f2	f1f2	f1f2
per GMM	fb3fb4p	fb3fb4p	f3f4p	f3f4p	f3f4	f3f4	f3p	f3p
2	72.6	5.9	59.8	6.0	38.0	2.1		
4	86.0	6.3	80.4	9.2	35.0	2.7		
6	91.1	6.8	80.4	6.0	69.9	3.0		
8	92.3	5.9	81.3	7.1	69.9	3.0	52.7	12.2
10	92.3	6.3	82.7	7.7	69.3	2.4	50.9	11.3
12	91.1	6.3	79.2	7.1	70.5	2.4	55.7	10.4
14	90.0	6.5	78.9	7.1	71.7	2.4	54.2	12.2
16	87.5	5.9	76.2	6.5	71.7	7.4	52.1	11.9
18					70.5			
20					68.2			
22					66.4			
24					64.3			

channel SID accuracy to a peak of 92.3 %. Some additional conclusions to be drawn relate to the usefulness of the features with reasonable values of cross-correlation, i.e. formants and bandwidths, to be used in an ANN for mapping of testing feature matrices from NTIMIT.

4.4 Train on TIMIT and Test on NTIMIT (T/N)

There is a significant drop in SID accuracy when measuring SID with channel mismatch conditions.

Observing Table 4.1 and Figure 4.2, the best results seem to occur with three formants and pitch; with eight or 14 components, SID accuracy was 12.2%. This seems to verify the need for a good fourth formant and, conversely, the degradation with a poor fourth formant. The degradation occurs by both the upper bandwidth limitations of the channel and the nonlinearities of the microphone [3]. The bandwidths made little difference, but the pitch estimates, in a relative sense, were valuable as a feature under those test conditions. We feel the few percentage differences are of no great significance, since SID rates of 10% are of no practical use.

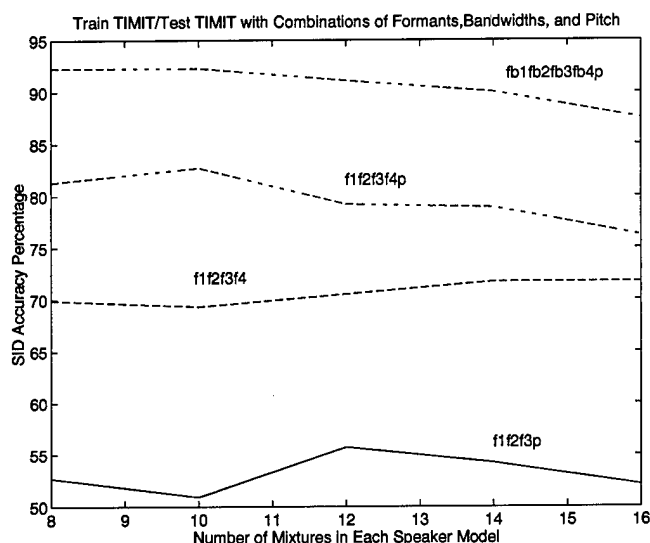


Figure 4.1 T/T with Combinations of Formants, Bandwidths, and Pitch

4.5 Train on TIMIT and Test on Transformed NTIMIT Features (T/ANN)

The results of the ANNs were promising. See Tables 4.2 through 4.5 and Figures 4.3 through 4.6, while recognizing a mapped feature is signified by a tilde on top. Although valuable as a feature, the pitch from the ANNs both in early small scale attempts and with the full ANNs did not map well to the desired TIMIT feature elements. The original NTIMIT pitch was, therefore, usually substituted. The use of all nine available features is shown in Tables 4.4 and 4.5 and Figures 4.5 and 4.6. The accuracies each increase about 20% with ANNs with 45 hidden nodes and 15% with 72 nodes, regardless of the amount of mixtures in the models, when uncompensated pitch is substituted for the transformed pitch.

Just as the importance of the fourth formant with or without its bandwidth was shown by Sambur, it is also shown by the pre- and post-compensation accuracies. It seems the ANNs are able to reconstruct the missing fourth formant from the available formant structure and the cross-correlation of the features. The peak accuracy for all three transformed formants with uncompensated pitch was 23.5%, while the use of the fourth formant in the ANNs and as a feature increased this combination's rate to 31.0%.

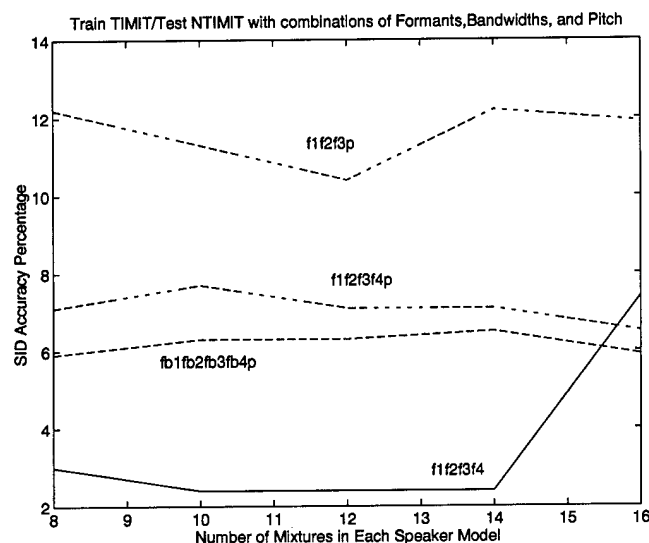


Figure 4.2 T/N with Combinations of Formants, Bandwidths, and Pitch

In fact, this addition raised the SID rates with all mapped combinations, peaking at 34.2% if all the uncompensated second formants were used instead of the transformed versions.

The incorporation of bandwidth as another feature and ANN input boosted rates again, peaking at 58.3% with 45 nodes and 54.2% with 72 nodes when uncompensated pitch was the sole, unmapped feature. Much of this data also shows the presence of the fourth formant in the ANNs greatly assists in the reconstructing of the third and, to a lesser extent, second formant.

4.6 Summary

The use of all features easily outperformed any feature subset combination when no channel mismatch is involved. The SID rate of 92.3% with four formants, bandwidths, and pitch was excellent, confirming the choice of features as reasonable for SID. All rates plummet to about ten percent or less upon cross-channel conditions. Very promising results occurred with the use of certain combinations of transformed features. Generally, the uncompensated pitch provided better data as the ANNs were apparently not able to transform the pitch well. And the importance of the fourth formant was established in several ways. They include the better uncompensated, cross-channel performance when

Table 4.2 SID Accuracy with Combinations of Compensated and NTIMIT Features for Three Formants and Pitch Using ANNs (45 Hidden Nodes)

Number of Mixtures per GMM	T/T	T/N	(T/ANN) $\widetilde{f_1}\widetilde{f_2}\widetilde{f_3p}$	(T/ANN) $\widetilde{f_1}f_2f_3p$	(T/ANN) $\widetilde{f_1}\widetilde{f_2}f_3p$	(T/ANN) $\widetilde{f_1}f_2\widetilde{f_3p}$
2	37.5	7.7				
4	48.8	9.8				
6	50.0	11.9				
8	52.7	12.2	21.1	17.6	18.2	14.0
10	50.9	11.3	23.5	19.0	13.1	14.9
12	55.7	10.4	21.4	18.4	14.3	14.3
14	54.2	12.2	22.0	18.2	16.7	12.3
16	52.1	11.9	22.0	17.6	16.7	14.0

Table 4.3 SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants and Pitch Using ANNs (45 Hidden Nodes)

Number of Mixtures per GMM	T/T	T/N	(T/ANN) $\widetilde{f_1}\widetilde{f_2}\widetilde{f_3}\widetilde{f_4p}$	(T/ANN) $\widetilde{f_1}f_2f_3\widetilde{f_4p}$	(T/ANN) $\widetilde{f_1}\widetilde{f_2}f_3\widetilde{f_4p}$	(T/ANN) $\widetilde{f_1}f_2\widetilde{f_3}\widetilde{f_4p}$
8	81.3	7.1	29.8	22.0	19.6	33.6
10	82.7	7.7	31.0	24.4	22.6	34.2
12	79.2	7.1	29.2	26.2	22.0	32.7
14	78.9	7.1	27.4	25.6	21.4	32.7
16	76.2	6.5	29.2	23.8	20.2	31.0

Table 4.4 SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch Using ANNs (45 H-Nodes)

