STOP CONSONANT CLASSIFICTION USING RECURRANT NEURAL NETWORKS

NSF Summer Undergraduate Fellowship in Sensor Technologies
David Auerbach (physics), Swarthmore College
Advisors: Ahmed M. Abdelatty Ali, Dr. Jan Van der Spiegel, Dr. Paul Mueller

ABSTRACT

This paper describes the use of recurrent neural networks for phoneme recognition. Spectral, Bark scaled, and cepstral representations for input to the networks are discussed, and an additional input based on algorithmically defined features is described that can also be used as input for phoneme recognition. Neural networks with recurrent hidden layers of various sizes are trained to determine, using the various input representations, whether a stop consonant is voiced or unvoiced, and whether the stop consonant is labial, alveolar, or palatal. For voicing detection the peak accuracy was 75% of the phonemes not used to train the network identified correctly, and for placement of articulation, the peak accuracy was 78.5% of the testing set identified correctly. Using the algorithmically defined features and a three-layer feedforward network, an average accuracy of 80% for voicing and 78% for placement of articulation. Implications of these results and further research needed are discussed.

Table Of Contents

1. INTRODUCTION
2. METHODOLOGY
    - 2.1 Problem Description
    - 2.2 Input Format
        - 2.2.1 Spectrograms
        - 2.2.2 Bark Scaling
        - 2.2.3 Cepstral Representation
    - 2.3 Feature Extraction
3. NETWORKS
    - 3.1 Network Architecture
    - 3.2 Network Simulators
    - 3.3 Network Input
    - 3.4 Network Output
4. EXPERIMENTS AND RESULTS
    - 4.1 Voicing Detection
    - 4.2 Placement of Articulation Detection
    - 4.3 Feature Analysis
5. DISCUSSIONS AND CONCLUSIONS
6. ACKNOWLEDGMENTS
7. REFERENCES

1. INTRODUCTION

Even though computers get more and more powerful every year, they still have some of the limitations inherent in their design. Programmers describe one of these limitations with the acronym GIGO, which stands for "Garbage In, Garbage Out." If the instructions and the input to a computer do not make sense, then the output will not make any sense either.

One type of input which computers cannot use because it is too unclear is speech. Human speech is too complex and variable to be used as direct input to a computer. The pitch of our voices changes radically between speakers, we do not space out our words when speaking continuously, and there are huge numbers of different languages and accents. These problems and many others pose an enormous challenge to programmers trying to allow computers to accept speech as input. Yet the human brain successfully manages to interpret the wide variety of dialects, pitches, and speeds of speech with ease, and can learn several different languages without many problems.

The human brain is a computer too, one that is much better suited to the wide varieties of inputs that we encounter. It performs its functions by using an enormous number of interconnected neurons, and it seems to deal easily with difficult tasks such as speech recognition. It is our hope that by using neural networks, which in a simplistic way model how the human brain functions, we will be able to get a computer to successfully recognize speech. More specifically, we hope to be able to get a neural network to efficiently recognize phonemes, which would greatly simplify the further problem of word recognition.

Several programs, both hardware and software based, are currently used for speech recognition, but all suffer from one of two flaws. They are either not speaker independent, in that they need separate training to understand each person using the system, or they have very limited vocabularies. Such programs include the commercially available Dragon Naturally Speaking software, which needs to be trained on each individual user, and the software that allows telephone customers to speak their menu selections instead pressing a button on their phone, which recognizes only spoken numbers.

The scope of this research was limited to the recognition of one class of phonemes, the stop consonants. Nor did it attempt to separate out these phonemes from continuous speech. Instead it used pre-segmented phonemes from the TIMIT database as its inputs.

2. METHODOLOGY

2.1 Problem Description

The neural networks we designed were intended to distinguish the six stop consonants. Our goal was to have a neural network take as input one of these consonants in some form and output which of the six phonemes had been fed through the system. The stop consonants are distinguished by the fact that they are produced by a portion of the vocal tract momentarily closing off the flow of air from the lungs, and then releasing the built up pressure. In the palatal consonants /k/ and /g/, the back of the tongue contacts the soft palate, closing off the flow of air momentarily. In the alveolar consonants /t/ and /d/, the front of the tongue contacts the roof of the mouth directly behind the teeth before releasing. The labial stops /p/ and /b/ are produced by the lips closing off the flow of air and then releasing it. Thus one way to categorize the stop consonants is by the location of their production.

