Determining Mental State from EEG Signals Using Parallel Implementations of Neural Networks

Charles W. Anderson
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
anderson@cs.colostate.edu

Saikumar V. Devulapalli
Department of Computer Science
Colorado State University
Fort Collins, CO 80523

Erik A. Stolz
Department of Electrical Engineering
Colorado State University
Fort Collins, CO 80523

Abstract

EEG analysis has played a key role in the modeling of the brain's cortical dynamics, but relatively little effort has been devoted to developing EEG as a limited means of communication. If several mental states can be reliably distinguished by recognizing patterns in EEG, then a paralyzed person could communicate to a device like a wheelchair by composing sequences of these mental states. EEG pattern recognition is a difficult problem and hinges on the success of finding representations of the EEG signals in which the patterns can be distinguished. In this article, we report on a study comparing three EEG representations, the unprocessed signals, a reduced-dimensional representation using the Karhunen-Loève transform, and a frequency-based representation. Classification is performed with a two-layer neural network implemented on a CNAPS server (128 processor, SIMD architecture) by Adaptive Solutions, Inc.. Execution time comparisons show over a hundred-fold speed up over a Sun Sparc 10. The best classification accuracy on untrained samples is 73% using the frequency-based representation.

1. Introduction

A physically-disabled person who has no control over their motor responses has no means of communicating to the outside world. Is there a way for such a person to use their mental capabilities to affect their environment? This question drives the search for patterns in EEG signals that are related to a person's mental state. A device that can reliably and quickly discriminate between several mental states could be used, for example, to generate commands to control a wheelchair.

Computerized analysis of EEG signals has evolved over the past three decades [Gev87], with much of the effort directed towards a better understanding of the functioning of the brain. The work reported here has a different goal, to extract information from EEG signals with which we can discriminate mental states. Primarily two approaches have been taken towards this goal.

The first approach is based on the discovery that a characteristic signal appears in the EEG approximately 300 ms following the occurrence of a relatively rare, but expected, stimulus. Such signals are referred to as event related potentials, or ERP's. An example of how ERP's can be used to communicate with a computer is the work of Farwell and Donchin [FD88], who used

ERP's to detect which letter of the alphabet a human subject wished to select. The letters of the alphabet were displayed in a matrix form on the visual display of a computer. The subject selected a row and column by observing a marker step through the rows and columns. When the correct row and column was marked the subject's EEG contained a recognizable ERP.

This kind of interaction between a person and stimulus device would be too cumbersome and probably too slow to be useful in a real-time control application. Much more practical would be a system for detecting patterns in normal EEG without the aid of an external stimulus device.

This is the second approach, often referred to as spatial analysis, because patterns are sought in EEG signals simultaneously recorded from multiple electrodes. A number of studies have found differences in the power of the alpha band (8-13 Hz) in signals recorded from left and right hemispheres, depending on the tasks [GO73, DOG74, KMLW84]. Asymmetries were most reliably found for motor tasks. Ehrlichman and Wiener [EW80] found verbal tasks produced greater hemispheric differences than did visual tasks.

The detection of patterns in EEG produced from normal mental states is a very difficult problem. EEG signals are recorded by surface electrodes and can contain noise as a result of electrical interference and movement of the electrodes on the scalp. Another problem is that EEG can be corrupted by eye blinks and other muscular activity that produce signals of greater magnitude than produced by cortical activity. Other problems are more cognitive in nature. For example, the concentration of a person can vary while the person is supposedly performing a single mental task.

The work described in this article is based on previous work by Keirn and Aunon [KA90, Kei88]. Keirn and Aunon recorded EEG from seven subjects while the subjects performed each of five mental tasks. A simple Bayesian classifier was applied to data collected from pairs of tasks. A frequency-based representation was found to result in 70% to 100% correct classification.

This is an encouraging result, but the study was limited in several ways. A single quarter-second or two-second segment was selected from each 10 second recording session. The segment was chosen to be devoid of eye blinks and near the middle of the session, assuming during that period the subject was most likely concentrating on the requested mental task. Another limitation is the use of a quadratic Bayesian classifier, which assumes the classes have a Gaussian distribution and thus would not be able to represent complex, nonlinear relationships. Also, in the previous study classifiers were constructed and tested on data from single subjects and pairs of tasks. Questions remain regarding generalization across subjects and more than pair-wise discriminations. This is not addressed here.