Number of Mixtures per GMM	T/T	T/N	(T/ANN) $\widetilde{fb_1}\widetilde{fb_2}\widetilde{fb_3}\widetilde{fb_4p}$	(T/ANN) $\widetilde{fb_1}\widetilde{fb_2}fb_3\widetilde{fb_4p}$	(T/ANN) $\widetilde{fb_1}fb_2\widetilde{fb_3}\widetilde{fb_4p}$	(T/ANN) $\widetilde{fb_1}\widetilde{fb_2}fb_3fb_4p$
8	92.3	5.9	58.3	38.1	33.3	18.4
10	92.3	6.3	53.9	35.4	32.1	22.9
12	91.1	6.3	54.2	35.7	33.6	17.6
14	90.0	6.5	50.3	32.1	30.4	27.4
16	87.5	5.9	49.4	32.4	29.2	17.9

Table 4.5 SID Accuracy with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch Using ANNs (72 H-Nodes)

Number of Mixtures per GMM	T/T	T/N	(T/ANN) $\widetilde{fb1}\widetilde{fb2}\widetilde{fb3}\widetilde{fb4p}$	(T/ANN) $\widetilde{fb1}\widetilde{fb2}\widetilde{fb3}\widetilde{fb4\tilde{p}}$	(T/ANN) $\widetilde{fb1}\widetilde{fb2}\widetilde{fb3}\widetilde{fb4p}$	(T/ANN) $\widetilde{fb1}\widetilde{fb2}\widetilde{fb3}\widetilde{fb4p}$
8	92.3	5.9	54.2	36.0	36.0	19.0
10	92.3	6.3	53.0	37.5	35.4	18.4
12	91.1	6.3	49.7	32.1	31.0	17.6
14	90.0	6.5	45.2	31.0	29.2	15.2
16	87.5	5.9	47.6	34.2	31.8	17.3

not used, its effects when transformed properly, and its value in reconstructing the third and, to a lesser extent, the second formant. The best results were when all nine features were input to each ANN, and all transformed output features were used except pitch; the distorted version was substituted as closer to TIMIT target. The SID accuracy peaked at 58.3%, up from 5.9% with no compensation for eight mixtures components.

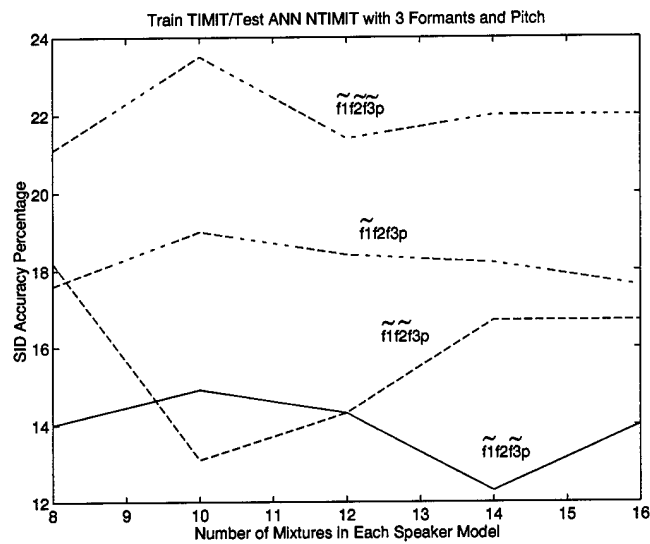


Figure 4.3 T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Three Formants and Pitch

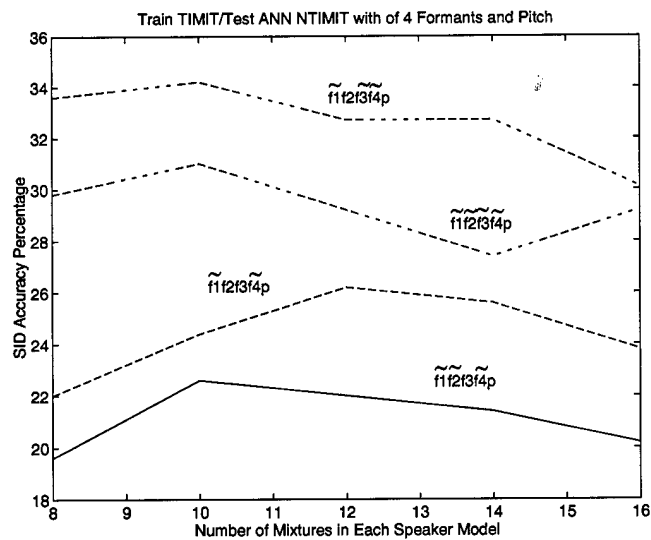


Figure 4.4 T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants and Pitch

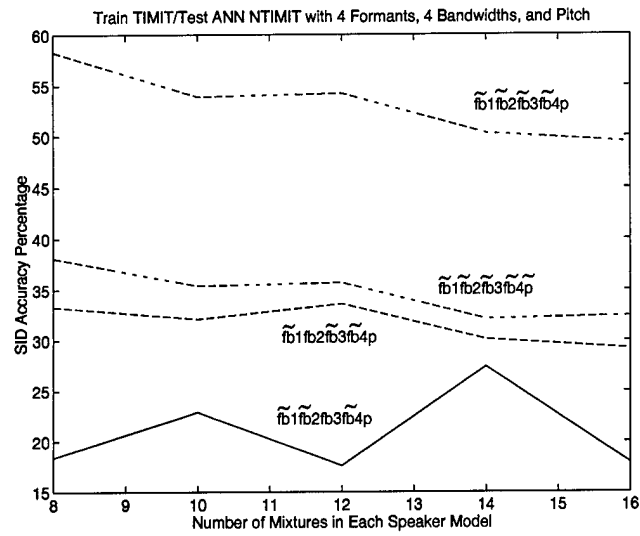


Figure 4.5 T/ANN(45 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch

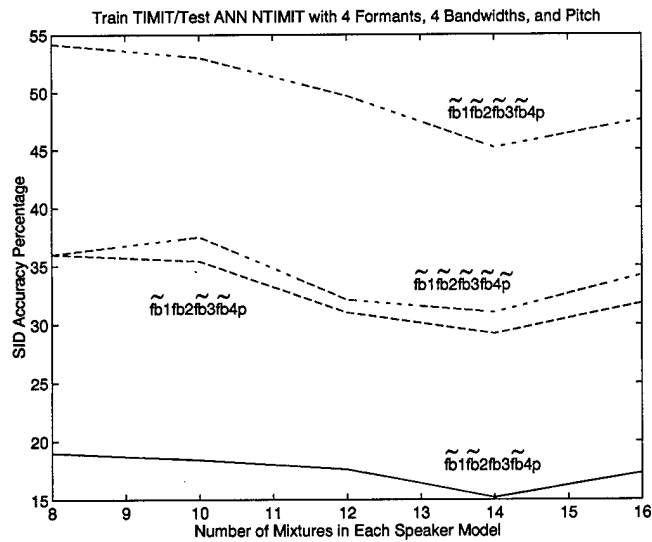


Figure 4.6 T/ANN(72 Nodes) with Combinations of Compensated and NTIMIT Features for Four Formants, Bandwidths and Pitch

V. Conclusions

5.1 Compensation Technique Development

Through early testing with other compensation methods [11], [21], previous research [3], and spectrum analysis, it became apparent to us the chief cross-channel SID problem was the nonlinear distortion imposed by the microphone. We felt a compensation technique using ANNs should be developed. After many architectural attempts, best results were found when using all nine features to train individual ANNs for mapping each feature for each speaker.

5.2 Baseline Testing

With channel conditions kept consistent with training and testing, good SID accuracies of over 92% were obtained by GMMs with formants, bandwidths, and pitch as features. This accuracy result experimentally confirmed our choice of features and our use of GMMs as a classification tool. Telephone equipment and network distortions caused large degradations in the ability to use the previously trained GMMs for SID. When we used ANNs to compensate for the distortion effects, improvements were substantial with various feature combinations. For example, the peak accuracy using features transformed with our technique across all 168 speaker models was 58.3%. This compares favorably to the 5.9% with no channel mismatch compensation and the 27.4% from previous research with different features, MFCCs, and another compensation technique.

