Neural Networks and Neurophysiological Signals

by

Srikant Sarda

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 19, 1999

Author .....
Department of Electrical Engineering and Computer Science
May 19, 1999

Certified by .....
Steve Burns
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Thesis

Neural Networks and Neurophysiological Signals

by

Srikant Sarda

Submitted to the
Department of Electrical Engineering and Computer Science

May 19, 1999

In Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The purpose of this thesis project is to develop, implement, and validate a neural network which will classify compound muscle action potentials (CMAPs). The two classes of signals are "viable" and "non-viable." This classification system will be used as part of a quality assurance mechanism on the NC-stat nerve conduction monitoring system. The results show that standard backpropagation neural networks provide exceptional classification results on novel waveforms. Also, principal components analysis is a powerful preprocessing technique which allows for a significant reduction in processing efficiency, while maintaining performance standards. This system is implementable as a real-time quality control process for the NC-stat.

Thesis Supervisor: Steve Burns
Title: Professor, Electrical Engineering and Computer Science

Table of Contents

Chapter 1

Introduction

1.1 Company and Product

This thesis project is being researched with the help of scientists and physicians from NeuroMetrix, Inc. This company has created novel tools for the diagnosis of neuromuscular disorders. Among those tools is a device called NC-Stat which is used to detect systemic and entrapment neuropathies of the median nerve.

The NC-Stat nerve conduction monitoring system consists of a hand-held electronic monitor, single use disposable biosensors and a docking station that communicates with a remote information service. This device measures median nerve distal motor latency (DML) and F-wave latency by stimulating the nerve with short, painless, electrical impulses and detecting and processing the biopotentials evoked in the thenar muscles of the hand. The DML and F-wave latency provide the clinician with the highly valuable objective information to complement the history and physical examination thereby optimizing patient care and decreasing associated costs.

1.2 Problem Statement

The ability to quantitatively evaluate peripheral nerve function in the point-of-care setting (e.g. in the physicians office) has recently been advanced by the development of hand-held nerve conduction monitoring systems such as the NC-stat. These instruments allow physicians to objectively diagnose and manage common neuromuscular conditions such as Carpal Tunnel Syndrome, diabetic neuropathy and back pain. However, because these instruments measure relatively complex and low-level bioelectrical signals, the probability of acquiring non-viable signals due to noise or various artifacts (e.g. due to nerve stimulation) is significant. Furthermore, because the users of these diagnostic devices are not

experts in the field of neurodiagnosis, it is the responsibility of the instrument to differentiate viable from non-viable signals. This distinction is particularly important in the clinical environment because misclassification of signals can lead to potentially dangerous misdiagnoses.

Thus far, the aforementioned classification problem has been approached using traditional signal processing techniques with some success. However, these techniques require rigorous characterizations of the universe of non-viable signals, which is a very difficult task. An alternative approach to the algorithmic techniques, is to utilize neural networks to learn and capture the "expert" knowledge carried by an experienced clinical neurophysiologist. It is the general aim of this research project to develop, implement and validate such a neural network as part of the NC-stat nerve conduction monitoring system.

1.2.1 Description of Signals

Before the goals of this project can be stated, the waveforms in question must be described in more detail. The signals acquired by the NC-stat which measure the DML and M-wave are also known as compound muscle action potentials (CMAPs). These signals have distinct regions that follow a certain structural pattern (Please refer to Figure 1.1). The first part of the signal (0-1 ms) is characterized by an artifact region that gives no relevant information about the DML or M-wave. After this region comes a delay region for about 5 ms that leads to a takeoff point. This point is where the DML value is calculated. Following the takeoff is the actual M-wave continuing until about 13 ms.

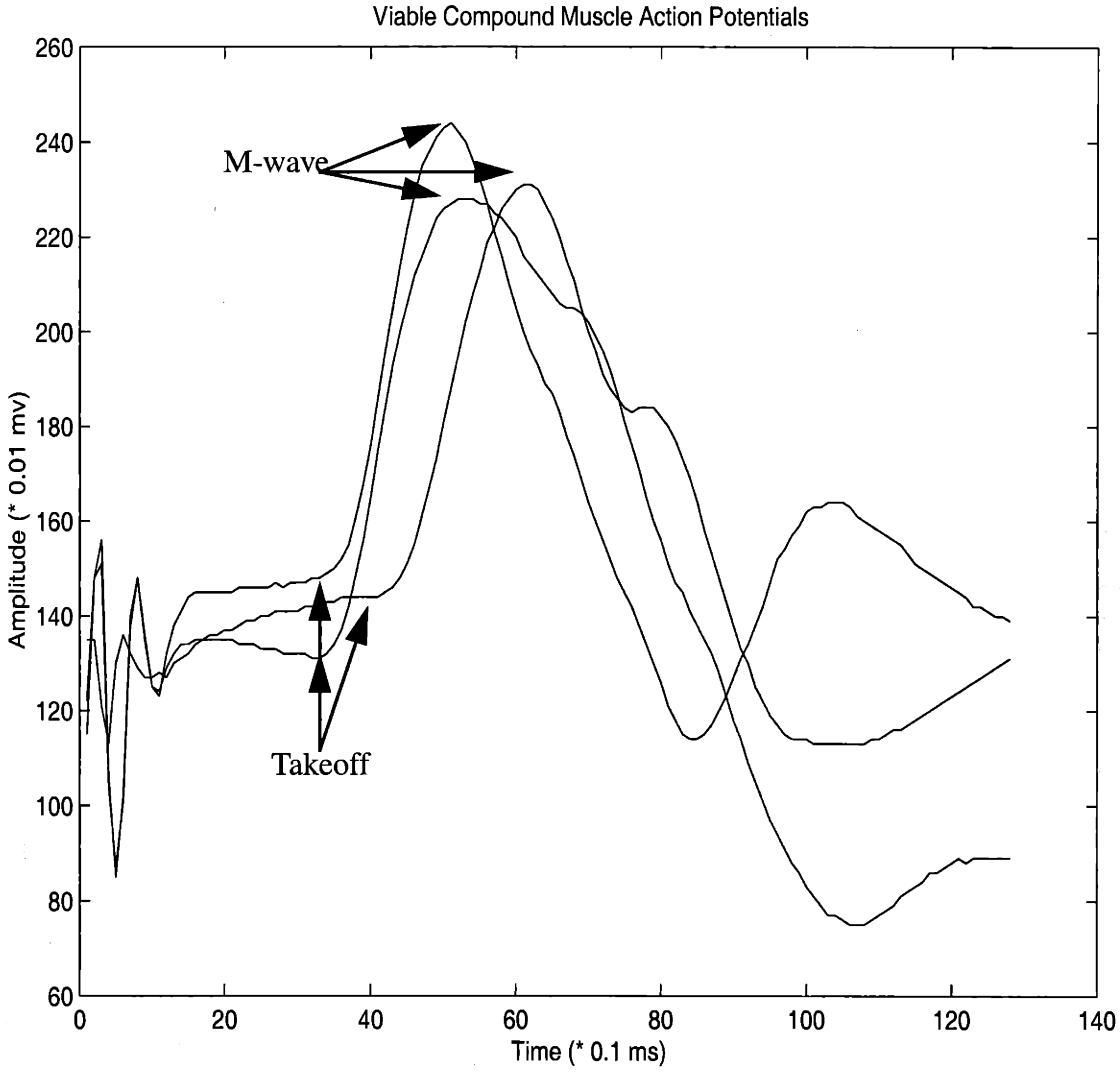


Figure 1.1: Viable Compound Muscle Action Potentials

These signals represent the viable CMAPs that can be used by the neurophysiologist. There is a well-defined takeoff point where the DML can be calculated. However, the NC-stat does not always produce viable signals. The following signals represent non-viable signals that have been acquired by the device (Figure 1.2). The most common causes of poor waveforms are that the sensor has not equilibrated to the patients skin because the wrist was not clean, the cleansing alcohol has not dried or the test was initiated too soon after placement of the sensor.