The other way to classify the stop consonants is to determine whether they are voiced or unvoiced. For the voiced stop consonants, /b/, /d/, and /g/, the vocal chords vibrate as the air flows over them. The unvoiced stops, /p/, /t/, and /k/, are produced in the same manner as the voiced stops at the same location, but without the vocal chords vibrating [Edwards, 1992].

2.2 Input Format

Several different representations of speech can be used for speech recognition. A computer records speech in a format that consists simply of sound pressure levels sampled at a certain frequency. For the TIMIT database this sampling frequency is 16000 Hz. While this format is useful for sound reproduction it is less useful for speech analysis, as it consists only of a long series of numbers in a row (Figure 1).

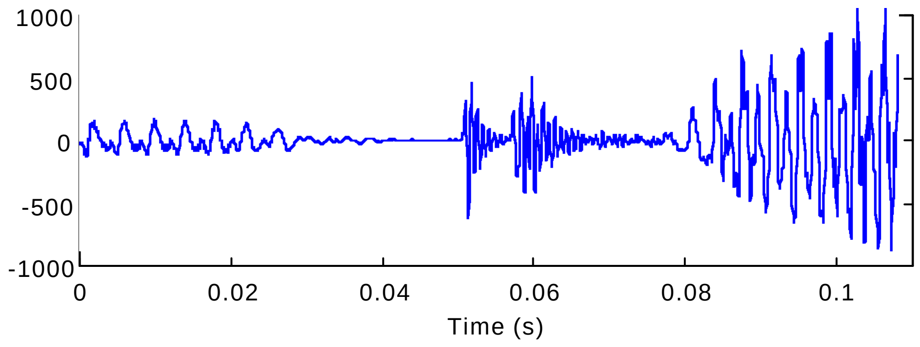

Figure 1: A sampled recording of the phoneme /g/

2.2.1 Spectrograms

One much more common way of representing sounds is to display them in the form of a spectrogram (Figure 2). This is done by taking the Fourier transform of the sound in small segments, and using the output to describe the intensity of each component frequency at each segment in time. Similarly, the cochlea of the human ear breaks down sound signals into activation levels at separate frequencies. However, in a spectrogram the frequency increases linearly, so a large number of frequencies are needed to cover the range needed for speech. The spectrogram shown in Figure 2 has 129 channels, which is a large amount of data for the network to be handling at each time step.