5.3 SID with Transformed Features

The best results for channel compensation were when structuring the ANNs with nine inputs and one output, and then taking all of the transformed output feature data except the pitch. Although good results occurred when using the cleaned pitch, much better results (typically 20% improvement) were given when substituting the original NTIMIT pitch values. This improvement was also verified when using the other feature combinations which lacked bandwidth. When comparing various examples of the training inputs and

outputs, pitch does not seem to transform well to the ideal targets. Theoretically, this probably is due to the lower cross-correlation between the pitch and the other features.

When comparing many of the inverse covariance matrices HTK 2.0 gives for the GMMs of each speaker, calculations show the autocorrelation of the pitch to typically be the lowest value of any element. The autocorrelation values often are two orders of magnitude less than the other autocorrelation values. And the cross-correlation values with pitch are most often among the average or lower values of the matrices. Consistent in a general sense with Sambur's results [8] was the outcome that pitch was a valuable feature and did improve SID.

The importance of the fourth formant with or without its bandwidth was shown by Sambur, as is also shown by the pre- and post-compensation accuracies. It seems the ANNs are able to reconstruct the missing fourth formant from the available formant structure and the cross-correlation of the features. It also seems the presence of the fourth formant in the ANNs greatly assists in the reconstructing of the third and, to a lesser extent, the second formant.

5.4 Final Thoughts

We have demonstrated relative improvements in cross-channel SID accuracy from 6% to about 60%. Recognizing the best previously-published rates using other compensation methods was near 27% [13], this technique, theoretically and experimentally, seems to hold great promise for eventually solving the channel mismatch problem.

5.5 Recommendations for Further Study

The results proved promising for a new avenue for channel mismatch compensation. Further use of ANN software, with MM-NNT and other types, along with different architectural schemes should be employed. Other features should be looked at which have reasonable cross-correlation which may assist in the transformation through ANNs. A logical, direct follow-on would be to test the transformed features in a speaker verification task. But this mapping technique may also be applicable to other fields of research.

VI. Appendix

6.1 Artificial Neural Networks Theory

6.1.1 Network Classification Theory. The simplest classification problem is the two-class problem with the use of the linear discriminant function, *i.e.* if $y(x) > 0$ then classify x as a member of class one; if less than zero classify as a member of class two. The general equation is [5]

$$y(x) = w^T x + w_0 \quad (6.1)$$

where w is a d -dimensional weight vector. The threshold or offset, w_0 , is often referred to as the bias. Equation 6.1 corresponds to a hyperplane of dimensionality $(d - 1)$. If we set the offset to zero and set two points, x_1 and x_2 , on the hyperplane boundary where $y(x) = 0$, equation 6.1 becomes

$$w^T (x_2 - x_1) = 0. \quad (6.2)$$

Therefore, the weight vector is geometrically normal to any x vector contained in the hyperplane. As mentioned, the bias determines the offset position of the hyperplane. Hence, a linear discriminant function can be represented by Figure 6.1, where the input x_0 is permanently set to unity. This is often referred to as a single-layer perceptron.

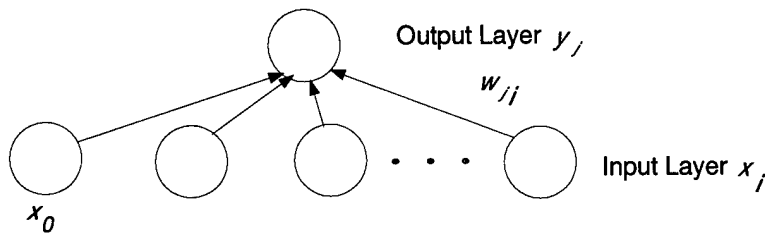


Figure 6.1 Simple Artificial Neural Network

6.1.2 Multi-class ANN. An easy extension to multiple classes is performed if each class, class k , has its own discriminant function. The hyperplane decision boundary

is

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0. \quad (6.3)$$

The network from Figure 6.1 has been changed for multiple classes, as demonstrated in Figure 6.2. Each connection has an associated weight with it. The network outputs can now be expressed by

$$y_k(x) = w_k^T x + w_{k0} \quad (6.4)$$

or

$$y_k(x) = \sum_{i=1}^d w_{ki} x_i + w_{k0}, \quad (6.5)$$

if the function associated with output nodes is merely linear with unity slopes. Otherwise,

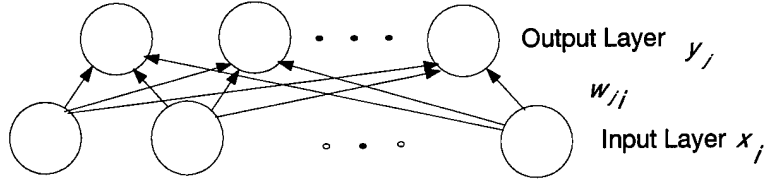


Figure 6.2 Single-Layer, Multiple Output Network

these activation functions may perform a nonlinear distortion to this sum to obtain the desired outputs.

6.1.3 Multilayer Perceptron Theory. We now extend these single-layer networks to multiple-layered networks, where the outputs of the first layer become the inputs of the second. They are called hidden layer nodes. Refer to Figure 6.3.

They are obtained, as before, by a linear combination of d weighted inputs, along with a bias term.

$$a_j(x) = \sum_{i=1}^d w_{ji}^{(1)} x_i + w_{j0}^{(1)} x_0 = \sum_{i=0}^d w_{ji}^{(1)} x_i \quad (6.6)$$

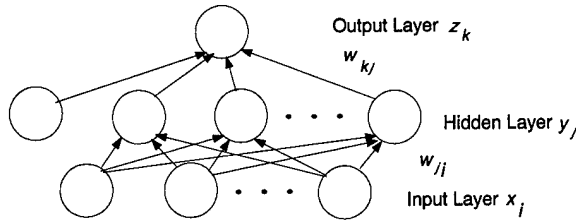


Figure 6.3 Multilayer Perceptron

where $w_{ji}^{(1)}$ denotes a first layer weight from input node i to hidden node j . Combining the fact a is the input the activation functions operate on and the existence of layers of hidden weights, the output of the k nodes are

$$y_k(x) = g^{(2)} \left(\sum_{j=0}^M w_{kj}^{(2)} g^{(1)} \left(\sum_{i=0}^d w_{ji}^{(1)} x_i \right) \right) \quad (6.7)$$

with the numbers in parentheses relating to the assigned layer affiliation. The concept can be extended to multiple layers of weights.

6.1.3.1 Nonlinear Activation Functions. Now consider a function, $g(w, x)$, called an activation function, which operates on the weighted sum. A popular nonlinear function is the logistic sigmoid, $(1 + e^x)^{-1}$; another is the hyperbolic tangent. Since the updating of the weights for convergence is based on the partial derivatives of the error function, the logistical sigmoid function is useful since its derivative is merely a function of itself, $g'(x) = g(x) * [1 - g(x)]$. This leads to easier calculations and weight updates. Although linear functions are used, the logistic sigmoid and, a sibling, the hyperbolic tangent, are good for dealing with nonlinearities. This fact is evident by the Exclusive-OR data separation problem example in [2]. [1] The problem was solved more readily with a sigmoid-sigmoid or tanh-tanh combination for the activation function of the first and second layer of weights than with a combination which included one layer of linear functions.

6.1.4 ANN parameters. Often with classification tools, training must occur, and, therefore, a measurement for validity must be used. One often used is the Sum-of-Squared

Errors (SSE), [2].

$$E = \frac{1}{2} \sum_{k=1}^K (d_k - z_k)^2 \quad (6.8)$$

where d_k is the desired output and z_k is the actual output. As adaptation of the weight for optimum performance is desired, an update rule for them must be established with relation to their impact on the SSE. This rule is [2],

$$W_{h+} = W_{h-} - \eta \frac{\delta E}{\delta W_{h-}} \quad (6.9)$$

which is equivalent to

$$\Delta W_h = -\eta \frac{\delta E}{\delta W_{h-}} \quad (6.10)$$

all representing vector notation, where W_+ is the updated set of weights, W_- is the old set of weights, h is the perceptron layer, and η is learning rate [2]. This formula uses gradient descent, which it seeks out the minimum SSE by use of the partial derivative of the error. An adaptive learning rate is best; for example, an η which is inversely proportional to the weight index number. This often reduces oscillations in SSE plot, since smaller corrections as proceeding towards convergence is desired. Another option is to tie η to the difference between the current output and actual target value.