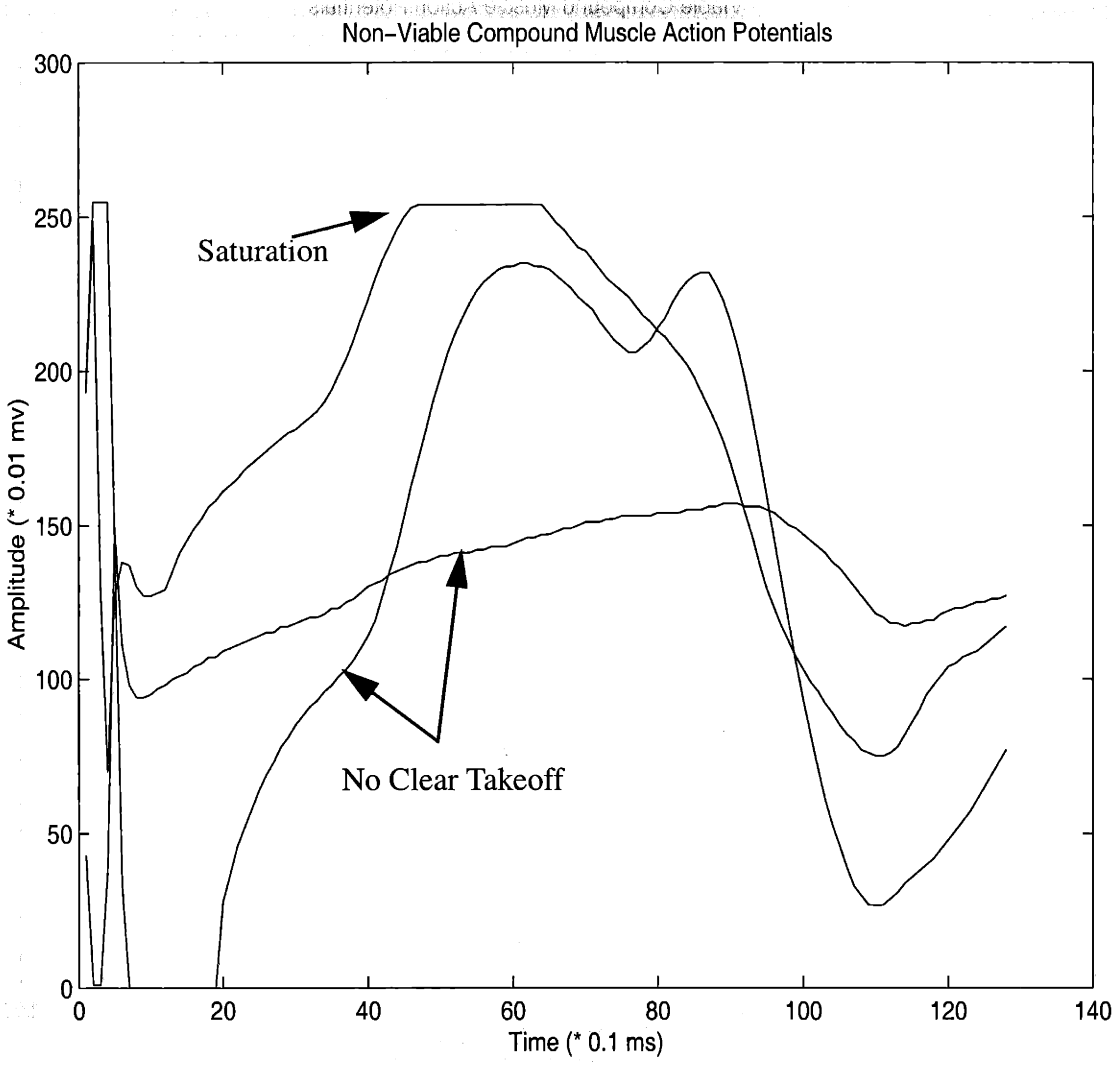


Figure 1.2: Non-Viable Compound Muscle Action Potentials

These non-viable waveforms can be characterized by some subtle qualities. First, there is no clear takeoff point for the signals, so no accurate DML can be calculated. Also, there is clear saturation in the signals, causing discrepancies in the M-wave. This also affects the takeoff value. Non-viable signals occur infrequently, but must be detected by the device so that new signals can be acquired. A classification system which distinguishes between these two types of signals can be used as a quality control mechanism on the NC-stat that can potentially save valuable resources.

1.3 Goals

The project consists of the following three specific aims:

1. Investigate, design and prototype a neural network that classifies non-invasively measured compound muscle action potentials (CMAPs) as "viable" or "non-viable." (Please see Figures 1.1 and 1.2) This specific aim will include the following sub aims.

• Construct and catalog appropriate training and validation databases from the existing NeuroMetrix database of normal and pathologic CMAPs.

• Work with neurophysiologist at NeuroMetrix to define concept of CMAP "viability" and classify CMAPs in training and learning databases.

• Define rigorous criteria for comparing performance of different implementations of networks and network models. Sensitivity and specificity of neural network classifications will be included among the criteria.

• Explore various network types, learning strategies and other parameters and converge on the best implementations.

2. Engage in a detailed exploration of the best implementations obtained in specific aim #1 with the goal of identifying the optimal neural network candidate for subsequent real-time implementation. This specific aim will include the following sub aims.

• Thorough quantitative comparison of the best implementations.

• Definition of decision criteria.

• Application of decision criteria to determine optimal implementation.

3. Develop a real-time implementation of the optimal neural network on the multiprocessor NC-stat nerve conduction monitoring system. This specific aim will include the following sub aims.

• Create a detailed system specification.

• Propose a design that will meet the specifications.

• Implement the design. Implementation will be submitted to both a static (code review) and dynamic (processing of simulated data) validation and verification prior to release.

• Design and conduct a prospective experiment that evaluates the performance of the NC-stat with the real time CMAP classification neural network. This experiment will demonstrate that the network correctly classifies non-viable CMAPs which will be created by physically (e.g. modifying the electrochemical transduction gel in the sensor) or electronically (e.g. introducing an impedance mismatch) altering the detection sensor.

Chapter 2

Background

2.1 Neural Networks

The system is being implemented with the use of neural networks. A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use [1]. The fundamental architecture of a neural network is based upon many neurons interconnected with different weights and parameters. The most important property of a neural network is its ability to "learn" from its environment. Learning is a process by which the weights and parameters of a network are adapted through an ongoing process of stimulation by the environment in which the network is embedded. The learning is an iterative process that depends on many different parameters.