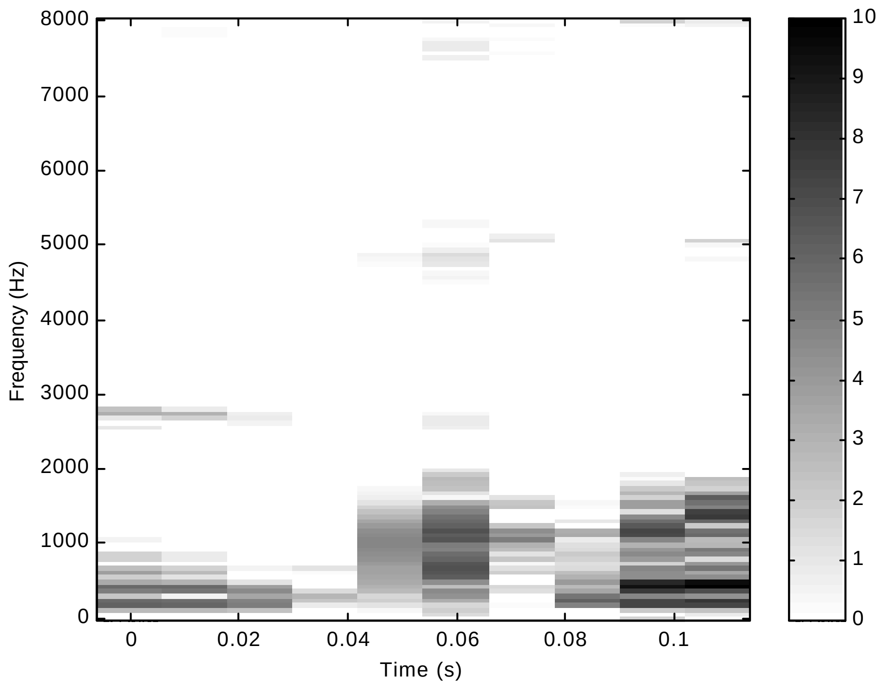

Figure 2: A spectrogram representation of the same /g/. Higher intensities are dark.

2.2.2 Bark Scaling

Another, more biologically realistic way to represent sound for speech recognition is to use Bark scaling [Zwicker, 1961]. This is similar to a simple spectrogram, in that it describes the power at certain frequencies over time, but instead of individual frequencies, it describes power in certain bands of frequencies. These bands are defined by the properties of the human cochlea; they are narrow bands where the cochlea has higher frequency resolution (the low frequencies) and are wider in the higher frequencies where the cochlea has lower resolution. This allows us to reduce 129 bands of information to just 20, without losing much of the information important for speech recognition. The bands used for this project can be seen in Figure 3.

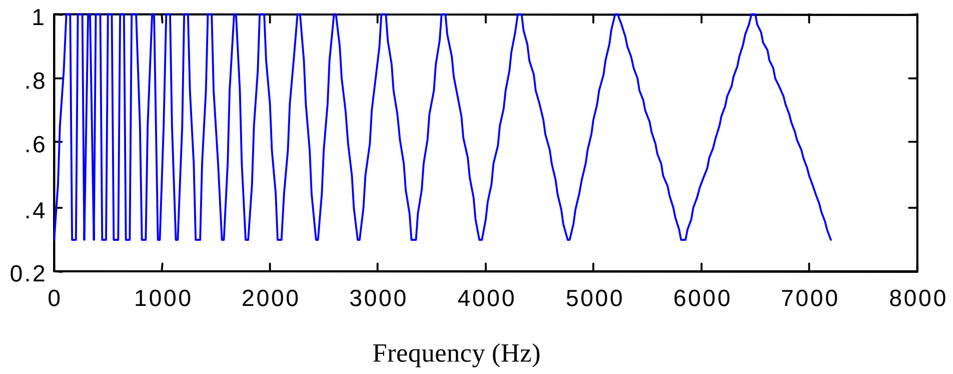

Figure 3: The Bark bands used.

2.2.3 Cepstral Representation

A third representation useful for speech recognition, which was also used for this project, is the cepstral representation of speech [Schafer, 1975]. This representation comes from modeling human speech as a carrier signal, produced by the vocal chords, and an envelope signal, produced by the mouth, the nose, and the rest of the vocal apparatus. This envelope contains most of the speech information; the carrier signal is simply a sign wave or a set of sign waves if the speech is voiced or a noise signal if the speech is unvoiced. The envelope signal can be separated out from the carrier signal by taking the Fourier transform of the signal, then taking the log of the result, and then taking an inverse Fourier transform. The envelope signal can then be analyzed without the extra information about pitch and tone that is in the carrier signal. This technique is one of the ways that researchers have attempted to create speaker independent software. For more information about the mathematics behind cepstral analysis, see Schafer,1975.

2.3 Feature Extraction

In addition to using pre-processed sound as input to the network, we also used a set of features useful for identifying phonemes. These features, developed by Ali et al. [Ali et al., 1999; Ali et al, 1998], are determined algorithmically from processed spectrograms for use in a phoneme recognition code. While this code has a very good success rate, it is purely algorithmic and uses threshold values of these features to determine phoneme classification. We hoped that by using these features in addition to direct speech as input to a neural network, we could achieve higher rates of recognition than would be possible with either input alone.