In addition to an aeta term, a momentum term may be used to avoid local minimums in error, and thereby find the total minimum error. Momentum, which is proportional to the change in the weight values resulting from the last update, often speeds up convergence [2]. With momentum, the weight update equation becomes

$$\Delta W_h = -\eta \frac{\delta E}{\delta W_{h-}} + \mu \Delta W_h \quad (6.11)$$

6.1.5 Backpropagation. Error backpropagation is the process where the partial derivatives of the ANN output errors with regards to the second layer of weights is passed back to evaluate the partial derivatives of the outputs of each hidden layer node with

regards to the first layer of weights. It is due to backpropagation that updating weights become more efficient [1]. Again realizing the output of the activation function is

$$z_j = g(a_j) = g\left(\sum_i w_{ji} z_i\right) \quad (6.12)$$

the drive is to minimize the error in relation to the weights; the weight is connecting nodes i to j , and ϑ_j refers to errors of the hidden units relating to inputs. The errors of the network output units are the same, merely substituting k for j as layer indicator.

As the SSE metric demonstrates, the error of an ANN is a differentiable function of output variables. For backpropagation, the error relationship of network weights to be minimized is found by combining equations.

$$\frac{\delta E^n}{\delta w_{ji}} = \frac{\delta E^n}{\delta a_j} \frac{\delta a_j}{\delta w_{ji}} = \vartheta_j z_j \quad (6.13)$$

From the chain rule, the input error relationship for the output and hidden nodes is, respectively,

$$\vartheta_k = \frac{\delta E^n}{\delta a_k} = g'(a_k) \frac{\delta E^n}{\delta y_k} \quad (6.14)$$

and

$$\vartheta_j = \frac{\delta E^n}{\delta a_j} = \sum_k \frac{\delta E^n}{\delta a_k} \frac{\delta a_k}{\delta a_j}. \quad (6.15)$$

Note in the latter equation, the sum encompasses all the outputs k which is connected to j . Note also this equation demonstrates variations in a_j , the weighted sum of network inputs impact the a_k variables, the weighted sum of the upper layer inputs. Combining equations, the hidden unit input errors can be calculated by the backward propagating the input errors from the output nodes.

$$\vartheta_j = g'(a_j) \sum_k w_{kj} \vartheta_k \quad (6.16)$$

6.2 Gaussian Mixture Model Classification Theory

6.2.1 Semiparametric Classification Theory. The probability density function of mixture models is formed from a linear combination of basis functions.

$$p(x) = \sum_{j=1}^M p(x | j) P(j) \quad (6.17)$$

To demonstrate the use of Gaussian Mixture Models (GMMs) and the optimization algorithm, the connect with posterior probabilities is investigated via Bayes' theorem [1]

$$P(j | x) = \frac{p(x | j) P(j)}{p(x)} \quad (6.18)$$

The value of $P(j | x)$ represents the probability that component j was responsible for creating the data point x . The log-likelihood equation, a type of error function is given as [1]

$$E = -\ln \mathcal{L} = -\sum_{n=1}^N \ln p(x_n) = -\sum_{n=1}^N \ln \left\{ \sum_{j=1}^M p(x_n | j) P(j) \right\} \quad (6.19)$$

The error is minimized by maximizing the likelihood score, *i.e.* the Maximum A posteriori Probability (MAP).

6.2.2 Expectation-Maximization Theory for Solving the Mixture Model Parameters.

Given the adjustable parameters of this formula are the mean, variance and prior probability, the desire is to minimize this error function in relation to these parameters. Partially differentiating E with respect to each parameter, we get

$$\frac{\delta E}{\delta \mu_j} = \sum_{n=1}^N P(j | x_n) \frac{(\mu_j - x_n)}{\sigma_j^2}, \quad (6.20)$$

$$\frac{\delta E}{\delta \sigma_j} = \sum_{n=1}^N P(j | x_n) \left\{ \frac{d}{\sigma_j} - \frac{\|x_n - \mu_j\|^2}{\sigma_j^3} \right\}, \quad (6.21)$$

and

$$\frac{\delta E}{\delta \gamma_j} = \sum_{n=1}^N P(j | x_n) - P(j), \quad (6.22)$$

where

$$P(j) = \frac{\exp(\gamma_j)}{\sum_{k=1}^M \exp(\gamma_k)}. \quad (6.23)$$

Setting the derivatives equal to zero, and solving yield the highly nonlinear trinity of equations

$$\hat{\mu}_j = \frac{\sum_{n=1}^N P(j | x_n) x_n}{\sum_{n=1}^N P(j | x_n)}, \quad (6.24)$$

$$\hat{\sigma}_{j^2} = \frac{1}{d} \frac{\sum_{n=1}^N P(j | x_n) \|x_n - \hat{\mu}_j\|^2}{\sum_{n=1}^N P(j | x_n)}, \quad (6.25)$$

and

$$\hat{P}(j) = \frac{1}{N} \sum_{n=1}^N P(j | x_n). \quad (6.26)$$

which can only be solved in a practical sense by an iterative process. Baum-Welch [5] devised a way to better optimize through an iterative process by more carefully calculating updates to the iterations. The error function is manipulated into an updating process,

$$E_{new} - E_{old} = - \sum_n \ln \frac{p_{new}(x_n)}{p_{old}(x_n)}. \quad (6.27)$$

The resulting updatable parameter equations are

$$\mu_{j,new} = \frac{\sum_{n=1}^N P_{old}(j | x_n) x_n}{\sum_{n=1}^N P_{old}(j | x_n)}, \quad (6.28)$$

$$(\sigma_{j,new})^2 = \frac{1}{d} \frac{\sum_{n=1}^N P_{old}(j \mid x_n) \|x_n - \mu_{j,new}\|^2}{\sum_{n=1}^N P_{old}(j \mid x_n)}, \quad (6.29)$$

and

$$P(j)^{new} = \frac{1}{N} \sum_{n=1}^N P_{old}(j \mid x_n). \quad (6.30)$$

```

% fbf02htkgfbpu.m
% Edmund Fitzgerald 22SEP97, some from lines from Arb's files %fbf02htkfpuo2.m
% Takes default feature vectors (unvoiced) of the .fb files, combines 4-formants
% and 4-bandwidths into single feature vectors of 8 features and strips
% unvoiced records from them, via finding ESPS-imposed defaults for unvoiced;
% adds pitch also

clear all
close all
filenames=[];
indexunvs=[];
filecounter=1;

[fid1,message]=fopen(['/home/fuggles1/efitzger/toy/timit/test/dr5/total.gf0'],'r');
% listing of get_f0 files
diffleings=[];
done=0;
counter=0;
while ~done
    nextline=fgetl(fid1);

    if ~isstr(nextline) % at end of path list
        done=1;
    else % not at end of path list
        fborf0=fliplr(abs(nextline));
        if fborf0(1)==48 % .fb or .f0 file
            flag=0;
            flagnum=abs(flag);
        else

```

```

    flag=1;
    flagnum=abs(flag);
    end

% now Behead and copy back w/ new extension; but work on tmpfeatgfbpudr5.fbx
cd /home/fuggles1/efitzger/toy/timit/test/dr5;
eval(['!cp ',nextline,' /home/fuggles1/efitzger/toy/timit/test/dr5/tmpfeatgfbpudr5.fb2'
eval(['!bhd tmpfeatgfbpudr5.fb2 tmpfeatgfbpudr5.fbx']); %data to tmpfeatgfbpudr5.fbx
cd /home/fuggles1/efitzger/toy/timit/test/dr5

if flag==1
flag
% pull in beheaded file to put into new matrix
[fid2,message]=fopen('tmpfeatgfbpudr5.fbx','r','b'); %;
A=fread(fid2,inf,'float64');
fclose(fid2);
counter=counter+1;
Arows=size(A,1);
Brows=fix(Arows/8);
trash=Arows-(Brows*8);
A=A(1:Arows-trash,1);
B=reshape(A,Brows,8); % now the matrix is 8 (featgfbpu) by whatever
%FMBW=B';% the mat2fea needs x by 8 matrices
C=reshape(B,8,Brows);%
FMBWI=C';
% Now change order for easy missing features: f2f3bw2bw3f1bw1f4bw4
FMBWX=[];
FMBWX=FMBWI'; % 8 by x
FMBW=[FMBWX(2,:);FMBWX(6,:);FMBWX(3,:);FMBWX(7,:);FMBWX(1,:);FMBWX(5,:);FMBWX(4,:);...
...FMBWX(8,:)];
%F2B2F3B3F1B1F4B4