There are two different classes of learning: supervised and unsupervised. In supervised learning, a neural net is "trained" or "learned" with a set of training data and output using one of many different types of training algorithms [1]. Then the resulting trained network is tested with other test data in order to evaluate its performance. This process is iterated until a desired performance is established. Some parameters which the training and learning depend on are the size and variety of the training data, the size of the neural net, the learning algorithm used, and the resemblance of the test data to the training data. The difference with unsupervised, or self-organized learning, is that there is no external teacher to oversee the learning process. There are no specific examples of the function to be learned by the net. Unsupervised learning algorithms perform clustering of the data into similar groups based on the features or measured attributes serving as input to the network [2].

Both supervised and unsupervised pattern classification can be performed by neural networks.

The architecture of the neural network is a vital characteristic which must be chosen carefully. This architecture will be different depending on the kind of training (supervised or unsupervised) selected. Supervised pattern classification is often accomplished by means of a feedforward architecture, like a backpropagation network. This is the architecture that will be primarily investigated for this problem due to its popularity and proven effectiveness. Backpropagation refers to the adjustment of the weights of a net in order to reach an optimal output (i.e. minimize the error).

The topology of a neural network refers to the number of hidden layers (excluding input and output layers) and the number of neurons per layer (see Figure 2.1). Determining the number of layers and neurons for a network is similar to picking which training algorithm to use. Backpropagation networks require the investigator to guess the number of hidden units that will be adequate to solve the problem. It is usually an iterative process in which the number and sizes of the layers are varied until an optimal solution (if one exists) is found. The general rule of thumb is that each hidden layer in a network correlates to one feature of the input. For example, if a net contained two hidden layers, then the network is probably extracting two features from the input. There are many trade-offs that must be considered when choosing an architecture for a neural network. The larger a network becomes, the more training and processing time that the network requires.

The training algorithm is another important aspect to the creation and performance of a neural network. Many training algorithms already exist and have been proven to provide good results for specific problems. The training algorithm is what determines how the weights are adjusted during each pass of the training of a backpropagation network. Some algorithms, such as batch training and gradient descent training, use very simple

approaches to weight adjustments. Others, like the Quasi-Newton method, involve sophisticated algorithms whose ultimate goal is to minimize the error of the projected outputs from the actual outputs [1]. Choosing which training algorithm to use for a particular problem, pattern classification in this case, is usually based on a trial-and-error basis. There is no specific training algorithm that works well for a problem.

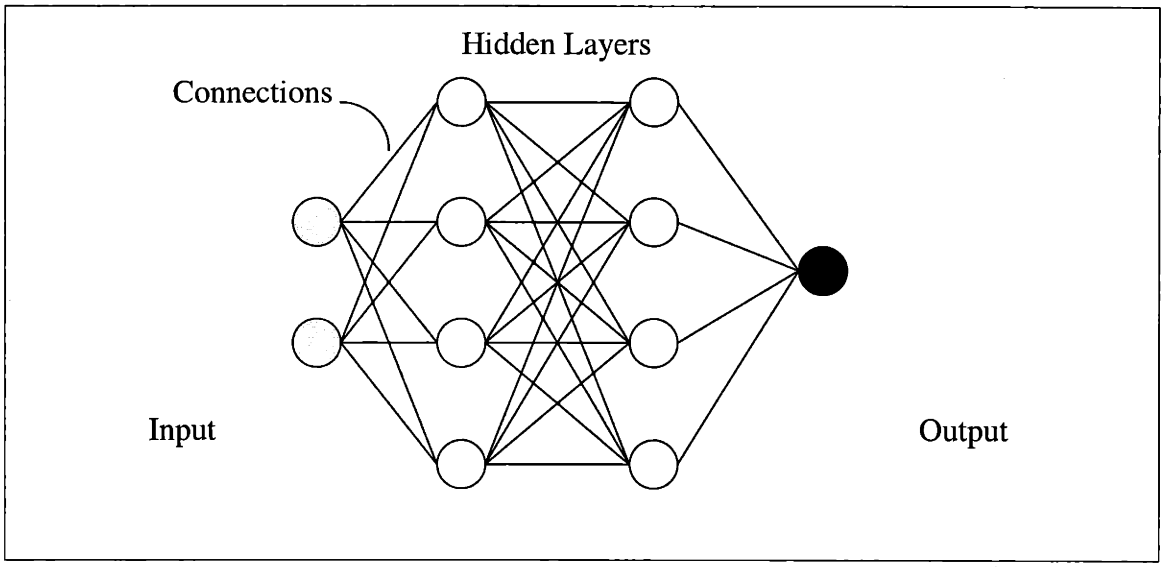


Figure 2.1: Neural Network

2.2 Principal Component Analysis

Another common statistical technique, principal component analysis (PCA), can be applied to this pattern classification problem. PCA is an ordination method whose goal is to reduce the dimensionality of a set of data. PCA is often used as a preprocessing technique to inputs for neural networks to solve these types of problems. Since the dimensionality of the input (CMAP) in this problem is a vector composed of 128 points, it would be beneficial to reduce the number of dimensions while maintaining the same vital characteristics of the data.

The principal components of a set of data are found by first forming the covariance (or correlation) matrix of a set of patterns and then finding the minimal set of orthogonal vec-

tors that span the space of the covariance matrix. Once the basis set has been found, it is possible to reconstruct any vector in the space with a linear combination of the basis vectors. So the principal components are vectors that maximize their linear correlation with all other variables. They summarize the general variation in the data.

2.3 Why Neural Networks

Neural networks are universal approximators, and they work best if the system you are using them to model has a high tolerance to error. They work very well for capturing associations or discovering regularities within a set of patterns where: the number of variables or diversity of the data is very great; the relationships between variables are vaguely understood; or, the relationships are difficult to describe adequately with conventional approaches.

Depending on the nature of the application and the strength of the internal data patterns, a neural net is generally expected to train quite well. This applies to problems where the relationships may be quite dynamic or non-linear [2]. Neural networks provide an analytical alternative to conventional techniques which are often limited by strict assumptions of normality, linearity, variable independence etc. Because a neural net can capture many kinds of relationships it allows the user to quickly and relatively easily model phenomena which otherwise may have been very difficult or impossible to explain otherwise [2].

The appeal of neural networks as pattern classification systems is based upon several considerations. They appear to perform as well or better than other classification techniques and require no assumptions about the nature of the distribution of the input data. A comparison of neural networks to classical methods, like K-nearest neighbor and discriminant analysis, has shown that neural networks can achieve equal or better performance using a much smaller set of training data.

2.4 Related Research

Much research has been completed in the area of neural networks and pattern classification. The most commonly cited applications of neural networks can generally be termed as classification problems. Even though this precise problem has never been explored, other classification problems can be related closely. These problems range from the field of medicine to the world financial markets. Here are a few examples of problems involving the classification of bioelectrical signals with the use of neural networks.

Studies have been completed regarding the usefulness of neural networks in medical diagnostic applications. In a paper written by Chappell, Lee, and Taylor [4], a comparison was made between neural network solutions versus traditional statistical methods in the classification of somatosensory evoked potentials. In this particular problem, quantitative studies and heuristic insight led to the definition of four distinct classes of the potentials. "Benchmarking neural network solutions against statistical techniques has been undertaken and the results seem favorable." This was concluded from the results of the classification problem involving these somatosensory evoked potentials recorded from patients with severe head injuries. The results indicated that the classification was system was successful 77% of the time, which was greater than that of any other statistical technique.