Some of these features look at temporal aspects of the phoneme, including the length of certain portions of each phoneme. This is important information for the network, because the recurrent neural networks that we used are not very successful at analyzing signals with respect to lengths of time. Other features examine the formant frequencies of the stop consonants and the phonemes on either side of it. The formant frequencies are the primary frequencies that make up a phoneme, and the way they change can indicate certain phonemes. The 7 features used as input to the network are mnss (max in beginning), mdp (min in beginning), mdpvowel, b-37 (max in beginning), Lin (max in beginning), reldur, and closedur. Full details about the makeup of the features used for stop consonant identification can be found in Ali et al., 1999.

3. NETWORKS

3.1 Network Architecture

Many neural networks designed to deal with temporal signals are what are known as time delay neural networks [Watrous, 1988; De Mari & Flammia, 1993]. These networks take one frame of input at a time, where the full input is a set of these frames. A series of connections delays the propagation of the signal to the next layer of the network for any number of time steps. This results in the network having several time steps presented to it at once. The length of time the network can examine is limited, however, by the maximum delay in the network; a signal longer than that maximum delay will not be properly processed. Even given that limitation, time delay neural networks have been successfully used for phoneme recognition.

The networks used in this research are instead designed using a recurrent connection in the hidden layer to create a sort of "short term memory" that allows the network to process temporal information. The networks are all three-layer feedforward networks with the addition of a context layer that is fully connected from the hidden layer, meaning that every hidden layer node connects to every context node, and the context node is fully connected back to the hidden layer one time step later. Thus at every time step, the hidden layer has as input information about the current sound input to the network along with the information about the previous input to the network stored in the context layer. A diagram of the basic network architecture used in this project is shown in Figure 4. All of the networks used were trained using momentum and backpropagation of error.

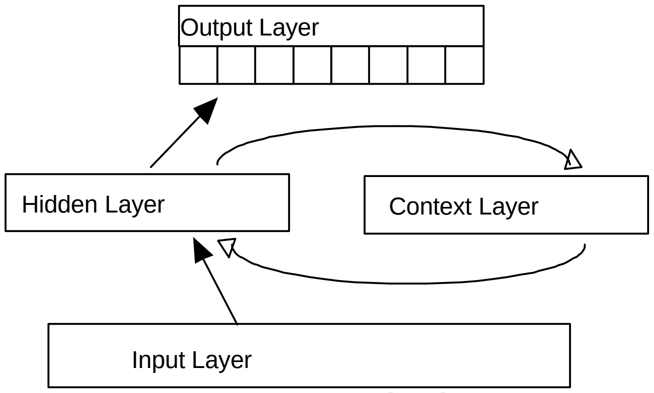

Figure 4: Basic network architecture.

3.2 Network Simulators

To train and run the networks, two different neural network simulator packages were used. The first package used was the Matlab neural network toolbox. This package is a very versatile package, and it was hoped that we would be able to set up a network with a series of smaller sub-networks trained to look for specific features in each phoneme. A significant amount of time was spent designing code that would set up the data in the form required by the toolbox. However, it turned out that the Matlab neural net toolbox could not be trained on many time-dependent patterns unless the patterns were all of the same length. The patterns we used in this project were taken from continuous speech and were of varying length. After much discussion with the Matlab developers, it was finally determined that we could not use Matlab for what we needed to do.

At that point, we began to using the TLEARN neural network simulator package. While not quite as versatile as Matlab in the design of the networks it could simulate, it was able to process time-dependent patterns of different lengths. At this point, more code was created to create the data files needed by TLEARN.

3.3 Network Input

For the three forms of data, the phoneme was fed to the network along with a small portion of the vowel on either side of the main stop consonant. This extra vowel information was included because the identity of a stop consonant can in part be determined by the way the vowel on either side behaves. The intensity of the vowel and the way it changes as it flows into the stop consonant all contain important information about the consonant. Including a portion of the vowel gives the network this extra information along with the information contained in the phoneme itself. Each phoneme was presented to the network one time step at a time. Thus the entire process of running a phoneme through the network extended over several time steps, and produced an output pattern that also extended over several time steps.