These questions led us to extend the previous study in three ways. We

- replaced the Bayesian classifier with a neural network and varied the number of hidden units, thus the complexity of the classifier;
- extracted overlapping quarter second segments that together cover the 10 second period of every recording session (no artifacts were removed);
- compared the classification accuracy of the neural network using different representations of the EEG signals.

The objective of these experiments was to determine which of the three representations results in the best classification accuracy. If the information needed to discriminate mental state can be extracted from the unprocessed EEG signals, or at most preprocessed by projecting to a relatively small number of principal components, then we can dispense with other forms of preprocessing, such as the frequency analysis used by [KA90].

In one other study, by Lin, Tsai, and Liou [LTL93], neural networks were applied to classify EEG signals collected in a paradigm similar to that of [KA90]. Lin, et al., used Kohonen's algorithm [Koh84] to train a matrix of units to identify clusters of similar patterns and associate each cluster with a particular mental task. They trained their classifier on data for all tasks performed by one subject in one recording session and tested the resulting classifier on data from other sessions and other subjects. Most tests showed very poor classification accuracy, though the accuracy tended to be higher for some tasks, particularly the mental arithmetic task (see Section 2.1).

The remaining sections are organized as follows. In Section 2, we describe the methods and algorithms used to collect, process, and classify the data. The implementations of the processing and classification algorithms are described in Section 3. Results of classification experiments are discussed in Section 4, and conclusions are presented in Section 5.

2. Data Collection, Preprocessing, and Neural Network Algorithm

2.1 Mental Tasks

All data used in this study were recorded previously by Keirn and Aunon. Their selection of the set of mental tasks was guided by Galin and Ornstien [GO73] whose results showed detectable hemispheric differences in some tasks. The following tasks were studied in [KA90]:

Baseline—Alpha Wave Production The subject is asked to open and close their eyes at approximately five second intervals. With their eyes closed the subject is to relax as much as possible. This is considered the baseline session for alpha wave production, and other asymmetries. Non-task associated levels of alpha wave production and asymmetries produced across different electrodes and across EEG bands are thus obtained.

Mental Arithmetic The subject is given a non-trivial multiplication problem to solve and, as in all of the tasks, is instructed not to vocalize or make overt movements while solving the problem. An example of such a task is to multiply the numbers 49 times 78. The problems are non-repeating and are designed so that an immediate answer will not apparent. The subject verifies at the end of the task whether or not they arrived at a solution.

Geometric Figure Rotation The subject is given 30 seconds to study a drawing of a complex three dimensional block figure after which the drawing is removed and the subject instructed to visualize the object being rotated about an axis.

Mental Letter Composing The subject is instructed to mentally compose a letter to a friend or relative without vocalizing. Since the task is repeated several times the subject will be told to try to pick up where they left off in the previous task.

Visual Counting The subject is asked to imagine a blackboard and to visualize numbers being written on the board sequentially, with the previous number being erased before the next number is written. The subjects is further instructed not to verbally read the numbers but to visualize them, and to pick up counting from the previous task rather than starting over each time.

The experiments reported here used only the data recorded from a single subject performing the baseline task and the mental arithmetic task.

2.2 Recording of EEG Signals

Subjects were seated in a sound-proof, dimly-lit, room. As shown in Figure 1, electrodes were placed at O_1, O_2, P_3, P_4, C_3, and C_4, standard electrode locations in the 10-20 System [Jas88]. The electrodes were connected to Grass 7P511 amplifiers that bandpass filtered the signals at

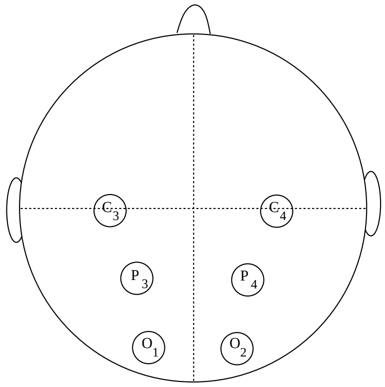

Figure 1: Location of the six surface electrodes used to record EEG signals.

0.1-100 Hz. The EEG signals were sampled at 250 samples per second and digitized with 12 bits of accuracy. Data was recorded from each subject for a duration of 10 seconds while the subject was performing a single task with their eyes open. Each session resulted in 250 samples/second x 10 seconds x 6 channels, or 15,000 values.

