NEURAL NETWORK WITH APPLICATION TO REAL TIME PHONEME RECOGNITION

NSF Summer Undergraduate Fellowship in Sensor Technologies
EunSik Kim (Electrical Engineering) – University of Pennsylvania
Advisors: Dr. Paul Mueller, Dr. Jan Van der Spiegel

ABSTRACT

In hopes of improving real-time speech recognition, a biologically based phoneme recognition algorithm was implemented on the NP-4 neural computer. The NP-4 neural computer, which contains programmable interconnects, neurons, synapses, and synaptic time constants, is extremely useful in computation of real-world dynamic patterns as they occur in speech. Prior to this summer, some implementations were done on the NP-4 neural computer, and the goal during this summer was to improve work done in early stages. The newly developed algorithm, implemented in the host computer, allows neurons to be trained to respond to a particular phoneme. Testing was performed once the network was trained to find the overall responses of the neurons. The algorithm shows much promise for recognition of phonemes, with over 90% positive responses.

1. INTRODUCTION

For the past few years, many researchers and companies have devoted time and money to developing reliable real-time speech recognition programs and designs. Many different approaches have been suggested, with variable degrees of success to date; the statistical analysis method has been one of the most successful. However, more and more researchers have come to gain a new perspective on biologically motivated systems, many of which have proved to be far more efficient.

The resurgence of neural network research has motivated efforts to utilize knowledge about biological aspects of speech recognition in hopes of increasing the reliability of speech recognition. This paper describes the approach taken using the Corticon/University of Pennsylvania NP-4 neural computer for real-time speech recognition.

2. ARTIFICIAL NEURAL NETWORKS AND THE ANALOG NEURAL COMPUTER

2.1 Artificial Neural Networks

Artificial neural networks are information-processing systems with certain performance characteristics in common with biological neural networks. Four basic assumptions underlie the generalization of mathematical models of neural biology [1]:

1) Many simple elements called neurons process information

2) Signals are passed between neurons over a connection link
3) Signals transmitted in neural nets are multiplied by synaptic weights, which are associated with the connection link
4) The output of the neuron is determined by the activation function applied at its input

2.2 The Analog Neural Computer

The NP-4 neural computer consists of programmable interconnects, neurons, synapses, and synaptic time constants. The computer runs in analog mode, but the whole network, including the synaptic weights, neuron parameters, and time constants, is set by the digital host. The synaptic weights are programmable over 3.5 orders of magnitude, the time constants are programmable between 1 microsecond and 1 second, and the neuron transfer function can be selected. The neurons operate by taking the sum of weighted inputs, which produces a corresponding output that relates either linearly or non-linearly to these inputs. Figure 1 below describes the simple operation of the neural computer:

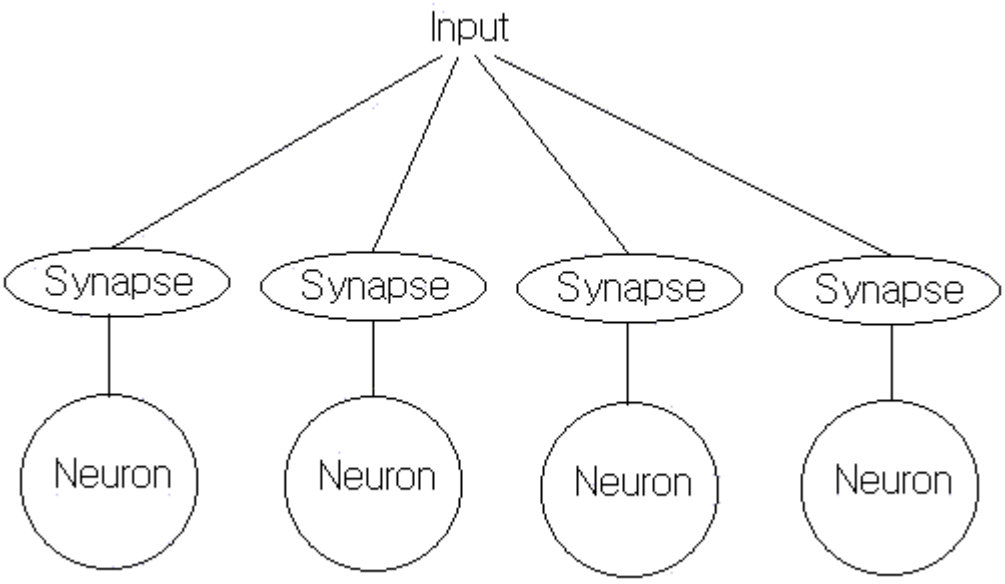

Figure 1: Neural network schematic

3. METHODOLOGY

3.1 Overview of Implementation

The implementation of vowel recognition algorithm is based on the two basic intrinsic elements of speech: amplitude and frequency. For different sounds, the amplitude and frequency combination differs. Since different amplitude-frequency distributions define different phonemes, the vowel recognition algorithm was implemented using this very property.

The algorithm was designed so that when the vowel sound is presented, the network is programmed and the network is trained. The sound needs to be presented only once. The training of neurons mean the synaptic values connecting to each output neuron are calculated and set so that the neuron will respond only to that sound. Only one output neuron can be trained at a time.

The input to the neural computer comes from band pass filters. The phoneme is passed through 16 such filters, and these 16 different band frequency signals are then fed into the neural computer. The overall topology of the algorithm can be broken into three major layers: the preprocessing layer, which includes the center surround function and the high energy cut-off function; the inverse layer; and the output layer, which includes the mutual inhibition that couples the output neurons. The overall topology of the network is shown in Figure 2 below:

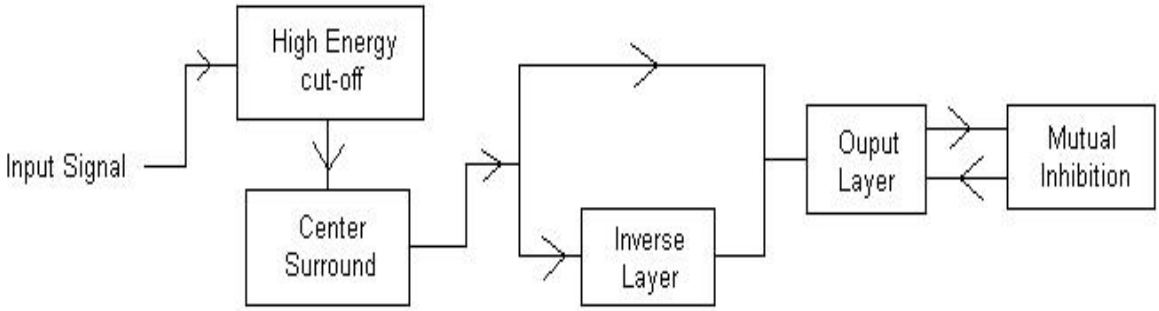

Figure 2: Overall schematic

3.2 Center Surround Function

The preprocessing stage of the algorithm is necessary for reducing the noise and emphasizing the input signals that come from the band pass filters. The 16 inputs received from the band pass filters are first routed into 16 neurons through the synapses in the neural computer. Since each signal consists of a different frequency band, the synapses are set to emphasize the signal of that band and repress the others. In other words, each neuron is responsible for heightening the band with which it is associated while reducing effectively the others. This ensures that the incoming signals are emphasized and thus easily recognized by the network, making formants more identifiable, and also has the effect of canceling out noise. The center surround function in Figure 3 below shows the relationship between the distance from a particular band and the multiplying factor.