3.4 Network Output

The network was trained to reproduce an output function that kept the incorrect output nodes at zero while the correct output node activation increased from zero to one as the phoneme progressed. This increase was an s shaped increase, with steeper slope in the center than at the beginning and end of the sample (Figure 5). This type of target function was used because the main portion of the stop consonant was in the center of the sample, with portions of the surrounding vowels on either side. This function emphasizes that the most important part of the data is the central portion, containing the actual stop consonant. Each input sample had its own target output function tailored to the correct length so that the output node activation started at zero and went to one at the very end of the pattern.

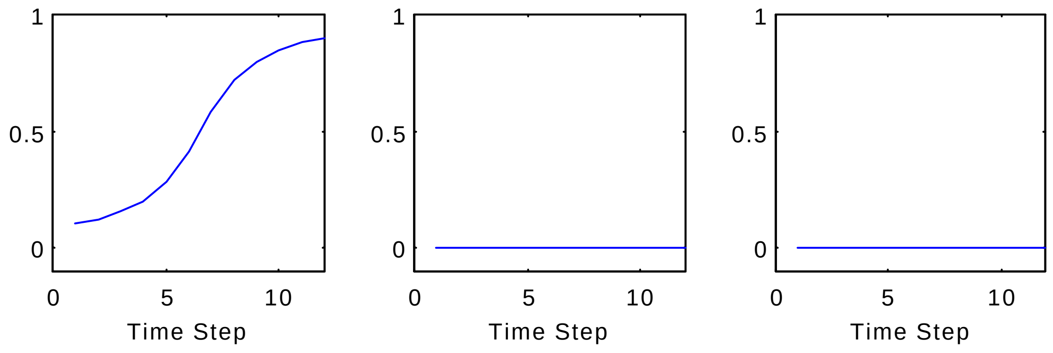

Figure 5: Training function a three output-node network

4. EXPERIMENTS AND RESULTS

One major challenge in designing the networks for these experiments was the difficulty in selecting the proper network parameters and sizes. While most of the networks tried could learn the training set fairly well, when we tested the network's generalization on novel data the performance depended significantly on the size of the hidden layer. Thus many of the tests we ran involved different size hidden layers, because there was no good way to determine the ideal setup without extensive trial and error experimentation. In addition, it was not obvious what learning rate and momentum constant would yield the best performance. A learning rate of 0.1 and a momentum constant of 0.5 were used for the experiments simply because these settings seemed to work.