A very similar problem to classification of CMAPs is that of ECG Beat Classification, researched by Hu, Mani, Palreddy, and Tompkins [5]. The classification of these beats has been a difficult problem due to the large dimension of the feature space and significant overlap between class boundaries. The approach used in this paper involved the use of feedforward networks along with Self Organizing Maps (SOM). The feature space was divided into several regions and individual classifiers were developed for each region separately. An architecture was proposed which yielded classification percentages in the high

to mid-nineties. This paper shows that classification of the ECG beats using feedforward neural networks was a viable and successful approach.

The classification of Visual Evoked Potentials (VEP) by feedforward neural networks was studied by Dorronsoro, Gonzalez, Lopez, and Siguenza [6]. VEP have served for the discrimination between normal and pathological stages, or early detection of diseases such as multiple sclerosis, hypothyroidism, and aging related problems. The object of the study was to classify the VEP into two different categories. The VEP were acquired from experiments performed on rats. The two classes of signals were control and hypothyroid rats. The feedforward network created consisted of two hidden layers that classified with a 71% success rate. This was superior to the 65% classification rate achieved by the human experts.

Chapter 3

Experiments and Results

3.1 Research Specifications

MATLAB provides a neural network toolbox that makes it easy to create and simulate virtually any type of neural network. Most of the research (training and testing) was conducted using this robust toolbox. It is a powerful toolbox which is easy to use with a prior knowledge of MATLAB commands.

3.1.1 Data Sets

In order to have trained and tested the neural networks thoroughly, it was imperative to obtain viable data. The data sets for this project are split into two categories: non-viable and viable CMAPs. The data sets with viable CMAPs were constructed from studies completed by Dr. Shai Gozani at NeuroMetrix. The non-viable signals were carefully constructed from filtered white noise. These sets grew enormously over the course of the entire project. Initially, there were only 60 unique viable signals to train and test with. Towards the end of the project, there existed three different sets of data from different studies totalling around 1200 acceptable CMAPs.

The abnormal CMAPs were easier to gather since this set consisted of signals that didn't resemble the normal CMAPs to any degree. At first, the set was composed of only band-limited white noise with the same power spectrum as the initial 60 normal CMAP data set. The band-limited white noise was an effective way to approximate non-viable signals since none were available at the time. The size of data set of band-limited white noise was unlimited, yet all of the signals were limited to the same power spectrum to that

of the viable signals. Also, the signals were scaled randomly according to amplitude statistics of the viable data set. The distribution amplitudes of the viable CMAPs was acquired and the noise was scaled randomly using this distribution. This was a true unbiased sample that represented the entire universe of possible non-viable CMAPs (See Figure 3.1). As more data was recorded, more non-viable signals recorded from the device became available.

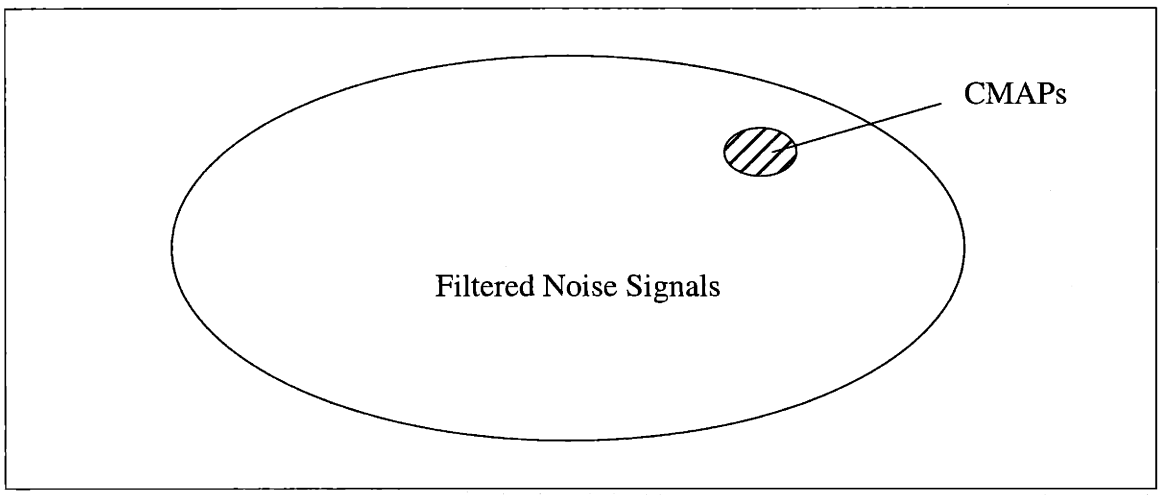


Figure 3.1: Diagram of Signals

Each data set was systematically labeled and stored in text files for easy access. It was necessary to develop a systematic approach to construct and catalog these data sets. As more data became available, it became more difficult to track and update the data sets, so a standard technique was valuable for maintaining the data. Once the data was obtained, it was also helpful to use the expertise of neurophysiologists to categorize the signals. As the data sets grew larger, distinguishing between viable and non-viable CMAPs became more difficult. Thus, an expert was needed to help classify such signals and other signals already existing in data sets.

As mentioned before, the construction of the training and validation sets are very critical for the performance of a neural network. In general, most of the viable CMAPs look alike in appearance. This made it easy to pick signals for training versus testing. A random process separated the signals into two different groups. The next question to be answered was how many signals to choose for training. This depended on how many signals were available. In the beginning of the research, very few viable signals were in the database, so the both the training and testing sets were limited in number. However, once this data set became larger, it was difficult to pinpoint the ideal number of signals to achieve the optimal solution. This was again an iterative process where the number of signals in the training set were varied and corresponding results were assessed (this is evident in the results below). The same method was applied for the non-viable (band limited white noise) signals.

3.1.2 Evaluation of Neural Networks

When a backpropagation neural network is trained with this MATLAB toolbox, the number of epochs must be specified. The epochs refer to how many passes (back and forth) the training algorithm makes adjusting the weights of the connections in the network. After each epoch, a mean-squared error (MSE) is calculated using the target output values and the actual output values of the network at that point. The object of training a neural network would be to minimize this particular error. However, a lower MSE does not necessarily mean that the network performs better. Another issue to consider is the possibility that the training leads to a local minimum error (i.e. a solution only for a certain part of the problem) instead of reaching to a global minimum. If the MSE is high and does not change after training for a few epochs, then the training has most probably led to a local minimum.

The evaluation of the neural networks is based on sensitivity and specificity on the test data sets. These two performance criteria rely on statistics produced by the testing sets. The statistics can be broken down as true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). Sensitivity is the percentage of true CMAPs that are correctly detected (Sens = TP/(TP+FN)). These statistics were available from the results of the viable CMAP validation set. Specificity is the fraction of non-viable CMAPs detected as false CMAPs (Spec = TN/(TN+FP)). The specificity was determined from the results of the non-viable CMAP test set.