```

```

%%Not till FO incorporated FMBW=FMBW'; get rid of it use nnz to reshape again for...
... size
stringfile=abs(nextline);
for index=1:length(stringfile)
    if stringfile(index)==47 % s115
        if stringfile(index+1)==115 % /47 .46
            if (length(stringfile)==65) & (stringfile(index+4)==46) ;% .
                filename(1:3)=stringfile(index+1:index+3); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==66) & stringfile(index+5)==46 ;% .
                filename(1:4)=stringfile(index+1:index+4); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==67) & stringfile(index+6)==46 ;% .
                filename(1:5)=stringfile(index+1:index+5); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==68) stringfile(index+7)==46 ;% .
                filename(1:6)=stringfile(index+1:index+6); % sa1 obtained, what about sa3456
            end %elseif
        end % 115
    end %47
end % for index      varname=[filename flagnum];
filenameB=setstr(filename); %;
save /home/fuggles1/efitzger/dr5featgfbpu FMBW filenameB
end % flag==1

if flag==0
load /home/fuggles1/efitzger/dr5featgfbpu %takes in the 4fms and 4bws
adjust=0; % now obtain pitch and prob of voicing
eval(['!fea2mat -f FO ',nextline,' /home/fuggles1/efitzger/toy/timit/test/...
...tmpfeatgfbpudr51.mat']);
eval(['!fea2mat -f prob_voice ',nextline,' /home/fuggles1/efitzger/toy/timit/test/...
...tmpfeatgfbpudr52.mat']);
load /home/fuggles1/efitzger/toy/timit/test/tmpfeatgfbpudr51

```

```

load /home/fuggles1/efitzger/toy/timit/test/tmpfeatgfbpudr52
strayzero=0;
for nonzeroloop1=1:size(FMBW,1)
    for nonzeroloop2=1:size(FMBW,2)
        if FMBW(nonzeroloop1,nonzeroloop2)==0
            FMBW(nonzeroloop1,nonzeroloop2)=0.01 %eliminates stray zeros so no nnz ...
            ...confusion w/Pr<0.5
            strayzero=strayzero+1;
        end %if
    end % for nonzeroloop2
end % for nonzeroloop1

diff leng=length(F0)-max(size(FMBW))
F0=F0';
    if length(F0)>max(size(FMBW))
        F0=F0(1,1:max(size(FMBW))); %incase of a one frame discrepancy;v&unv regions ok
    end %since , 10ms overlap of 20ms frames
    if length(F0)<max(size(FMBW))
        FMBW=FMBW(:,1:length(F0)); %FMBW already made into matrix from first time through;
    end %Now incorporating pitch into matrix
FMBW0=[];lengF0=length(F0);
[max(size(FMBW)) length(F0) 1];
counterzero=0; tolerout=0;
FMBW0=[FMBW(1,:);FMBW(2,:);FMBW(3,:);FMBW(4,:);F0;FMBW(5,:);FMBW(6,:);FMBW(7,:);...
...FMBW(8,:)];
FMBW=FMBW0'; % Now ready to output
toler1=mean(FMBW(:,1))+2*std(FMBW(:,1));toler1n=mean(FMBW(:,1))-2*std(FMBW(:,1));
toler2=mean(FMBW(:,2))+2*std(FMBW(:,2));toler2n=mean(FMBW(:,2))-2*std(FMBW(:,2));
toler3=mean(FMBW(:,3))+2*std(FMBW(:,3));toler3n=mean(FMBW(:,3))-2*std(FMBW(:,3));
toler4=mean(FMBW(:,4))+2*std(FMBW(:,4));toler4n=mean(FMBW(:,4))-2*std(FMBW(:,4));

```

```

toler5=mean(FMBW(:,5))+2*std(FMBW(:,5));toler5n=mean(FMBW(:,5))-2*std(FMBW(:,5));
for indexrec=1:length(F0)
%NOT length(prob_voice) since F0 might have been corrected by one frame=FMBW leng
if prob_voice(indexrec)<0.5 %%%ANY of the following will zero out a feature vector
counterzero=counterzero+1;
FMBW(indexrec,:)=zeros(1,9);
elseif FMBW(indexrec,1)>toler1 | FMBW(indexrec,2)>toler2
tolerout=tolerout+1;
FMBW(indexrec,:)=zeros(1,9);
elseif FMBW(indexrec,2)==1000 & FMBW(indexrec,4)==1000
tolerout=tolerout+1;
FMBW(indexrec,:)=zeros(1,9);
elseif FMBW(indexrec,3)>toler3 | FMBW(indexrec,4)>toler4 | FMBW(indexrec,5)>toler5
tolerout=tolerout+1;
FMBW(indexrec,:)=zeros(1,9);
elseif FMBW(indexrec,1)<toler1n | FMBW(indexrec,2)<toler2n | ...
...FMBW(indexrec,3)<toler3n
tolerout=tolerout+1;
FMBW(indexrec,:)=zeros(1,9);
elseif FMBW(indexrec,4)<toler4n | FMBW(indexrec,5)<toler5n
tolerout=tolerout+1;
FMBW(indexrec,:)=zeros(1,9);
end %if
end % for indexrec
%sizeofNZideal=9*(length(F0)-counterzero);
FMBWZNZ=nonzeros(FMBW); %check for stray zero
[max(size(FMBWZNZ))/9 length(F0)-counterzero-tolerout counterzero 2] % check sizes

if flag~=2
FMBWP=reshape(FMBWZNZ,length(F0)-counterzero-tolerout,9); %New matrix

```

```

FMBWP=FMBWP';
% since .gf0 vs .fb or .f0 (2 char)
stringfile=abs(nextline);
stringfile=stringfile(1:(length(stringfile)-1));
setstr(stringfile)
for index=1:length(stringfile)
    if stringfile(index)==47 % s115
        if stringfile(index+1)==115 % /47 .46
            if (length(stringfile)==65) & (stringfile(index+4)==46) ;% .
                filename(1:3)=stringfile(index+1:index+3); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==66) & stringfile(index+5)==46 ;% .
                filename(1:4)=stringfile(index+1:index+4); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==67) & stringfile(index+6)==46 ;% .
                filename(1:5)=stringfile(index+1:index+5); % sa1 obtained, what about sa3456
            elseif (length(stringfile)==68) stringfile(index+7)==46 ;% .
                filename(1:6)=stringfile(index+1:index+6); % sa1 obtained, what about sa3456
            end %elseif
            %if (length(stringfile)==69) & (stringfile(index+7)==46) % .
            %other order did not work
        end % 115
    end %47
end % for index
varname=[filename flagnum];
filenameA=setstr(filename); %;
absfile=[39,abs(nextline(1:58)),abs('/featgfbpu/'),abs(filenameA),46,104,116,50,39];
filename=setstr(absfile)

filenameA % check
filenameB
if filenameA==filenameB

```

```

    % don't write out until F0 (pitch) taken in
    % Keep all variable in a dialect region, then cat those F0, FB same spkr
    eval(['!mkdir ',nextline(1:58),'/featgfbpu']));
    eval(['!chmod -fR g+w,g+x ',nextline(1:58),'/featgfbpu']));
    cd /home/hawkeye19/97d/efitzger/thesisum/SID/makefbwp/fbf02htk
    numfeat=min(size(FMBWP));
    FMBWMAT=FMBWP;
    FMBWT=FMBWP;
    FMBWQ=reshape(FMBWT,max(size(FMBWT))*min(size(FMBWT)),1);
    FMBWP=reshape(FMBWQ,max(size(FMBWP)),min(size(FMBWP)));

    % Developed empirical method ("Plaid-Shirt Method") to put into HTK 2.0 format
    eval(['w_error=whkparm(FMBWP,',filename,');'])
    % eval(['w_error=alwrthtkwav(FMBWP,',filename,',numfeat);'])
    if w_error== -1
        w_error
        flag=2
        end
        if flag~=2
            [max(size(FMBWI)) max(size(F0)) max(size(FMBWP))]
            FMBWT=FMBWP';
eval(['save ',nextline(1:58),'/featgfbpu/',filenameB,' FMBWMAT adjust strayzero ...
...tolerout;'])); %,counter is simple distinguishment
        end %if flag~=2
        end %if filename==filenameB
        elseif flag==2
        end % if flag~=2
        end % if F0
        diffleings=[diffleings;diffleing];
        clear FMBW FMBWP F0 FMBWX FMBWO filenameB filename FMsize %%%No confusion with other ...