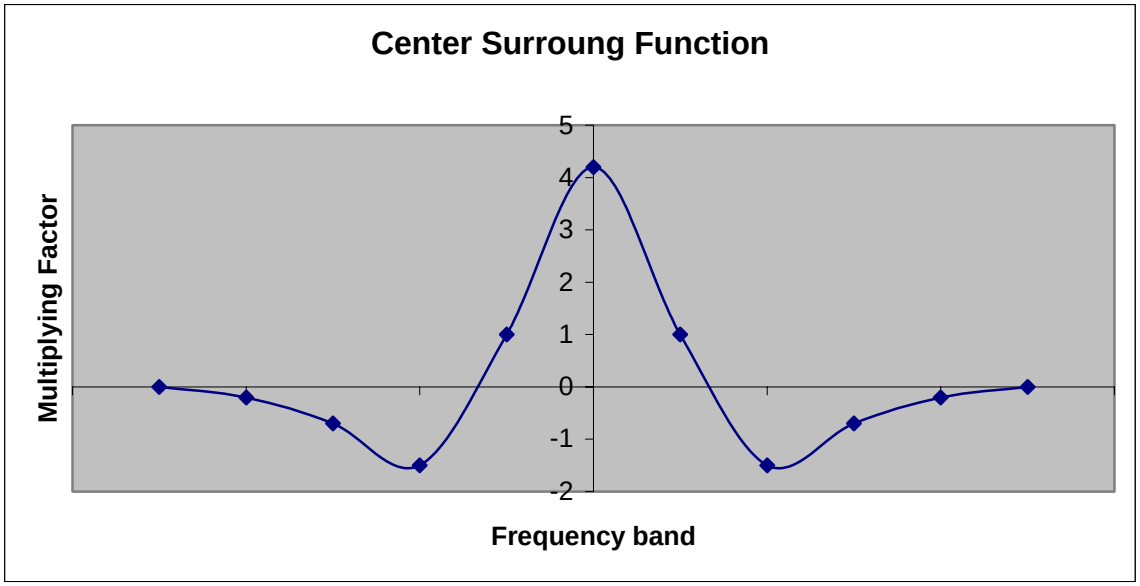

Figure 3: Plot of Center Surround Function

3.3 High-Energy Cut-off Function

One difficulty in speech recognition is the variability of different speakers. Besides the obvious accentual variations of different people, the volume of speech is one example. People, not being robots, vary in their volume when they speak, with some generally loud and others soft. But the problem arises when an individual speaker talks loud. As speech becomes louder, the frequency distribution seen by the neural computer becomes wider and wider, hence making it hard to discern the formants of a particular phoneme. The neural computer is uncertain how to respond to that sound.

To deal with loud inputs, a set of neurons was implemented. A logarithmic relation between the output functions of a neuron and its input could avoid the problems caused by loud speakers by allowing for quick growth when volume is low and inhibiting the growth when the volume becomes higher. However, the output functions of the neurons in the NP-4 neural computer are linearly related to their inputs. Hence the high-energy cut-off layer was implemented at two different points to imitate the logarithmic curve. This layer examines the output from the center surround layer and inhibits it as it reaches a certain cut-off threshold. The relationship between the input and the output of the high-energy cut-off block is shown in Figure 4 below:

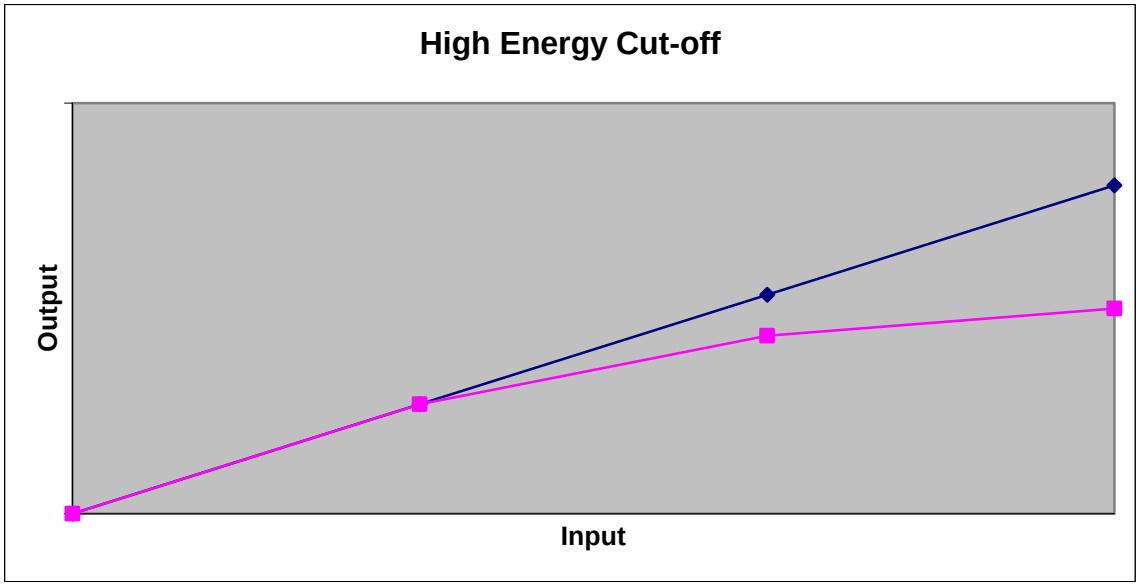

Figure 4: Logarithmic growth of the algorithm

3.4 Inverse Layer

The inverse layer, created as a safety check for recognizing phonemes, has a rather simple function. The inverse neurons of the inverse layers act as an inverter of the input to the network. The outputs from the center surround function are routed into synapses that have negative weights. The function curve looks exactly like the center surround function curve, except it is inverted. Thus these inverse neurons are always firing, and are suppressed only when the output of the input neurons rises. The output of these inverse neurons is in turn connected to the output layer negatively, thus inverting the input signal twice, creating a fuzzy AND function. The inverse function curve is shown in Figure 5 below:

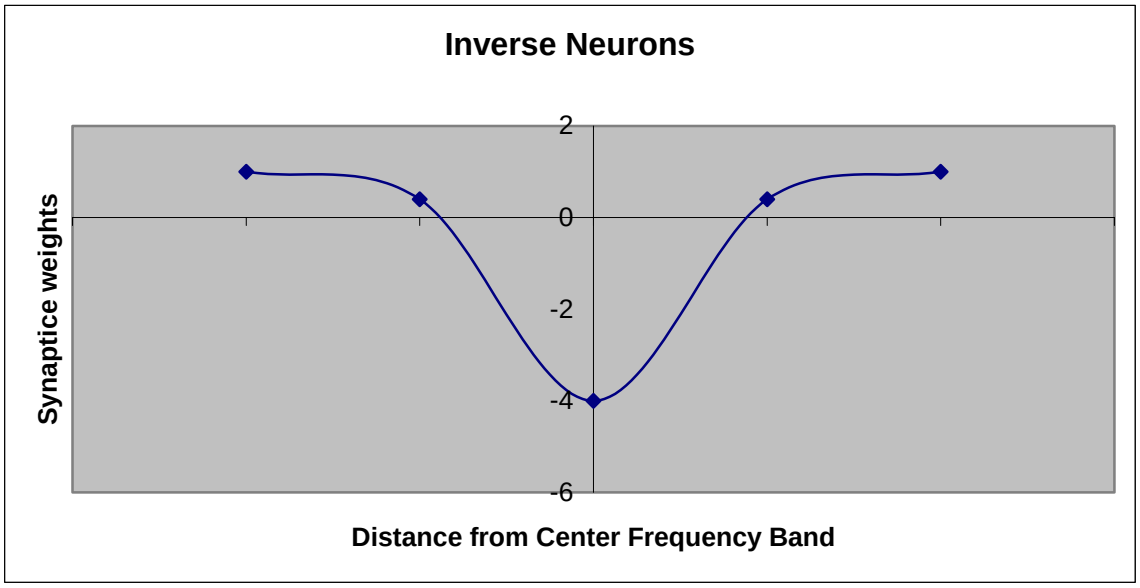

Figure 5: Inverse function

The inverse neuron inverse the incoming signals, making it fire constantly when no input signal is present.

Figure 6 below shows an example of how the inverse neurons are connected from input to the output neurons:

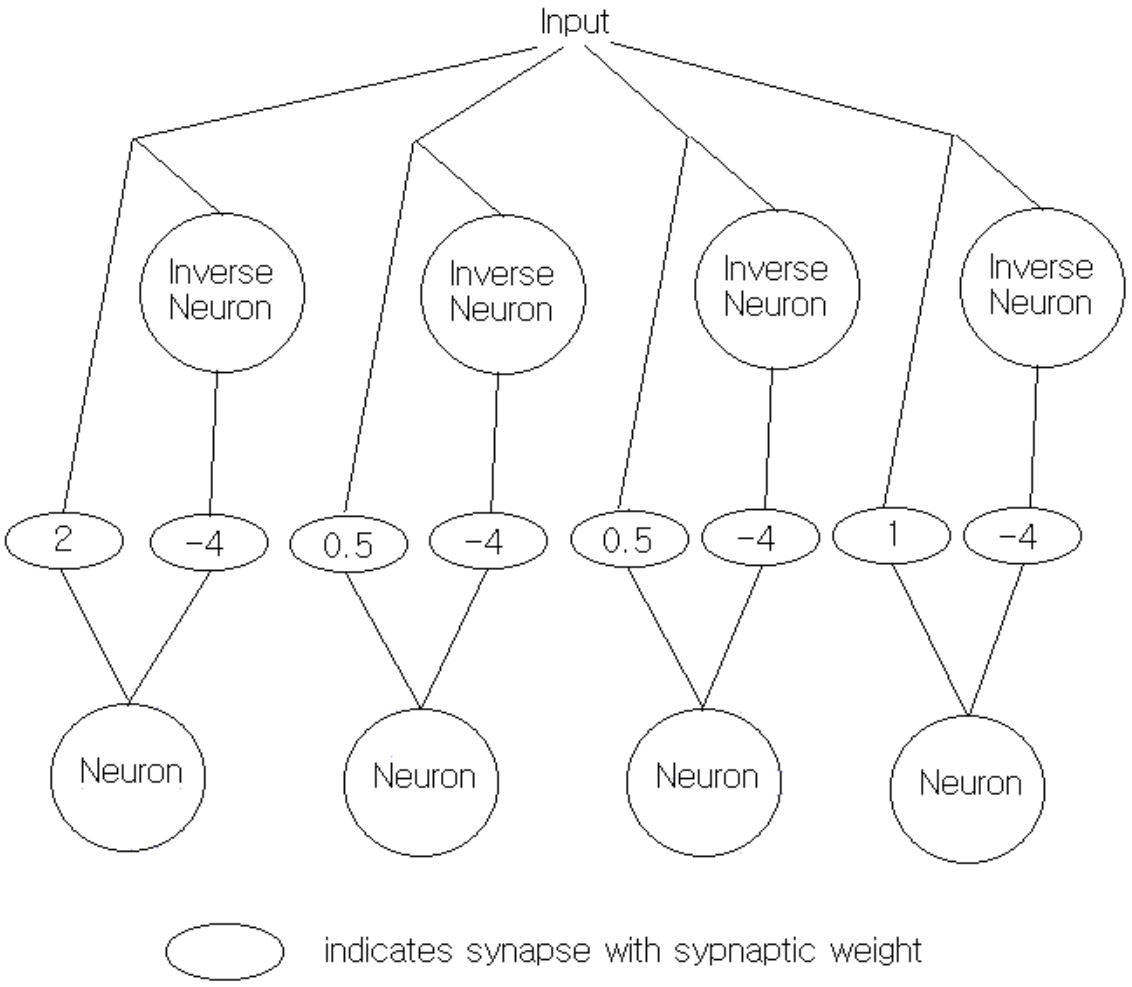

Figure 6: Example showing the operation of the inverse neurons

3.5 Mutual Inhibition

Mutual inhibition is set up in such a way that the outputs of each output neuron are routed into every other neuron, including itself. It is used to select the strongest firing output neuron. All output neurons are set such that they excite themselves but strongly inhibit the other output neurons. This setup enables the stronger neuron to fire more frequently while repressing the other neurons. Eventually only one neuron will produce output from the set of neurons that are mutually inhibited.

The Pseudo-code implementation describing this function is given below:

```
for (i = 0; i < n; i++) {
if (i  = current_neuron) {
     excite;
} else {
     inhibit;
}
}
```

3.6 Output Layer

There are two different ways to train the output neurons. One is taking the maximum point of each time slice, and the other is taking the average of all the points in each time slice. These two methods are considered below.

3.6.1 Maximum point method

The training algorithm is a one-shot training, meaning the output neuron is trained only once when the sample is presented to the network; no other training for that neuron is necessary. As only one neuron is trained at a time, several training must be done in order to set several output neurons. The activity of both the input neurons and the inverse neurons are sampled at 2 ms intervals during the time the sample is presented to the network. Since there are 16 inputs from the cochlea, there are 16 input neurons, and for each input neuron *i*, the sample data is cut into several time slices. This means that the sample points are divided into several time slices S_n. Within each time slice for every input neuron, the maximum point $M_{i,n}$ is found, which effectively leaves one data point in each time slices.