Each of the signals in the training set was trained with a set of target values. Since there are only two sets of signals to categorize, it was obvious to train one set to achieve a value of zero and the other set to target the value of one. This way it was easy to distinguish how the signals were classified in the validation data sets. However, some of the validation signals produced outputs in the entire range of zero to one. This made it necessary to create some method of classifying these signals which were categorized "in the middle." Therefore, a cutoff had to be established for classification of the validation CMAPs and noise. Initially this cutoff was set to a predefined value (0.7), and all of the networks were evaluated with sensitivity and specificity measurements using this cutoff value. However, as the research progressed, a more effective method was used to evaluate the performance of the netoworks. This evaluation technique involved the use of ROC curves to determine the most optimal cutoff value.

Receiver Operating Characteristic, or ROC, analysis can be used to compare the performance of classifiers when prior probabilities of occurrence and/or costs of misclassification are either unknown or varying [2]. The ROC curves for this problem are made up of sensitivity and specificity readings for different values of the cutoff. If the cutoff is

changed for a particular network, then the performance of the network should change accordingly. This evaluation technique allows for the determination of the optimal cutoff for a particular neural network, depending on which performance value (sensitivity or specificity) needs priority. Ideally, one would want both criteria to approach unity. However, trade-offs must be made. Among these criteria for the classification of CMAPs, sensitivity is considered most critical. This is because missing a valid CMAP would be more expensive and time-consuming than missing an invalid signal. ROC analysis can be used to easily and robustly assess the performance of all classifiers, whether they are neural networks or statistical models.

The entire analysis was performed in MATLAB. The toolbox allowed a simple way to record and test the neural nets that were created. If the net produced acceptable results, then the neural net was saved and tested with other data sets as they became available. The evaluation of each network included a ROC curve and corresponding sensitivity/specificity readings.

3.2 Phase One Results

The research was performed in two separate phases. The first phase (performed before the proposal for this project was written) tried to narrow the possible types and sizes of neural nets that could solve the problem. The next phase tried to pinpoint the most optimal structure and actually implement the network in a real-time environment. The results are presented in tables which give the different neural network architectures and corresponding performances on the test data. Each row of the table gives the parameters for that particular network.

The first phase of the research started with an initial set of sixty viable signals. The set of non-viable data was created (as mentioned above) from white noise which was filtered, yielding signals that had the same power spectrum as that of the viable signals. The filter was created using the power spectrum of the viable data set. The next step was to identify how to split up the data sets into training and testing portions. Since only sixty viable CMAPs were available, the initial training set consisted of thirty randomly chosen viable CMAPs and fifty random noise signals as non-viable CMAPs. More noise signals were used in training because of the broadness of the spectrum of non-viable signals. The argument was that if more non-viable signals were included in training, then this would give a better representation of the possible non-viable CMAPs to the network. Since the viable CMAPs looked very similar to each other, the number of valid signals for training should not matter as much.

The next task to complete was to identify the relative size of the neural networks that would be needed to correctly classify these CMAPs. This was a trial and error session where many different layers and sizes of networks were trained. The input to every network was 128 points representing the signal. The output was a single value computed by a sigmoidal transfer function that limited the range from zero to one. The number of hidden layers and neurons per layer were essential parameters that were varied during this study. Another parameter explored in this phase was the training algorithm. The factors used to assess the algorithms were training time (slow or fast) and performance of the network

(MSE, sensitivity, specificity). Table 3.1 summarizes the findings of this phase of the research (these were standard backpropagation feed-forward neural networks):

# of Neurons in Hidden Layers	Training Algorithm	MSE	Size of Training Set	Sensitivity/ Specificity
5	BFG	0.098	80 (30/50)	0.6/0.8
8	DA	0.006	80 (30/50)	0.9/0.8
10	LM	0.088	80 (30/50)	0.7/0.83
40	RP	0.089	80 (30/50)	0.8/0.83
10,3	LM	0.037	80 (30/50)	0.9/0.8
10,3	DX	0.056	80 (30/50)	0.83/0.93
8	DX	0.007	80 (30/50)	0.97/0.73
8,2	LM	0.058	130 (30/100)	0.75/0.9
10	SCG	0.009	130 (30/100)	0.96/0.91
10,3	DX	0.024	130 (30/100)	0.9/0.85
10,5	DX	0.011	130 (30/100)	0.92/0.91
12,5	DX	0.007	130 (30/100)	0.99/0.89

Table 3.1: Phase One Results

The table can be interpreted as follows:

• The first column describes the topology of the network. The values represent the number of neurons in each hidden layer (if there are two values, then that network has two hidden layers).

• The second column tells which training algorithm was used. BFG = BFGS quasi-Newton backpropagation; DA = gradient descent with adaptive learning rate; LM = Levenburg-Marquardt backpropagation; DX = batch training with momentum and variable learning rate; SCG = scaled conjugate gradient training; RP = resilient backpropagation

• The third column gives the final mean-squared error value after training.

• The fourth column is the number of signals used for training. It is further broken down by how many were viable and how many were non-viable (viable/non-viable).

• The final column gives the sensitivity and specificity readings of the network on the test sets, which are the same sizes as the training sets.

3.2.1 Analysis of Phase One Results

Many conclusions could be drawn from these results:

• The number of neurons for the best performing networks were in the ten to twelve range. Also, two hidden layers seemed to do better than a single hidden layer on average.

• As far as training algorithm, the DX (batch training with momentum and variable learning rate) algorithm achieved the best performance in terms of speed and sensitivity/ specificity. This algorithm trained in the smallest amount of time compared to SCG, LM, and RP. It also produced some of the best performance numbers. This algorithm is a simple gradient descent training, or

• MSE did not really factor into the evaluation process, since most of the errors were significantly small (as suspected earlier) and there was no correlation between a small MSE and improved performance.

• As the number of non-viable signals increased in the training set, performance of the networks tended to improve.

• For the better performing networks, both the sensitivity and specificity values seemed to lie in the same ballpark (i.e. there was not much trade-off if one value was preferred over another).

These findings are based on a consistent cutoff value of 0.7. This was because ROC analysis was introduced in the next phase of the research. The 0.7 value was arbitrary and seemed to give the most optimal performance numbers for every network.

These results gave a very good impression of the type and topology of neural network that should be used to solve this classification problem. This was a good start to the next phase of the research, where the most optimal network was explored and implemented in a real-time system.

3.3 Phase Two Results

This phase of the research attempted to pinpoint the exact topology of a neural network that would perform best in the classification of viable compound muscle action potentials. Also, this presented an opportunity for other techniques and parameters to be explored and analyzed.