```

```

...FMBW FO data
end %isstr

end %while loop
diffleings'
status=fclose(fid1);


% nnetpx921inov13e2.m 1nov97
% Captain Edmund Fitzgerald
% This is for nine input, one output mapping ANN generation using training data
% and mapping of the corresponding test features. Would be difficult to insure
% correct ANN with corresponding test utterances if separated to two programs.
%
close all
clear all
flops(0)
tic
hidcounter=0;ptrnmatch =0;ptrnmiss =0;ptstmatch =0;ptstmiss =0;
flopnum=[];T=[];P=[];totrand=75;
convergetry=0;fmiterx=0
[fid1,message]=fopen(['/home/fuggles1/efitzger/toy/ntimit/test/dr1/totalfbptnt.tst'],'r')
% list of test feature file paths

done=0;
tstcounter=1;
while ~done

```

```

spkrtemp=fgetl(fid1)

if ~isstr(spkrtemp) % at end of path list
    done=1;
else % not at end of path list

%strip off speaker name for variable attachment
ntimortim=(abs(spkrtemp));
    %for iter=1:length(ntimortim)
        if ntimortim(29)==110 % 'n' ,116=t
            flag=0; %;
            flagnum=abs(flag);
            spkr=spkrtemp(45:49) %if ntimit
            else
                flag=1;%;
                flagnum=abs(flag);
                spkr=spkrtemp(44:48) %if timit
            end
        if flag==0
            % Will bring in 2 tst sentences for conversion with the trained NNet

            if tstcounter==1
                spkrtemp1=spkrtemp; % hold first test sentence

            % Now generate nnet..only do once/spkr/feature
            % The training data is combined, not the test data,i.e. test individual utters
            eval(['load /home/fuggles1/efitzger/toy/ntimit/test/dri/','spkr,','FMBWPNtrn']);
            % loading NTIMIT training data, i.e. formants, bandwidths, and pitch (trn)

            FMBWPNS=FMBWPNS';

```

```

%FMBWPNs=[FMBWPNs(1,:);FMBWPNs(2,:);FMBWPNs(4,:)];
P=(1/10000).*FMBWPNs; % Need to divide data by constant to be input to...
    ...nonlinear activation functions.

eval(['load /home/fuggles1/efitzger/toy/timit/test/dr1/',spkr,'/FMBWPTtrn']);
% loading corresponding TIMIT data
FMBWPTs=FMBWPTs';
F1ts=FMBWPTs(1,:);F2ts=FMBWPTs(2,:);F3ts=FMBWPTs(3,:);
F4ts=FMBWPTs(4,:);F5ts=FMBWPTs(5,:);F6ts=FMBWPTs(6,:);
F7ts=FMBWPTs(7,:);F8ts=FMBWPTs(8,:);F9ts=FMBWPTs(9,:);
for fmitter=1:9 %need an end later

eval(['Fts=F',int2str(fmitter),'ts;']);
T=(1/10000).*Fts;
    F1='tansig';F2='logsig'; %activation functions
hidvctr=[45]; % Used 45 hidden layer nodes
maxcount=max(size(hidvctr));

% ANN parameters
for hidcounter=1:maxcount %use same randn dat in e xter fcn 10x,rtrn
    trnmsclsfy=0;tstmsclsfy=0;trnmatch=0;tstmatch=0;
    %note learnbpm.m is contained in trainbpx.m/tbpx3.m
    dsplyepochs=50; maxepochs=300; sse=0.01; lr=0.01;
    lrinc=1.05; lrdec=0.5; mo=0.90 ;maxerr=1.04;
    % defaults: dsplyepochs=25; maxepochs=100; sse=0.02; learnrate=0.01;
    %lrinc=1.05; lrdec=0.7; momentum=0.9;a)0.7,0.96; b)0.5,0.96 maxerr=1.04;

    tp=[dsplyepochs,maxepochs,sse,lr,lrinc,lrdec,mo,maxerr];
    hidden=hidvctr(hidcounter);

```

```

[W1,b1,W2,b2]=initff(P,hidden,F1,min(size(T)),F2);
[W1,b1,W2,b2,te,tr]=trainbpx(W1,b1,F1,W2,b2,F2,P,T,tp);
% ANN trained using training data

format bank

['dsplyepochs ' 'maxepochs ' 'sse ' 'lr ' 'lrinc ' 'lrdec ' 'mo ' 'maxerr ' 'hidden...
...-nodes']

%parameters=[dsplyepochs, maxepochs, sse, lr, lrinc, lrdec, mo, maxerr, F1, F2]
[tp, hidvctr(hidcounter)]
filenameex='pxnov13_drizde2'
eval(['save /home/hawkeye19/97d/efitzger/thesisum/SID/makefbwp/makegmm/gmmfpu/nnet...
.../netwtsnov13/pxnov13_drizde2',num2str(fmiter),spkr,num2str(hidden),...
...num2str(maxepochs),' W1 b1 W2 b2 te tr tp flopnum']);
%%move downend % of hidcounter loop
filenameex='pxnov13_drizde2'
%Even number times of bringing in files, skipped over making nnet..go to (.ht2) o/p...

spkrtemp % log file check
    eval(['load ',spkrtemp]);%combine tim and ntim separately, but insure same length
    FMBWPT=FMBWPT./10000; % TIMIT test formants bandwidths and pitch ;divide by const
    FMBWPN=FMBWPN./10000; %NTIMIT test formants bandwidths and pitch from NTIMIT...div
    p=FMBWPN;
t=simuff(p,W1,b1,F1,W2,b2,F2);% now map the test data using the trained ANN
eval(['Nop',int2str(fmiter),'=t;']);

meanNop1=10000.*mean(Nop1)
if meanNop1<100 % This is decision area; if true then test data transformed badly so
    % ...try the ANN generation again (using training data)
    % The following loop insures a convergence to reasonable SSE

```

```

fmiterx=fmiter;%%%
convergetry=convergetry+1

while fmiterx==fmiter %go through nnet generation until meanNop>100
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T=(1/10000).*Fts;
F1='tansig';F2='logsig';
hidvctra=[45];

%use same randn data in exter ANN generation fcn 10x,return
%note learnbpm.m is contained in trainbpx.m/tbpx3.m
dsplyepochs=50; maxepochs=300; sse=0.01; lr=0.01;
lrinc=1.05; lrdec=0.5; mo=0.90 ;maxerr=1.04;
% defaults: dsplyepochs=25; maxepochs=100; sse=0.02; learnrate=0.01;
%lrinc=1.05; lrdec=0.7; momentum=0.9;a)0.7,0.96; b)0.5,0.96 maxerr=1.04;

tpa=[dsplyepochs,maxepochs,sse,lr,lrinc,lrdec,mo,maxerr];
hiddena=hidvctra;

[W1,b1,W2,b2]=initff(P,hiddena,F1,min(size(T)),F2);
[W1,b1,W2,b2,te,tr]=trainbpx(W1,b1,F1,W2,b2,F2,P,T,tpa);
format bank
['dsplyepochs ' 'maxepochs ' 'sse ' 'lr ' 'lrinc ' 'lrdec ' 'mo ' 'maxerr ' 'hidden...
...-nodes'] %check progress in log file
%parameters=[dsplyepochs, maxepochs, sse, lr, lrinc, lrdec, mo, maxerr, F1, F2]
[tpa, hidvctra] %check in log file
filenamex='pxnov13_dr1zde2'
eval(['save /home/hawkeye19/97d/efitzger/thesisum/SID/makefbwp/makegmm/gmmfpu...
.../nnet/netwtsnov13/pxnov13_dr1zde2',num2str(fmiter),spkr,num2str(hidden),...