Next, the maximum points of each input neuron are totaled. This is done for every time slice in order to find and compute the slice with the most activity. Once the maximum slice is found, the input neurons are positively connected to the output neurons using the data point within the maximum slice to compute the synaptic weights. The exact number of output neurons to be set can be varied, but four connections seem to work best. The synaptic weights for the connections between the input neurons and the output neurons are based on the percentage of the activity of the particular neuron to the overall activity during the maximal time slice. These synaptic weights are automatically set immediately after the training.

The synaptic weights connecting the inverse neurons to the output neurons are set such that the value of weight is equal to the sum of weights connecting the input neurons to the output neurons. This is shown as

$$\sigma_k = \sum \sigma_{i,}$$

where σ_k is the synaptic weight connecting the inverse neurons and σ_i is the synaptic weight connecting the output neurons.

All the inverse neurons are connected to the output neurons at this same value. This prevents the output neurons from responding if the input signals that were present during the training are missing.

3.6.2 Average point method

The only difference between the average point method and the maximum point method is in the choosing of data points for each time slice when reducing the number of sample points in each of them. Unlike the maximum point method, which ignores all sample points except for the maximum point within each time slice, the average point method takes into account every sample point and averages them. The rest of the method is the same, finding the maximal slice and computing the synaptic routine.

3.7 Inhibition of the Weak Inputs

After the synaptic weights are set positively between the input neurons and the output neurons, input neurons with input signals smaller than the firing-threshold are negatively set. This prevents the neurons from responding to background noise. The synaptic weights are set at an arbitrary value no bigger than 1.

4. RESULTS

Both the training and testing samples were obtained from the TIMIT database of speech samples made jointly by Taxes Instrument and MIT. For training and testing, samples of vowel sounds “ah,” “ee,” and “oh” were used. They were chosen because of the differences between “ah” and “ee” and the similarities between “ah” and “oh”.

Several series of test were carried out using the database. First, the network built was tested to find the most effective number of synapses that connect the input neurons to the output neurons. This was done by choosing one particular phoneme and training 12 neurons to different representations of that particular phoneme sound from the TIMIT database. Then all 16 synaptic weights connecting the input neurons to the output neurons were set and their responses tested. Next, the number of synapses set was reduced.

However, the responses in input neurons made it obvious that only four synapses from the input neurons to the output neurons had to be set in order to identify the formants. In most cases only four bands constituted the core frequencies and the rest were not needed. Performance-wise, the network’s response was better when only four synapses were set, since setting more synapses made the neurons more susceptible to background noise.

4.1 Results for Maximum Point Method

Next, the neurons were trained with the maximum point method and tested, both with and without the inhibition of weak inputs. The results are displayed in a confusion matrix. The horizontal represents the sounds responded, and the vertical represents the sounds presented. The main diagonal represents the correct responses, and the rest represent unwanted responses. Therefore the closer the entries in the main diagonal are to 100 % and the closer the entries in the rest are to 0%, the better. The columns and rows do not

necessarily add up to 100%, since more than one neuron fired at times. The testing without the inhibition of weak inputs produced the results shown in Table 1:

	/iy/	/aa/	/ow/
/iy/	80%	10%	0%
/aa/	30%	80%	30%
/ow/	0%	20%	80%

Table 1: Confusion matrix using the maximum point method without the inhibition of weak inputs.

Table 2 shows the results obtained with inhibition of weak inputs:

	/iy/	/aa/	/ow/
/iy/	80%	10%	0%
/aa/	30%	80%	30%
/ow/	0%	20%	80%

Table 2: Confusion matrix using the maximum point method with the inhibition of weak inputs.

As the results show, inhibiting the weak inputs had little effect on the response of each neuron. The result is not very satisfying.

4.2 Results for Average Point Method

Finally, the neurons were trained with the average point method and tested. Again, they are displayed using the confusion matrix. Table 3 shows the results obtained without using the inhibition of weak inputs:

	/iy/	/aa/	/ow/
/iy/	90%	0%	0%
/aa/	22%	88%	22%
/ow/	0%	10%	100%

Table 3: Confusion matrix using the maximum point method without the inhibition of weak inputs.