3.3.1 Backpropagation Networks

One important aspect of the evaluation of the networks added in this phase was the ROC analysis. This allowed for an optimal cutoff value to be chosen for each separate network. Also, the DX training algorithm was isolated in this part, due to the positive results from the first section. This gave for the opportunity to study this specific algorithm in depth. The first series of networks trained were an extension from the previous phase of research. Every net was a fully connected backpropagation network with two hidden layers of varying size. Here are the results of this set of networks:

# of Neurons in Hidden Layers	Learning Rate	Cutoff Value	Sensitivity	Specificity
10,5	0.03	0.7	0.93	0.84
10,5	0.02	0.7	0.93	0.90
10,6	0.02	0.7	0.95	0.87
10,6	0.015	0.9	0.90	0.92
11,4	0.03	0.8	0.94	0.86
11,4	0.03	0.9	0.87	0.92
11,4	0.025	0.8	0.91	0.89
11,5	0.035	0.7	0.91	0.94
11,5	0.035	0.8	0.86	0.96
11,6	0.02	0.7	0.93	0.92
11,6	0.02	0.8	0.91	0.95
12,4	0.02	0.6	0.85	0.92
12,4	0.02	0.5	0.93	0.86
12,5	0.04	0.5	0.94	0.90
12,5	0.04	0.6	0.91	0.92
12,5	0.04	0.7	0.90	0.94
12,6	0.02	0.7	0.93	0.90
12,6	0.02	0.7	0.92	0.92
13,4	0.02	0.6	0.93	0.91
13,4	0.02	0.7	0.92	0.93
13,5	0.03	0.7	0.93	0.92
13,5	0.03	0.8	0.89	0.95
13,6	0.04	0.5	0.93	0.90
13,6	0.04	0.6	0.92	0.93

Table 3.2: Phase Two Results

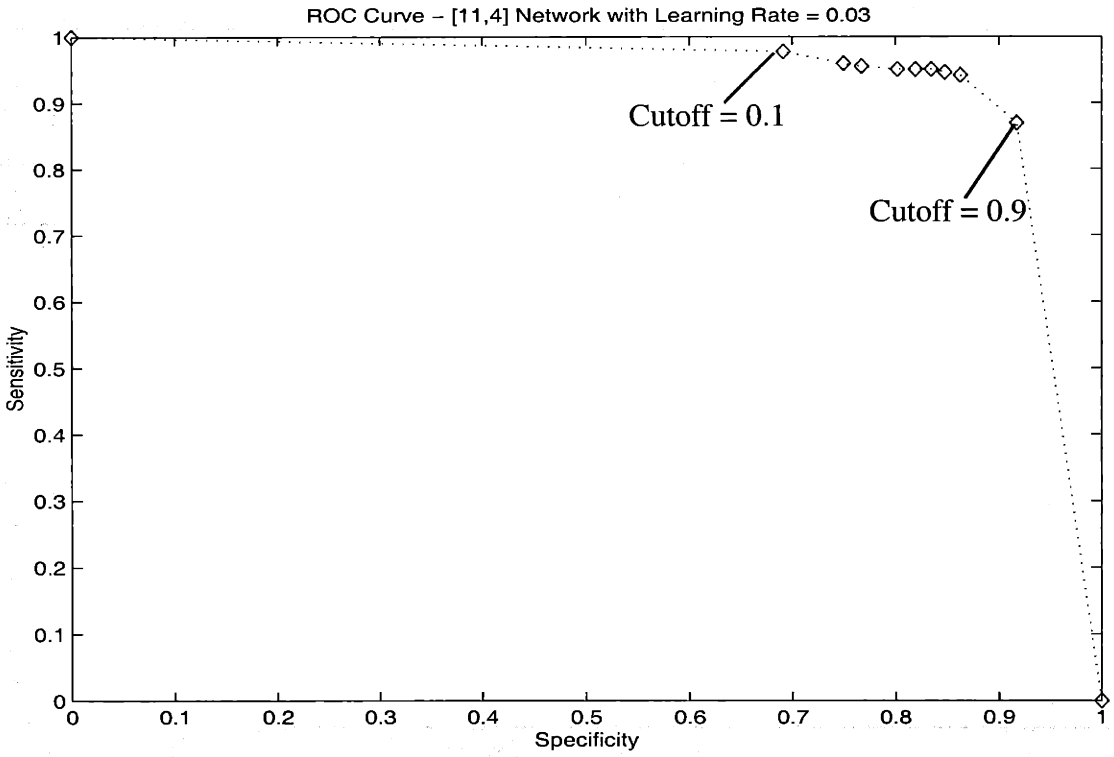
Additional viable data was obtained from studies for this phase of the research. The viable CMAP set included over 260 signals. This table does not include a MSE column due to its insignificance in the network evaluation process. Instead, there are columns for the initial learning rate value and cutoff value for each network. There are many redundant network topologies because either one of these parameters is varied for each network. Another column that is excluded from Table 3.2 is the number of signals in the training set. In this series of networks, the training set was kept constant at 40 viable and 40 non-viable CMAPs. However, the sizes testing data sets were dramatically increased. The viable CMAP test set consisted of 222 signals and the non-viable test set was composed of 460 noise signals. The validation sets were large in order to thoroughly test the robustness of the networks.

The learning rate was an important characteristic to keep track of for this set of networks because the training algorithm was DX. Since DX varies the learning rate throughout the training process, it could be hypothesized that the initial learning rate could have an effect on the outcome of the neural network. However, from the results shown above, there is no correlation between setting a particular initial learning rate and better performance of the network. This means that the learning rate varies so much during training that it didn't really matter what the initial learning rate was.

The cutoff value was noted for each network trained. From the data, it could be gathered that as the cutoff was changed, trade-offs had to be made between sensitivity and specificity values. For example, rows five and six (with the [11,4] sized hidden layers) compare the same neural network, but with different cutoff values. The network from row five produced a slightly better sensitivity, whereas the network from row six gave a higher specificity. This showed that the same network can be used to prioritize either sensitivity or specificity based on which cutoff value is chosen. This was also evident from the ROC

curve produced by this particular network (See Figure 3.1). This curve plots specificity versus sensitivity for all cutoff values and gives a detailed visual representation of the ROC analysis.

Figure 3.2: ROC Curve (Each diamond represents a cutoff value from 0-1)



The overall conclusion that could be drawn from this set of results was that most of these networks provided adequate performance. It was just a matter of which topology was preferred over another (this will be discussed further in the Chapter 4). Also, the hypothesis from the earlier phase that more non-viable signals in the training set will lead to better performance was proved wrong with these findings.

In the next series of networks, the size of the training set was increased from 80 waveforms to 120 (60 viable and 60 non-viable). This experiment was performed to explore the possibility of better performance due to an increase in the number of total training signals. The results are shown in Table 3.3.

# of Neurons in Hidden Layers	Learning Rate	Cutoff Value	Sensitivity	Specificity
11,6	0.02	0.6	93	90
11,6	0.02	0.7	93	90
11,6	0.02	0.8	87	94
11,6	0.03	0.4	94	84
11,6	0.03	0.5	94	87
11,6	0.03	0.6	92	89
11,6	0.03	0.7	91	91
12,5	0.03	0.5	92	87
12,5	0.03	0.6	91	89
12,5	0.03	0.7	89	93

Table 3.3: Phase Two Results

Only a few topologies of the networks were trained in this session. It was clear that increasing the training set did not have a significant positive effect on the sensitivity and specificity values. In fact, this seemed to decrease the values in some cases. Since the smaller training sets required less time to train, it was more beneficial to have a smaller training set.

3.3.2 Backpropagation Networks with Principal Components Analysis

Up to this point in the research, no preprocessing techniques had been applied to the inputs (CMAPs) of the neural network. Since each CMAP consists of 128 elements, it was reasonable to assume that some sort of processing could reduce this size, while maintaining the important traits of the waveform.

As mentioned in section 2.2, principal components analysis (PCA) is a popular technique used to reduce the dimensionality of vectors. This analysis transforms the input data so that the elements of the input vectors will be uncorrelated. In addition, the size of the input vectors may be reduced by retaining only those components which contribute more than a specified fraction of the total variation of the data set [3]. In order to perform a correct PCA analysis, the entire set of data must be normalized such that the mean of zero and a standard deviation of one. Once the normalization is completed, a fraction must be specified to limit the number of principal components.