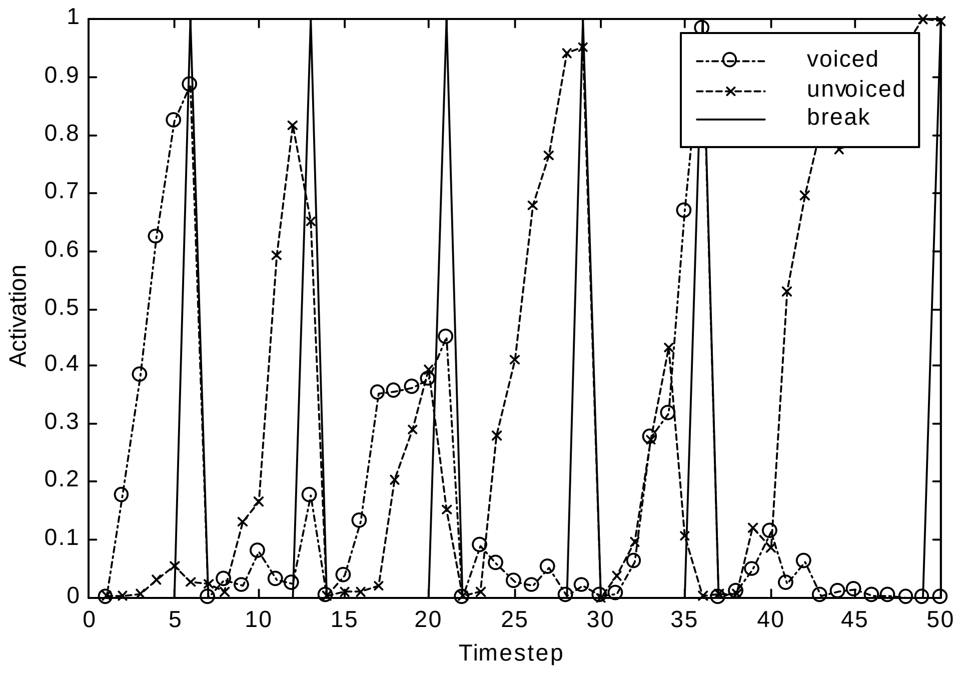

Figure 6: Example of network output from a voicing detection network

One other difficulty was determining how to classify the output of the networks. While the goal for the output was to keep all but the correct nodes at zero while the correct node itself ended up having an activation level of one by the end of the pattern, this rarely happened. Instead, the activity of other output nodes fluctuated up and down as the pattern progressed (Figure 6). This sometimes meant that the activity of an incorrect node would become higher than the correct node for a frame or two, and then drop back to zero. Other times, the correct node would have a high level of activation for the entire pattern, only to drop down close to zero at the final time. We avoided the problem of trying to figure out which output peak to use by taking the integral of the signal produced by each output node over the length of the pattern. The node with the highest activation for the most time was judged to be the network's decision.

4.1 Voicing Detection

The first experiment we ran involved detection of whether the presented phoneme was voiced or not. We used the three different types of sound representations discussed earlier as inputs to three separate neural networks in an attempt to see which representation was best suited for the problem. There were 159 total stop consonants in the training set used for these tests, all taken from the TIMIT database. These patterns were recorded from four speakers, two male and two female, all of whom spoke the same dialect of English. Out of these 159 patterns, 61 were voiced and 98 were unvoiced. To create a balanced training set, 50 random samples were taken from the voiced and the unvoiced groups, for a total of 100 training samples. The rest of the samples were used to test the network on its performance. To make sure that one specific set of input was not more conducive to training the network than another, several different random sets of data were used.

We started by simply using a spectrogram as input. The input consisted of 129 channels of spectrogram information, and the output was two nodes, one for voiced and the other for unvoiced. Each frame of the spectrogram was taken from 256 samples of phoneme, and the frames overlapped by 64 samples. This resulted in each pattern being on average 12 frames long, with some as short as 8 frames and some as long as 17 frames. The output layer of the network consisted of two nodes, one representing voiced and one representing unvoiced. The target function for the output nodes had the correct output node's activation increase from zero to one, as described earlier, while the incorrect node stayed at zero.

Two different network designs were used with the spectrogram input-- one with 20 hidden nodes and one with 10 hidden nodes. The network with 20 hidden nodes was trained for 200 epochs and learned the training patterns almost flawlessly, with 96% accuracy. The test patterns were identified with an average accuracy of 62%. Further training was deemed unnecessary since the training patterns were being reproduced well already.

The 10 hidden node network was trained for 400 epochs. It learned the training set with approximately 95% accuracy. However, some of the training runs did not generalize very well at all. Instead, they seemed to have learned to identify all of the patterns as either voiced or unvoiced, depending on the run. Six out of eight runs did this. The other two networks identified the stop consonants as either voiced or unvoiced with an average accuracy of 65%.

The next type of input used was the Bark scaled input. This consisted of 20 bands of information presented to the network at each time step. Each frame of data was taken from 512 sample section of phoneme, and each time step overlapped with the previous time step by 265 samples. This resulted in samples that were on average eight frames long, ranging in length from five to 10 frames. The change to the FFT size and the overlap was unintentional. It occurred because the code for generating each of the data files was written at separate times, and the change in parameters was not noticed until after the experiments were run. The target functions were the same as for the spectrogram input.

We used networks with 15 and 20 hidden units. The 15 hidden unit network was run several times for up to 1200 epochs. Training times varied because it was unclear what was the best length of time to train the network for. Once again, the network only learned to distinguish voiced from unvoiced some of the time; other times it seemed to only identify the patterns as mostly one or mostly the other. For the 15 hidden unit network, the maximum performance on the untrained phonemes was approximately 70%. This performance was achieved at 1000 epochs of training. The 20 hidden unit network was trained for 1000 epochs and again only learned to differentiate the two types of patterns about half of the time. The peak performance of the 20 unit network was an average of 75% accuracy.