Table 4 gives the results obtained using the inhibition of weak inputs:

	/iy/	/aa/	/ow/
/iy/	90%	0%	0%
/aa/	22%	88%	22%
/ow/	0%	10%	100%

Table 4: Confusion matrix using the maximum point method with the inhibition of weak inputs.

Neurons trained with the average point method produced a better result than the neurons trained with the maximum point method. The reason is that the inputs do not consist of one maximal point; in many cases, the inputs consisted of two maximal points. Thus the input does not look like one mountaintop, but rather like two, with a drop in between the maximal points. The maximum point method does not take this into account, as it searches only for the maximum point among the sample points within that time slice. Because the average point method *does* take it into account, it more accurately reflects the slope of the curve.

Also, in neither case did the inhibition of weak inputs make much difference. However, since the testing sounds were obtained from the TIMIT database, background noise or the noise signal in the cochlea was not significant enough to alter the results. If this algorithm is to be used in a real-time, real-world situation, where there is significant background noise, this function will inhibit those noises and prevent the trained neurons from responding to the wrong phonemes.

5. DISCUSSION AND CONCLUSIONS

This project has been quite successful. Despite many difficulties trying to recover old files and several crashes of the operating system, the results seem promising. They show that the neural network is currently capable of supporting all steady state phoneme recognition. The neurons were not only tested against the sounds they were trained for, but they were also tested with sounds they were not trained with, in other words, their negative responses. Though the number of testing sounds were quite limited, nevertheless there were enough to see the overall response of the network and it turned out to be much acceptable.

However, this algorithm does not seem very promising if extended beyond phoneme recognition, especially in real-time speech recognition. The overlapping of frequency bands between different sounds could create confusion for the network. Thus in order to further improve the performance of the network, a thought on third algorithm was raised. Instead of finding the time slice with the most activity, the maximum point of each frequency band could be found and the time constants could be set such that the incoming signals are passed through the synapses connecting to the output neuron synchronously.

This seemed possible to better the performance but due to time limitations this algorithm was not implemented.

6. RECOMMENDATIONS

Further examination of the vowel sounds and the preprocessing stage would be useful. This research found that, depending on the preprocessing stage - the center surround function in particular - the result could vary significantly.

Furthermore, the diphones could be taken into account. Since they are more complicated than the phonemes, additional functions may be needed. The algorithm seems quite promising for the phonemes, but extending it to diphones might prove not as easy.

In addition, as for day-to-day recommendation, all source files should be backed up at all times. Even when working on the program XPHYS on the digital host, it is recommended that the user save the project periodically since the Venix operating system could crash at any time. Also, the cochlea should be reset before training or testing, as it becomes unstable and gives erroneous results if left on for few hours.

7. ACKNOWLEDGMENTS

I would like to thank my advisors, Drs. Jan Van der Spiegel and Paul Mueller of the University of Pennsylvania, for their help, patience, guidance, insight, and support on this project. I would also like to thank Dr. Van der Spiegel for allowing me to participate in the SUNFEST program, which not only has broadened my knowledge in the neural networks field, but has also educated me in different areas of engineering, such as engineering ethics.

I must also thank Lois Clearfield for taking care of all the details and for all of her encouragement.

8. REFERENCES

1. Rabiner, L. and Juang B. *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1993
2. Van der Spiegel, J., Donham, C., Etienne-Cummings, R., Fernando, S., Mueller, P., Blackman, D. *Large Scale Analog Neural Computer with Programmable Architecture and Programmable Time Constants for Temporal Pattern Analysis*, IEEE International Neural Network Conference, Orlando, FL, 1994.
3. Mueller, P. *Speech Recognition based on Models of the Auditory System using a Neural Computer*.
4. Mueller, P., Van der Spiegel, J., Blackman, D., Donham, C., Etienne-Cummings, R. *A Programmable Analog Neural Computer with Applications to Speech Recognition*, Proc. Comp. & Info. Sc. Symposium (CISS), J. Hopkins, 1995
5. Rabiner, L., and Schafer, R. *Digital Processing of Speech Signals*. Prentice Hall PTR, 1978