The routine in MATLAB which performs the PCA analysis uses singular value decomposition to compute the principal components. The input vectors are multiplied by a matrix whose rows consist of the eigenvectors of the input covariance matrix. This produces transformed input vectors whose components are uncorrelated and ordered according to the magnitude of their variance. Those components which contribute only a small amount to the total variance in the data set are eliminated [3]. However, MATLAB places a restriction on the data that must be processes (the training set). The number of signals must be greater than the number of elements in the signals. Hence, the training set size had to be greater than 128.

Principal components analysis was implemented in the next series of experiments as preprocessing for the training set. The only parameter for this processing was the variation percentage, or the number principal components to be calculated for the data set. As this percentage was varied, the number of components fluctuated. For example, processing the training set (composed of 70 viable and 70 non-viable CMAPs) with a 95 percent retention of the variance of the data (or only those components which contribute more than five

percent to the variance in the data set) produced six components. Therefore, the dimensionality of the input to the neural network could potentially be reduced from 128 elements to only six elements. This represented a significant reduction and added the possibility of PCA to aid in the classification system.

Table 3.4 presents the results from neural networks trained with preprocessed input data.using various values of principal components.

Components (%)	# of Neurons	Cutoff	Sensitivity	Specificity
6 (95 %)	0	0.9	0.88	0.89
6	1	0.9	0.88	0.89
6	3	0.5	0.90	0.93
6	4	0.9	0.94	0.93
6	5	0.8	0.93	0.92
5 (93 %)	0	0.8	0.88	0.90
5	1	0.8	0.88	0.90
5	2	0.8	0.90	0.87
5	4	0.6	0.94	0.94
5	5	0.7	0.91	0.91
4 (91 %)	0	0.5	0.90	0.87
4	1	0.7	0.89	0.87
4	2	0.5	0.94	0.78
4	3	0.7	0.91	0.95
4	4	0.5	0.92	0.95
4	5	0.6	0.94	0.95
3 (89 %)	5	0.5	0.83	0.86
3	6	0.4	0.88	0.90

Table 3.4: Phase Two Results

The first column represents the number components in the input to the network along with the corresponding percentage used in the PCA calculations. All of the networks only contained one hidden layer, whose size is noted in the second column of the table. For each network, only one cutoff value is listed because it was chosen as the most optimal cutoff in terms of sensitivity and specificity values.

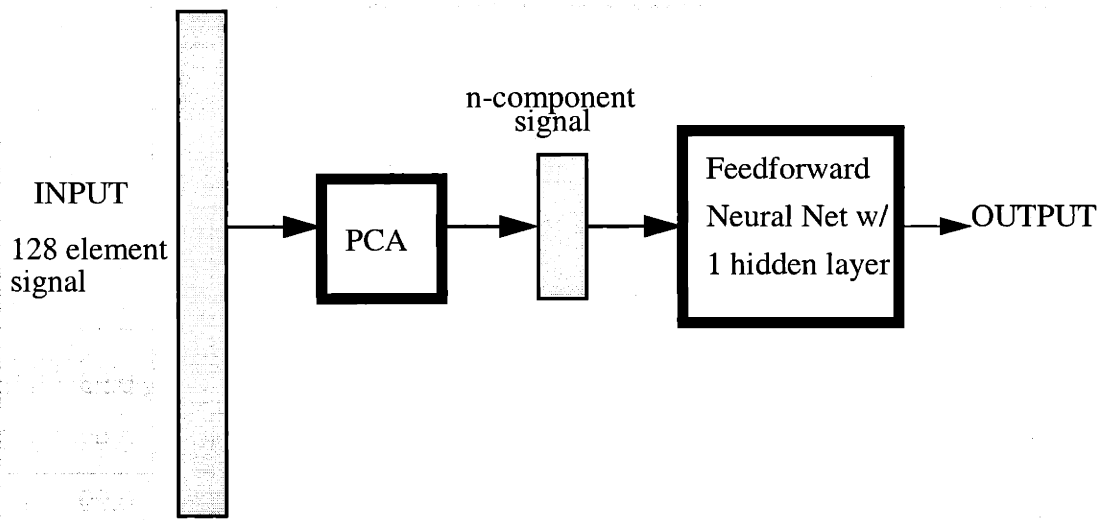


Figure 3.3: Diagram of Neural Nets with PCA experiments

The results from this series of experiments was astonishing. The sensitivity and specificity values for all the networks trained matched or beat the networks with no preprocessed inputs. Both the sensitivity and specificity values reached the mid-nineties, a feat yet to be accomplished by the earlier experiments. Only one hidden layer was used in the neural network because it was hypothesized that the principal components analysis would basically "act" as a layer in the network. However, the PCA seemed to perform much better than a layer by comparing these results to the earlier results. It could also be seen that as the number of components decreased from the PCA, the performance of the network also steadily decreased. This is confirmed by the fact that PCA usually becomes less effec-

tive on the data as the number of components decreases. (Additional discussion is provided in Chapter 4).

3.3.3 Unsupervised Neural Networks

As a additional experiment, unsupervised neural networks were trained with this data to compare against the supervised training methods (please see section 2.1 for a discussion on supervised vs. unsupervised training). It has been noted in academia that unsupervised training tends to perform better than supervised training. All of the previous results were accumulated from supervised training. Unsupervised learning is also known as competitive learning.

The basic way to train an unsupervised network is to present the training data without any target values. This allows the network itself to classify the signals the best way it seems fit. The only parameter to be specified is the number of categories for the data to be separated into. The training set was kept constant at 60 viable and 60 non-viable CMAPs. Here are the results with two categories

	Category 1	Category 2
Training Set	57	3
Test Set	187	15

Table 3.5: Viable CMAPs with 2 Categories

	Category 1	Category 2
Training Set	16	44
Test Set	119	321

Table 3.6: Non-Viable CMAPs with 2 Categories

Table 3.5 presents the results from the viable CMAP data sets. This showed that the network could recognize a viable CMAP and categorized most of them into the first category. However, the findings from the next table seemed to show that this was not exactly true. The network also classified many of the non-viable CMAPs into the first category. This result, although not disastrous, left much doubt that unsupervised training could outperform supervised training.

Training with three and four categories was also explored during this analysis:

	Category 1	Category 2	Category 3
Training Set	7	53	0
Test Set	35	162	5

Table 3.7: Viable CMAPs with 3 Categories

	Category 1	Category 2	Category 3
Training Set	19	10	31
Test Set	133	76	231

Table 3.8: Non-Viable CMAPs with 3 Categories

	Category 1	Category 2	Category 3	Category 4
Training Set	0	51	9	0
Test Set	4	142	53	3

Table 3.9: Viable CMAPs with 4 Categories

	Category 1	Category 2	Category 3	Category 4
Training Set	23	8	7	22
Test Set	146	54	63	177

Table 3.10: Non-Viable CMAPs with 4 Categories

These numbers also supported the data from the previous page that the classification scheme works well for the viable waveforms, but performs inadequately for the non-viable signals. The unsupervised training was performed to provide a comparison for the supervised training and also to provide more insight into the training.

Chapter 4

Discussion and Conclusion

4.1 Analysis of Results

From the results summarized in the previous chapter, it can be seen that many adequate solutions have been proposed to the classification of compound muscle action potentials. However, the main question that must be answered is which solution provides the best and most efficient performance?