2.3 Unprocessed Data Representation

Keirn and Aunon found that quarter-second segments of the 10 second data resulted in classification accuracy approximately the same as that obtained from two-second segments. Therefore, for the experiments reported here, we divided the data into quarter-second segments. Segments were actually 62 samples long, slightly less than one quarter second. Ideally, we would like to use all quarter-second segments to train the classifier. This would result in 2,438 (i.e., 2,500-62) overlapping segments, each offset from the previous one by one sample. To make this more tractable, we divided the data into segments that overlap by an eighth of a second, resulting in 79 segments (40 non-overlapping + 39 overlapping) shown in Figure 2. Data from each segment becomes one pattern, so each pattern is a 62 x 6, or 372, component vector. We collected 79 such patterns from one subject performing the baseline and mental arithmetic tasks 10 times each, for a total of 1,580 patterns. Repetitions were performed by the subject in two recording sessions on different days.

Examples of actual recordings are shown in Figure 3. The top two graphs show the first and last quarter-second segment from a subject performing the baseline task. The bottom two graphs are data from the same subject doing the mental arithmetic task. There are no obvious features that differentiate the top graphs from the bottom graphs.

2.4 K-L Representation

It is possible that the large number of components in each pattern is many more than actually needed. Often, when classifying high-dimensional data, equivalent accuracy can be achieved by classifying data obtained by projecting the original data onto the first n eigenvectors, where n is much smaller than the dimensionality of the original data. This is usually referred to as the Karhunen-Loève decomposition, called the K-L representation here.

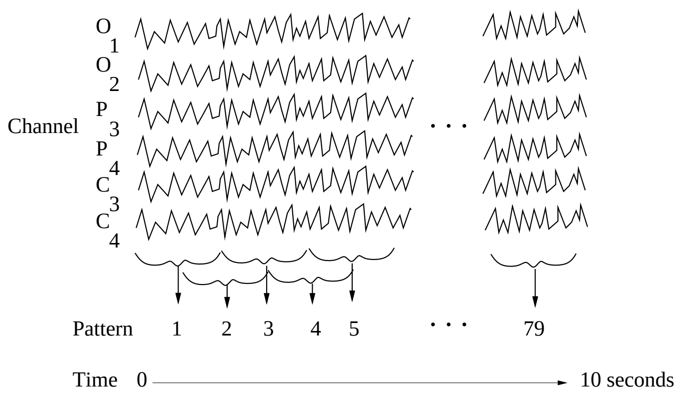

Figure 2: Division of data into quarter-second, overlapping windows from one subject doing one task for 10 seconds.