```

```

...num2str(maxepochs),' W1 b1 W2 b2 te tr tp flopnum']]);
%%%move downend % of hidcounter loop
filenameex='pxnov13_dr1zde2'
%Even number times of bringing in files, skipped over making nnet..go to .ht2 o/p...
%still have if flag==0
    spkrtemp
    eval(['load ',spkrtemp]);%combine tim and ntim separately, but insure same leng
    FMBWPT=FMBWPT./10000;
    FMBWPN=FMBWPN./10000;
    p=FMBWPN;
    t=simuff(p,W1,b1,F1,W2,b2,F2);% we just want #s!conf matrix
    eval(['Nop',int2str(fmiter),'=t;']);
    eval(['meanNopx=10000.*mean(Nop',int2str(fmiter),');']);
    if meanNopx<100 %junk data
        fmiterx=fmiter; %instead of 1
    else
        fmiterx=fmiter+1 %instead of 0
    end %if four lines up
end %while fmiterx=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% End of convergence-insurance loop

else
    fmiterx=fmiter+1
    convergetry=0
end %if 3lines up/mean<100

% Now can format and output

end %fmiter
end % of hidcounter loop

```

```
end % if tstcont=1
```

```
% Combinations of transformed and original NTIMIT features
```

```
fm1=[Nop1;Nop2;Nop3;Nop4;FMBWPN(5,:);Nop6;Nop7;Nop8;Nop9];
```

```
fm2=[Nop1;Nop2;Nop3;Nop4;Nop5;Nop6;Nop7;Nop8;Nop9];;%how get pitch in here...
```

```
...unchanged pitched
```

```
fm3=[FMBWPN(1,:);FMBWPN(2,:);Nop3;Nop4;Nop5;Nop6;Nop7;Nop8;Nop9];
```

```
fm4=[Nop1;Nop2;FMBWPN(3,:);FMBWPN(4,:);Nop5;Nop6;Nop7;Nop8;Nop9];
```

```
fm5=[Nop1;Nop2;Nop3;Nop4;Nop5;Nop6;Nop7;FMBWPN(8,:);FMBWPN(9,:)];
```

```
fm6=[Nop1;Nop2;Nop3;Nop4;Nop5;FMBWPN(6,:);FMBWPN(7,:);Nop8;Nop9];
```

```
fm7=[FMBWPN(1,:);FMBWPN(2,:);Nop3;Nop4;FMBWPN(5,:);Nop6;Nop7;Nop8;Nop9];
```

```
fm8=[Nop1;Nop2;FMBWPN(3,:);FMBWPN(4,:);FMBWPN(5,:);Nop6;Nop7;Nop8;Nop9];
```

```
%%%keyboard
```

```
FMBWPNtoT1=10000.*fm1;
```

```
FMBWPNtoT2=10000.*fm2;
```

```
FMBWPNtoT3=10000.*fm3;
```

```
FMBWPNtoT4=10000.*fm4;
```

```
FMBWPNtoT5=10000.*fm5;
```

```
FMBWPNtoT6=10000.*fm6;
```

```
FMBWPNtoT7=10000.*fm7;
```

```
FMBWPNtoT8=10000.*fm8;
```

```
%Want NTIMIT tst feature vectors to appear to be TIMIT tst fvectors for classification...
```

```
.../GMMs
```

```
%Need to put into .ht2 format files
```

```
stringfile=abs(spkrtmp);
```

```
stringfile=stringfile(1:(length(stringfile)));
```

```
setstr(stringfile)
```

```
for index=1:length(stringfile)
```

```
if stringfile(index)==47 % s115
```

```

if stringfile(index+1)==115 % /47 .46
    if (length(stringfile)==82) & (stringfile(index+9)==46) ;% .
        filename=stringfile(index+6:index+8); % sa1 obtained, what about sa3456
    elseif (length(stringfile)==83) & stringfile(index+10)==46 ;% .
        filename=stringfile(index+6:index+9); % sa1 obtained, what about sa3456
    elseif (length(stringfile)==84) & stringfile(index+11)==46 ;% .
        filename=stringfile(index+6:index+10); % sa1 obtained, what about sa3456
    elseif (length(stringfile)==85) & stringfile(index+12)==46 ;% .
        filename=stringfile(index+6:index+11); % sa1 obtained, what about sa3456
    end %elseif
    %if (length(stringfile)==69) & (stringfile(index+7)==46) % .
    %other order did not work
end % 115
end %47
end % for index      varname=[filename flagnum];
basic=setstr(filename)
absfile1=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi1/'),abs(filename),46,104,116,50,39];
absfile2=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi2/'),abs(filename),46,104,116,50,39];
absfile3=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi3/'),abs(filename),46,104,116,50,39];
absfile4=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi4/'),abs(filename),46,104,116,50,39];
absfile5=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi5/'),abs(filename),46,104,116,50,39];
absfile6=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi6/'),abs(filename),46,104,116,50,39];
absfile7=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi7/'),abs(filename),46,104,116,50,39];
absfile8=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi8/'),abs(filename),46,104,116,50,39];
%absfile9=[39,abs(spkrtemp(1:59)),abs('/fesgfbpi9/'),abs(filename),46,104,116,50,39];
filename1=setstr(absfile1) ; % A TICK (tick ') MARK IS ABS 39
filename2=setstr(absfile2);
filename3=setstr(absfile3);
filename4=setstr(absfile4);
filename5=setstr(absfile5) ; % A TICK (tick ') MARK IS ABS 39

```

```

filename6=setstr(absfile6);
filename7=setstr(absfile7);
filename8=setstr(absfile8);
    %filename9=setstr(absfile9);
%eval(['!mkdir ',filename1(1:71)]);
%eval(['!chmod -fR g+w,g+x ',filename1(1:71)]);
eval(['!mkdir ',filename1(2:60),'fesgfbpi1/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi1/']);
eval(['!mkdir ',filename2(2:60),'fesgfbpi2/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi2/']);
    eval(['!mkdir ',filename3(2:60),'fesgfbpi3/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi3/']);
    eval(['!mkdir ',filename3(2:60),'fesgfbpi4/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi4/']);
    eval(['!mkdir ',filename3(2:60),'fesgfbpi5/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi5/']);
    eval(['!mkdir ',filename1(2:60),'fesgfbpi6/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi6/']);
    eval(['!mkdir ',filename2(2:60),'fesgfbpi7/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi7/']);
    eval(['!mkdir ',filename3(2:60),'fesgfbpi8/']);
eval(['!chmod -fR g+w,g+x ',filename1(2:60),'fesgfbpi8/']);
filename;
for saveiter=1:8 % 5NOV
    if saveiter==1
        FMBWP=FMBWPNtoT1;
        FMBWT=FMBWPNtoT1;
        filename=filename1;
    elseif saveiter==2
        FMBWP=FMBWPNtoT2;

```

```

FMBWT=FMBWPNtoT2;
filename=filename2;
elseif saveiter==3
FMBWP=FMBWPNtoT3;
FMBWT=FMBWPNtoT3;
filename=filename3;
elseif saveiter==4
FMBWP=FMBWPNtoT4;
FMBWT=FMBWPNtoT4;
filename=filename4;
elseif saveiter==5
FMBWP=FMBWPNtoT5;
FMBWT=FMBWPNtoT5;
filename=filename5;
elseif saveiter==6
FMBWP=FMBWPNtoT6;
FMBWT=FMBWPNtoT6;
filename=filename6;
elseif saveiter==7
FMBWP=FMBWPNtoT7;
FMBWT=FMBWPNtoT7;
filename=filename7;
elseif saveiter==8
FMBWP=FMBWPNtoT8;
FMBWT=FMBWPNtoT8;
filename=filename8;
end %if 48lines up

FMBWQ=reshape(FMBWT,max(size(FMBWT))*min(size(FMBWT)),1);
FMBWP=reshape(FMBWQ,max(size(FMBWP)),min(size(FMBWP)));

```

```

cd /home/hawkeye19/97d/efitzger/thesisum/SID/makefbwp/xbf02htk
    eval(['w_error=whkparm(FMBWP,',filename,');'])
flopnum=flops;
cd /home/hawkeye19/97d/efitzger/thesisum/SID/makefbwp/makegmm/gmmfpu/nnet/netwtsnov13
eval(['save pxnov13_dr1zdwts2',spkr,num2str(maxepochs),F3,num2str(hidden),' ']);
% dontoverwrite
tstcounter=tstcounter+1;
if tstcounter==3
    tstcounter=1;
end %2lines above...reset
end %tstcounter
end %extra in here for
end
end
end
end %if isstr
end %while
flops
toc
fclose(fid1)