The cepstral network input consisted of 30 points of cepstrum data. Each 30 points of data were derived from 512 sound samples, and each frame overlapped by 256 points. The target function was the same as used for the other two voicing recognition experiments. These networks also identified a disproportionate number of the phonemes as either voiced or unvoiced. The networks used had hidden layers of 20 and 25 nodes, and were trained for 600 epochs, although the performance was typically better at 400 epochs.

Both networks, when they did not identify most of the stops as solely voiced or unvoiced, achieved an average performance of 70% accuracy. This broke down into an accuracy of 60% on the unvoiced and 80% on the voiced stops, however, so even these networks were somewhat biased in one direction.

4.2 Placement of Articulation Detection

The network design for determining the placement of articulation of the stop consonants was almost identical to the design for voicing. The only difference was that the output from the network was three nodes instead of two, one representing labial stops, one representing alveolar stops, and one representing palatal stops. The input data to the network was identical to the input for the voicing detection runs. There were 62 palatal stops, 61 alveolar stops, and 36 labial stops in the data set. The training set contained 30 random stops of each type, and the remaining stops were included in the testing set. Because of the poor results from and the long training times needed for the voicing detection networks trained on the spectrum data, only the Bark scaled data and the cepstral data were used for the articulation networks.

The networks for using Bark scaled input to identify place of articulation had 20 and 30 hidden nodes. The network with 20 hidden nodes could not learn to reproduce the training set correctly. It would instead identify the stops as being produced in one location much more often than in any other location. The 30 hidden node network did not have this problem, however. It was trained for 800 epochs several times, although the peak performance was at 600 epochs. This network had an average accuracy of 75% correct for place of production.

The cepstral networks used hidden layers of 30, 35, and 40 nodes. The 30 and the 35 performed very poorly, not achieving better than 55% accuracy. However, the 40 node networks performed fairly well. One trained 40 hidden node network achieved an average accuracy rate of 78.5%, and the cumulative average over all the 40 node networks was 74% correct. These networks were trained for between 800 and 1200 epochs, with the peak accuracy falling at various times within those limits.

4.3 Feature Analysis

The networks that took the extracted phonetic features as input were designed slightly differently than the rest of the networks used in this research. The main difference is that only one set of features represented each stop consonant in the set. Thus there was no need for any time dependency in the network; the network instead was simply a three-layer feedforward network. The input layer consisted of seven inputs (one for each feature), the hidden layer was 30 nodes, and the output was two nodes for voicing detection and three nodes for placement detection.

Each of the inputs had a different range of possible values. Using Matlab, each of these ranges was normalized to between zero and one, as that is what the TLEARN package requires for its input. The data were then exported to the data files used by TLEARN, and Matlab was later used to analyze the generated data. The output was similar to the output for the other experiments detailed in this report, but it was solely binary, with the training target of one for the correct output node and zero for the other incorrect output nodes.

A total of 498 phonemes were in the set of feature data used for this experiment, evenly divided between labial, alveolar, and palatal stops. For the training set, 400 randomly selected phonemes were used, with the other 98 being used as the testing set. The networks were trained for 1000 epochs with a learning rate of 0.2 and a momentum constant of 0.5.

For voicing, the average accuracy of the network on patterns that it had not been trained on was 80%. For placement identification, the average accuracy for untrained patterns was 78%. However, these numbers could probably be improved, as the network designed used was the only design tried, and further experiments using these data could probably come up with a more efficient network.

5. DISCUSSIONS AND CONCLUSIONS

The original intention of this project was to use both the extracted phonetic features and some form of time-dependent speech information as inputs to a neural network. Unfortunately, too much of the time allotted for this project was spent attempting to use the Matlab neural network toolbox to simulate the networks used in this research; only after struggling with the system for several weeks did we find that Matlab could not use the input in the correct format. However, all of the necessary preliminary steps have been taken towards this goal, so future research can continue where this project left off.