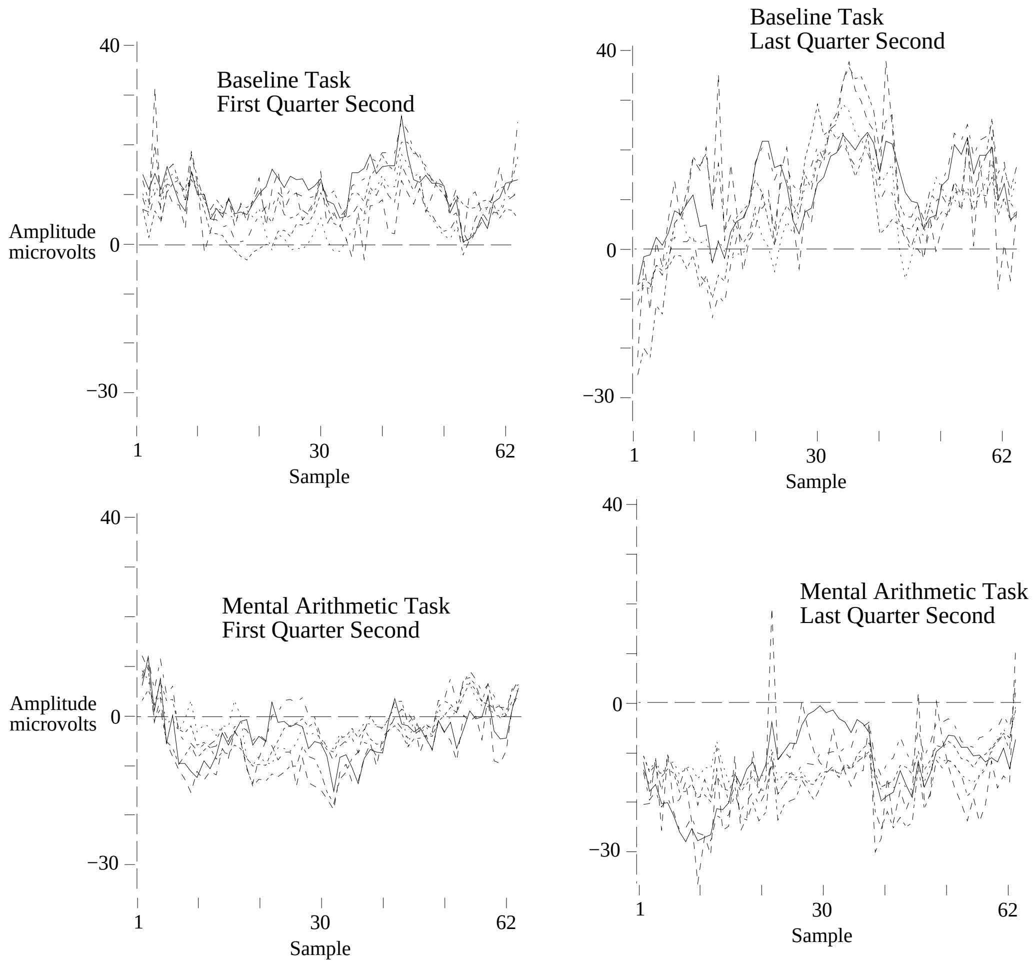

Figure 3: EEG signals recorded from six channels as subject performed the baseline or mental arithmetic tasks.

To perform the K-L decomposition, the covariance matrix of the mean-subtracted set of 1,580 372-dimensional patterns was calculated. The eigenvalues and eigenvectors of the covariance matrix were calculated using the Jacobi method. This involves performing a sequence of similarity transformations in order to reduce the off-diagonal elements to zero. A series of plane rotations are applied, each one annihilating one of the off-diagonal elements. Successive transformations result in non-zero values at these elements, but the process ultimately converges and the result is a diagonal matrix. The matrix of eigenvectors is the product of the transformation matrices, and the eigenvalues are the elements of the final diagonal matrix.

The first four eigenvectors for the combined baseline and mental arithmetic task data are graphed in Figure 4 as six curves corresponding to the six electrode channels.