```

Bibliography

1. Christopher M. Bishop, *Neural Networks for Pattern Recognition* , Clarendon Press, Oxford, UK, 1996.
2. Steven K. Rogers, *Introduction to Perceptrons: Advanced Topics in Neural Networks* , Air Force Institute of Technology (AFIT) Document, WPAFB, OH, 1996.
3. Douglas A. Reynolds et al., "The Effects of Telephone Transmission Degradation on Speaker Recognition Performance", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, , 1995, pp. 329-332.
4. J.R. Deller, Jr., J.G. Proakis, J.H.L. Hansen, *Discrete-Time Processing of Speech Signals* , Macmillan Publishing Company, Englewood Cliffs, NJ, 1993.
5. Thomas W. Parsons, *Voice and Speech Processing* , McGraw-Hill Book Company, New York City, NY, 1987.
6. Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition* , Prentice Hall, Englewood Cliffs, NJ, 1993.
7. Douglas A. Reynolds, *A Guassian Mixture Modeling Approach to Text-Independent Speaker Identification* , PhD Thesis, Georgia Institute of Technology, Atlanta, GA, September 1992.
8. Marvin R. Sambur, *Selection of Acoustic Features for Speaker Identification* , IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP 23, pp. 176-182, 1975.
9. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis* , John Wiley and Sons, New York, NY, 1973.
10. B.S. Atal, *Effectiveness of Linear Prediction Characteristics of the Speech Wave for Automatic Speaker Identification and Verification* , Journal of the Acoustical Society of America, JASA 55:6, pp.1304-1312, June 1974.
11. Richard P. Lippmann and Beth A. Carlson, *Speech Recognition by Humans and Machines under Conditions with Severe Channel Variability and Noise* , Proceedings of The Society of Photonics, Imaging, and Electronics (SPIE), Volume 3077, pp. 46-57, April 1997.
12. Eric J. Zeek, *Speaker Recognition by Hidden Markov Models and Neural Networks* , Masters Thesis, AFIT, December 1996.
13. R. Brian Reid, *Speaker Identification and Verification by Gaussian Mixture Models* , Masters Thesis, Air Force Institute of Technology, December 1997.
14. Bruce G. Secrest and George R. Doddington, *An Integrated Pitch Tracking Algorithm for Speech Systems* , International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1993.
15. Douglas A. Reynolds, *Speaker Identification and Verification Using Gaussian Speaker Models* , Speech Communication:17, pp 99-108, 1995.

16. ESPS Programs Version 5.1, Entropic Research Laboratories, 1995.
17. Steve Young, et al., The HTK Book: For Version 2.0, Entropic Research Laboratories, Washington, DC, 1995.
18. A. Lapedes and R. Farber, *How Neural Nets Work*, Neural Information Processing Systems, pp 442-456, 1988.
19. Howard Demuth and Mark Beale, *Neural Network Toolbox User's Guide* The Mathworks, Inc., March 1996.
20. Richard Mammone, Xiaoyu Zhang, and Ravi Ramachandran, *Robust Speaker Recognition: A Feature-Based Approach*, IEEE Signal Processing Magazine, pp. 58-71, September 1996.
21. Richard J. Mammone and Khaled T. Assaleh, *New LP-Derived Features for Speaker Identification*, IEEE Transactions on Speech and Audio Processing, pp. 630-644, October 1994.
22. Richard J. Mammone and Devang K. Naik, *Channel Normalization Using Pole-Filtered Cepstral Mean Subtraction*, Proceedings of The Society of Photonics, Imaging, and Electronics (SPIE), pp. 99-110, Volume 2277, 1996.
23. K. Funahashi, *On the Appropriate Realization of Continuous Mappings by Neural Networks*, Neural Networks, Volume 2, pp. 183-192, 1989.
24. R. Hecht-Nielsen, *Theory of the Back-Propagation Neural Network*, Proceedings of the IEEE International Joint Conference on Neural Networks, Volume 1, pp. 593-605, 1989.
25. G. Cybenko, *Approximation by Superpositions of a Sigmoidal Function.*, Mathematics of Control, Signals, and Systems, Volume 2, pp.304-314, 1989.
26. K. Hornik, M. Stinchcombe, and H. White, *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, Volume 2, pp. 359-366, 1989.
27. K. Hornik, *Approximation Capabilities of Multilayer Feedforward Networks*, Neural Networks, Volume 4, pp. 251-257, 1991.
28. Y. Ito, *Representation of Functions by Superpositions of a Step or Sigmoid Function and Their Applications to Neural Network Theory*, Neural Networks, Volume 4 (3), pp. 385-394, 1991.
29. V.Y. Kreinovich, *Arbitrary Nonlinearity is Sufficient to Represent All Functions by Neural Networks: A Theorem*, Neural Networks, Volume 4, pp. 381-383, 1991.
30. M. Stinchcombe and H. White, *Universal Approximation Using Feedforward Networks with Non-Sigmoid Hidden Layer Activation Functions*, IEEE Proceedings of the International Joint Conference on Neural Networks, Volume 1, pp. 613-618, 1989.
31. N.E. Cotter, *The Stone-Weierstrass Theorem and Its Application to Neural Networks*, IEEE Transactions on Neural Networks 1, pp.290-295, 1990.

Vita

Captain Edmund Albert Fitzgerald was born on April 24, 1968 in the Holmesburg section of Philadelphia, Pennsylvania. After graduating high school in June of 1986, he entered Drexel University in downtown Philadelphia. On June 16, 1991 he received the Bachelor of Science in Electrical Engineering degree and was commissioned a second lieutenant in the United States Air Force. He worked as a civilian government-service (GS) engineer at the Naval Air Development Center in Warminster, PA until his active-duty assignment began on May 15, 1992. His first Air Force assignment was at the Air Force Primary Standards Lab (AFPSL) at the Aerospace Guidance and Metrology Center, Newark AFB, Ohio. His first position was a calibration engineer responsible for calibration programming and technical support in DC/Low Frequency measurement area for the AFPSL and the 168 Precision Measurement Equipment Laboratories (PMELs) around the Air Force. During the Winter of 1994, then First Lieutenant Fitzgerald completed the six-week RF/Microwave Calibration Course at the Air Force PMEL School at Lowry AFB, Colorado with perfect scores. Upon his return to Newark AFB, he was appointed RF/Microwave calibration engineer, specializing in the procurement, testing, and technical support of microwave, TACAN, IFF, VOR, ILS, and MLS test equipment. This position culminated in the leading of a proposal evaluation team for a successful 13 million dollar procurement. In May of 1996, then Captain Fitzgerald transferred to AFIT at Wright-Patterson AFB, Ohio to pursue the Masters of Science degree in Electrical Engineering. At AFIT, he was elected president of Eta Kappa Nu honor society. Upon graduating, Captain Fitzgerald will transfer to the National Air Intelligence Center at Wright-Patterson AFB as an Electrical Intelligence Systems Engineer in the Directorate of Data Exploitation.

Permanent address: 2869 Walnut Hill Street
Philadelphia, PA 19152-2135

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 1998		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE CHANNEL-MISMATCH COMPENSATION IN SPEAKER IDENTIFICATION: FEATURE SELECTION AND ADAPTATION WITH ARTIFICIAL NEURAL NETWORKS				5. FUNDING NUMBERS
6. AUTHOR(S) Edmund Albert Fitzgerald				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/98M-02
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Captain John Colombi, Ph.D. National Security Agency, R221 9800 Savage Road STE 6516 Fort Meade MD 20755-6516				10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Distribution Unlimited				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words) We develop and present results of an artificial neural network (ANN) based compensation technique for mismatched classifier training and testing conditions in speaker identification (SID). One ANN per feature per speaker is trained to perform a mapping of that feature from a corrupted condition to an undistorted condition. Therefore, a classifier trained under one condition may be used to classify data collected under a different condition. Speech utterances from 168 speakers, collected in a studio, and also re-recorded after transmission over telephone networks, are used for developing and testing the method. Peak formant resonant frequencies, their bandwidths, and pitch are used as features. These features from the studio speech are used to train Gaussian Mixture Model classifiers. Portions of the studio and telephone speech are used to train the compensation ANNs. In mismatched train and test conditions, features from telephone speech are modified by the trained ANNs and applied to the GMMs trained with features from studio speech. Without compensation, SID accuracy is 6%. The compensation method developed in this work provides mismatch SID accuracy of 58.3%. Previous research on the same data with the commonly-used Mel-Frequency Cepstral Coefficients as features and a typical compensation method of Cepstral Mean Subtraction with Band-Limiting gives SID accuracy of 27.4% with the same type of classifiers.				
14. SUBJECT TERMS speaker identification, channel compensation, channel mismatch, Gaussian Mixture Models, artificial neural networks, multi-layer perceptron, TIMIT, NTIMIT				15. NUMBER OF PAGES 70
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED 20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements*.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available
(e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es).
Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities.

NASA - See Handbook NHB 2200.2.

NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.

DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
Leave blank.

NASA - Leave blank.

NTIS -

Block 13. Abstract. Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (*NTIS only*).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.