4.1.1 PCA versus no PCA

The critical issue in determining an optimal solution is whether preprocessing using principal components is advantageous for the performance of the neural network. This issue requires a more in-depth analysis of both solutions and other factors, such as real-time implementation, must be heavily considered.

In order to make a true comparison between the two methods, specific networks must be chosen from both techniques (Tables 3.2, 3.4). The network chosen from the non-PCA technique (Table 3.2) contains two hidden layers with 12 and 6 neurons, respectively. The network chosen with PCA preprocessing (Table 3.4) contains an input of 5 components and one hidden layer of 4 neurons. Both of these networks perform very well with the test data. The network with PCA performed slightly better (compared with only sensitivity and specificity values) than the other net (See Figures 4.1 and 4.2 for a comparison of the respective ROC curves).

One way to compare these two networks is to evaluate how much work the PCA is actually performing compared to one hidden layer of a neural network. This analysis can

be done by creating a network that has the same structure as the network utilizing the PCA, but performing no preprocessing at all. The results are shown in Table 4.1.

# neurons in 1st layer	# neurons in 2nd layer	Cutoff	Sensitivity	Specificity
5	4	0.6	0.64	0.69
5	5	0.5	0.65	0.60
6	3	0.6	0.80	0.81
6	4	0.7	0.77	0.72

Table 4.1: Neural Networks with no Preprocessing

From the results in the previous table, one can see that the principal components analysis actually executes a more thorough reduction of the input vectors than a normal hidden layer of a neural network. The PCA is extracting more information out of the input than a hidden layer of equal size in the number of neurons.

There is another advantage to using a network with PCA preprocessing. The input to the network is much smaller. The network without preprocessing consists of an input of 128 elements, which is significantly larger than only six inputs. However, this advantage comes with the cost of actually performing the principal component analysis to each input of the network.

An important comparison that must be made is the number of floating point operations (FLOPs) that are performed by each network. This is vital to the implementation of the network on a real-time system because it dramatically affects the evaluation and processing time. Since the network that utilizes the PCA is much smaller in terms of connections and operations, it is obvious that this network will compute the least number of FLOPs when classifying a waveform. To be precise, there is a 98.4 percent reduction (24 vs.

1536) in the number of FLOPs when using PCA. This translates to an enormous decrease in the processing time for the neural network, with no loss in performance.

Overall, the network that utilizes the PCA tends to have many more advantages than the network with no preprocessing. It performs much less FLOPs and gives a better performance rating in terms of sensitivity and specificity values. Even though there is some overhead for performing the PCA on the input vector, it is minimal compared to the advantage in performance that is gained.

4.2 Implementation

Due to the advantages discussed above, the network employing the PCA was chosen for the implementation onto the NC-stat device. The chip on the device has some limitations, such as 4K RAM and 128K code-space. However, this is enough space to write C code which simulates this particular neural network (Please see Appendix A). The input to the code is a 128 component vector and the output was a 1 or 0, specifying whether the signal is viable or not. All parameters, such as cutoff value, PCA matrix values, and neural network weights are constants which can be easily altered for future networks. The code performs the PCA on the input and processes the resulting vector with the neural network to produce an output.

Even though significant results have not been collected for the implementation at this point, early results indicate that the implementation is performing quite well with real-time data. The early results show that the system correctly classifies viable waveforms at about 100 percent. However, there is not enough data to evaluate the performance of this system on non-viable waveforms. These results indicate that implementation of this sys-

tem on the processor can provide an accurate, real-time quality control mechanism to detect non-viable CMAPs.

4.3 Conclusions

The research and experiments performed for this thesis project have uncovered significant results that have not been documented in neural network classification systems for these types of waveforms. These conclusions can be summarized as follows:

- Backpropagation neural networks provide more than satisfactory performance in the classification of compound muscle action potentials.
- Principal component analysis is a substantial preprocessing technique which significantly increases the performance of the classification system.
- This system, which employs PCA along with the neural network processing is a highly robust and implementable module which can serve as an dependable quality assurance mechanism.

These three conclusions are realized and supported by the results found in this thesis project. As a whole, these will benefit future studies in classification systems and provide for an accurate, reliable, and efficient real-time quality assurance for CMAPs on the NC-stat nerve conduction monitoring system.

Appendix A

Implementation of Neural Network System in C

```
#include <math.h>
#include <stdio.h>

#define Components 4
#define Neurons 5
#define CutOff 0.6

void main()
{
float waveform[128] = { }; /* input signal */
float TransMat[Components][128] = { }; /* transformation matrix for PCA */
const float Weights1[Neurons][Components] = {-0.6569,-0.1151,0.7364,0.1997,-
0.3516,0.1908,0.3209,-3.9678,-0.1910,-0.2028,0.1445,0.0481,-0.7301,0.4180,0.3342,-
3.4112,-1.2722,0.6372,0.6512,-0.2259}; /* Network weights for layer 1 */
const float Weights2[Neurons] = {-4.3996,5.6339,-0.9494,-5.5742,-0.3925};
/* Network weights for layer 2 */
const float Biases[Neurons] = {4.3907,3.2281,1.0915,-1.8211,-2.0014};
/* Bias values for layer 1 */
const float LastBias = -0.6951; /*Bias value for layer 2 */
const float Means[128] = { }; /* normalization values */
const float Stdevs[128] = { };

int i,j;
float output,f1;
float trans[Components];
float layer1[Neurons];

/* normalize the input */
for (i = 0; i < 128; i++) {
 waveform[i] = (waveform[i] - Means[i]) / Stdevs[i];
}

/* transform input using PCA */
for (i = 0; i < Components; i++) {
 f1 = 0;
```

```
 for (j = 0; j < 128; j++) {
  f1 = f1 + TransMat[i][j] * waveform[j];
 }
 trans[i] = f1;
}

/* process with the neural network */
for (i = 0; i < Neurons; i++) {
 for (j = 0; j < Components; j++) {
  layer1[i] = layer1[i] + Weights1[i][j] * trans[j];
 }
 layer1[i] = layer1[i] + Biases[i];
 layer1[i] = 2 / ( 1 + exp(-2 * layer1[i])) - 1;
 output = output + layer1[i] * Weights2[i];
}

/* compute output */
output = 1 / (1 + exp(-1 * (output + LastBias)));
printf("%6.4f",output);

}
```

References

[1] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, New York, 1994.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[3] Mark Beale and Howard DeMuth. *MATLAB Neural Network Toolbox Users Guide.* Version 3, 1998.

[4] G.J. Chappell, J. Lee, and J.G. Taylor. *A Review of Medical Diagnositic Applications of Neural Networks*, Neural Network Applications, pp.23-26, 1991.

[5] Yu Hen Hu, Vijay Mani, Surekha Palreddy, and Willis J. Tompkins. *A Multiple-Classifier Architecture for ECG beat classification*, Neural Networks for Signal Processing VII, pp.172-181, 1997.

[6] Jose R. Dorronsoro, S. Gonzalez, Vicente Lopez, and Juan A. Siguenza. *Automatic Classification of Visual Evoked Potentials by Feedforward Neural Networks*, Artificial Neural Networks Vol. 2, pp.1117-1120, 1991.