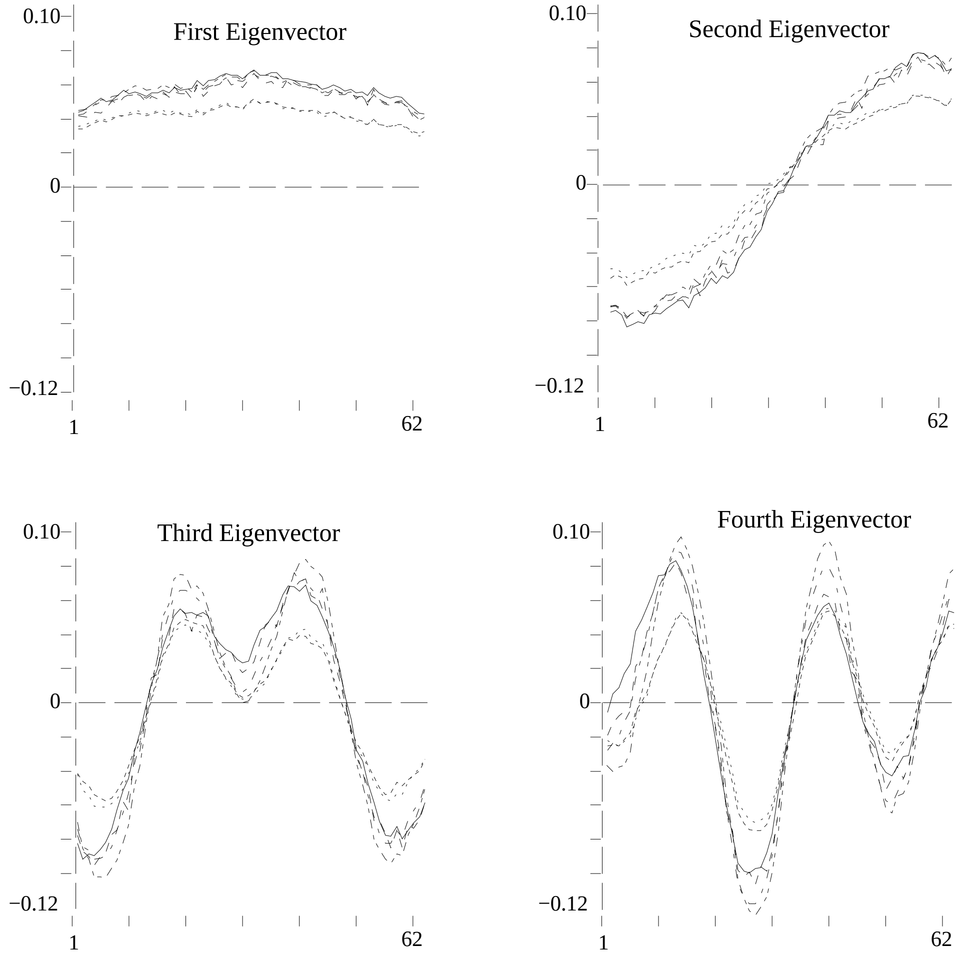

Figure 4: First four eigenvectors of mean-subtracted data for baseline and mental arithmetic tasks.

The number of eigenvectors to project to can be chosen empirically or determined by examining the eigenvalues. One estimate of dimensionality is the global Karhunen-Loève estimate, given by the index i for which $\lambda_i/\lambda_{max} \leq 0.01$, where the λ_i are the eigenvalues in decreasing order for $i = 1, 2, \ldots$. For the baseline and mental arithmetic data used in our experiments, $i =$ 50. Figure 5 shows the first 50 eigenvalues normalized by the total sum of the 372 eigenvalues.

The K-L representation was formed by projecting each 372-dimensional pattern onto the first 50 eigenvalues. Examples of patterns in this representation that correspond to the raw

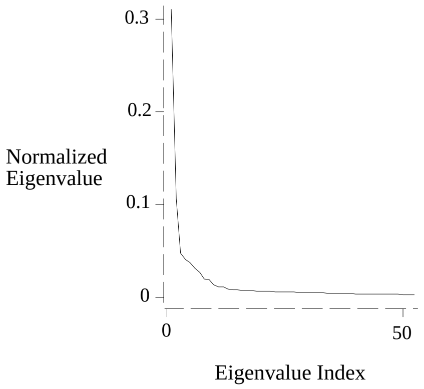

Figure 5: First 50 eigenvalues for mean-subtracted data from subject 3 for baseline and math tasks.

patterns in Figure 3 are shown in Figure 6.

2.5 Frequency-Band Representation

In [KA90] features were extracted from spectral density estimates using asymmetry ratios given by $(R - L)/(R + L)$, where R is the area under the spectral density curve of a right hemisphere channel for a specific frequency band and L is defined similarly for the corresponding left hemisphere channel. These asymmetry ratios were calculated for each possible right-to-left combination of channels and for each of four frequency bands: delta (0-3 Hz), theta (4-7 Hz), alpha (8-13 Hz), and beta (14-20 Hz). This resulted in 36 asymmetry ratios. In addition the 24 power values themselves (R and L) were added for a total of 60 features.

In the current study, the spectral density was estimated from autoregressive (AR) parameters calculated using the Burg method [PM92]. This method is based on the minimization of forward and backward linear prediction errors, subject to satisfying the Levinson-Durbin recursions. The spectral density is given by

$$P(f) = 2\sigma^2/[1 - \sum_{n=1}^{M} \alpha_n e^{-j2\pi nf}]^2$$

where α_n are the estimated AR coefficients. An AR model of order 6 ($M = 6$) was used here since it yielded good results in [KA90].

Example patterns in the frequency-based representation are shown in Figure 7. These were derived from the same data as the graphs in Figures 3 and 6.

2.6 Neural Network Classification Algorithm

In their previous work, Keirn and Aunon used a quadratic Bayesian classifier. This Bayesian classifier assumes a Gaussian shape to the probability density function of data from each class and a linear discrimination between these density functions is determined. This classification technique is insufficient accurate classification requires a nonlinear combination of the pattern components.

Neural networks have the capability of finding a nonlinear transformation of the pattern in order to classify more accurately. However, the increased complexity of a neural network can result in large computation times to train the network. The quantity of data involved in our experiments prompted out implementation of the neural networks on a SIMD parallel computer, described in the next section.