An important conclusion of this research is that neural networks using recurrent layers can handle the input in the format used and do something useful with it. While the peak accuracy rates of 75% for voicing and 80% for placement are not good enough to be used immediately for realistic computer-based phoneme recognition, they are much higher than random chance and show that there is good potential in this network design for the phoneme detection problem.

In addition, further refinement of the network design and of the training procedures will probably lead to even higher accuracy rates. Although we attempted to vary the parameters in a systematic was, not enough test runs were performed and not enough different designs were tested to determine the ideal network configurations for the problem.

Another important result of this research was that using the features used [Ali et al., 1999] could also be successfully used as input to a neural network. Both the sound-based networks and the feature-based networks achieved 75% and above accuracy rates at their best. However, it is likely that some of the time-dependent features used could not be detected by the network design as it stands, and it is also likely that the network picks up on features not included in the seven used as input to the feature-based networks. Thus each set of inputs contains some different data, so a combination of the two inputs into one neural network will probably lead to higher recognition rates.

One problem that needs to be addressed in future research is the question of which of the three forms of input used in this research is the best for phoneme recognition. The tentative result from the data collected here is that the linearly spaced spectrum input is the worst and the Bark scaled spectrum input is the best, with the cepstral input only slightly worse than the Bark scaled input. Too many parameters changed between the spectrum input and Bark input, however, for it to be a clear-cut case. Training the spectrum-based networks was certainly much slower, because of the higher number of internal connections involved.

Future research with these networks should run both identification networks simultaneously. Getting a 75% accuracy on both voicing and placement identification tells us only that the worst case identification rate would be 56%. This number would probably be higher, but only by running both tests simultaneously on the same set of data can the actual accuracy be obtained. Also, using more speakers would further validate the accuracy rates obtained here.

It can be concluded from the results obtained in this research that these recurrent networks could potentially be used for phoneme recognition, especially if further design modifications are made to improve their accuracy. The accuracy of these networks does not approach the accuracy of that Ali et al. (1999) achieve using a purely algorithmic approach, which was 97% accuracy for voicing and 90% of place of articulation. However, these networks solved the problem purely through backpropagation of error. It is hoped that through further design modifications recurrent neural networks will be shown to be even more useful for phoneme classification than has been shown here.

6. ACKNOWLEDGMENTS

I would like to thank Ahmed M. Abdelatty Ali, Dr. Jan Van der Spiegel, and Dr. Paul Mueller for their assistance and inspiration on this project, without which this would not have been possible. I would also like to thank the National Science Foundation for their support of undergraduate research through the Research Experience for Undergraduates program.

7. REFERENCES

1. E. Zwicker, Subdivision of the Audible Frequency Range into
Critical Bands (Fequenzgrupper), *J. Acoust. Soc. Am.,* 33, 1961, 248.

2. R. W. Schafer and L. R. Rabiner, Digital Representation of Speech Signals, *Reading in Speech Recognition (A. Waibel and K. Lee, eds.),* Morgan Kaufmann, San Mateo, 1st ed., 1990, p. 58-62.

3. R. L. Watrous, *Speech Recognition Using Connectionist Neural Networks,* Ph. D. Thesis, UPENN, 1988.

4. R. De Mori and G. Flammia, Speaker-Independent Consonant Classification in Continuous Speech with Distinctive Features and Neural Networks, J. Acoust. Soc. Am., 94 (6), 1993, p. 3091-3193.

5. H. T. Edwards, *Applied Phonetics: The Sounds of American English*, Singular Publishing Group, San Diego, 1st ed., 1992, p.p. 61-98.

6. A. M. A. Ali, J. Van der Spiegel, and P. Mueller, An Acoustic-phonetic Feature-Based System for the Automatic Recognition of Fricative Consonants, Proceedings of ICASSP, 1998.

7. A. M. A. Ali, J. Van der Spiegel, and P. Mueller, Acoustic-phonetic Features for the Automatic Classification of Stop Consonants, IEEE Transaction on Speech and Audio Processing, (in press, 1999).