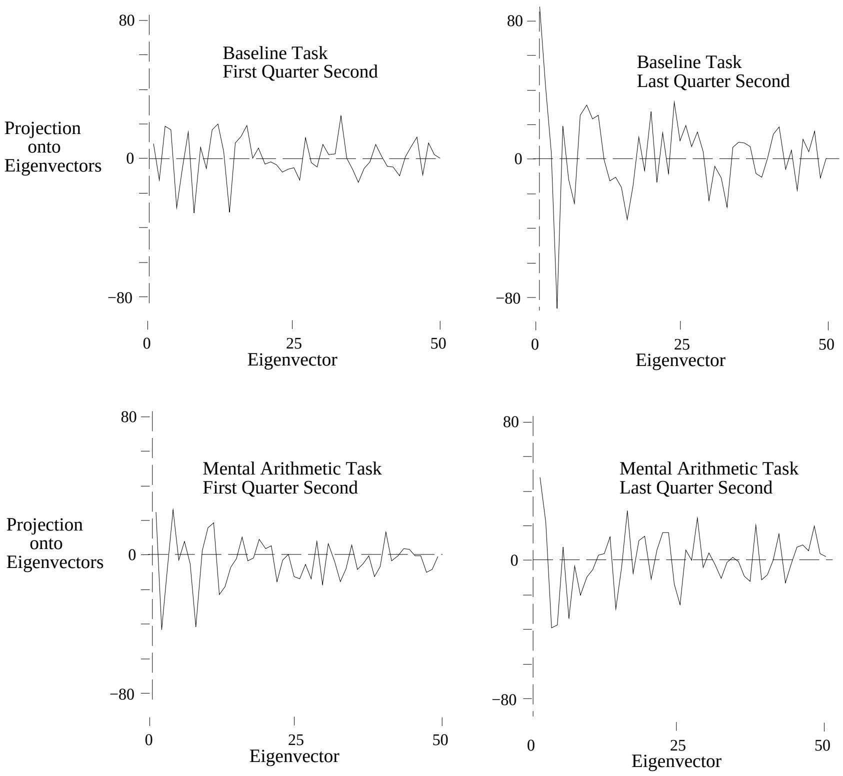

Figure 6: K-L representations of the first and last quarter-second of data from the baseline task (top graphs) and the mental arithmetic task (bottom graph).

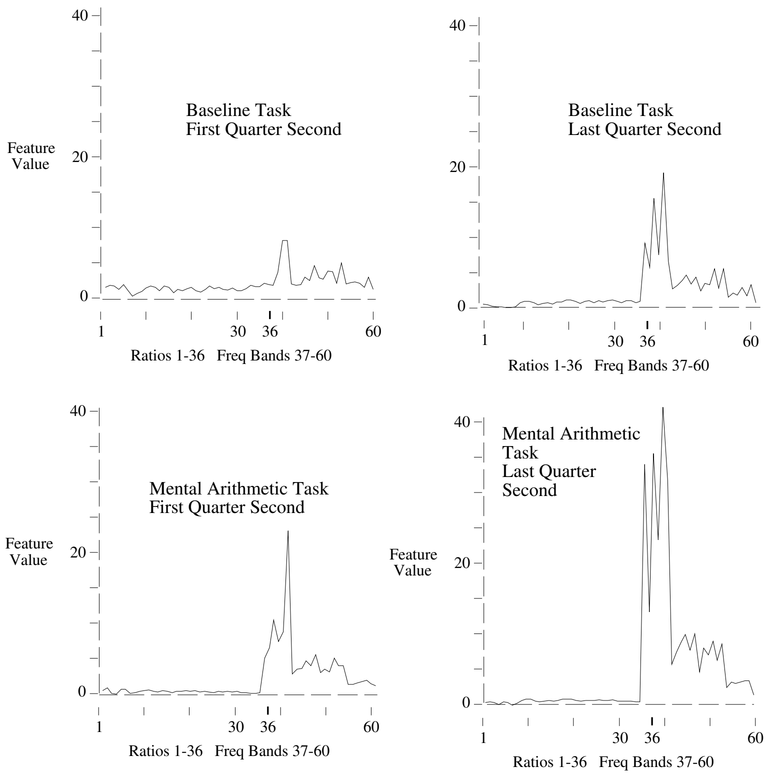

Figure 7: Frequency-based representations of the first and last quarter-second of data from the baseline task (top graphs) and the mental arithmetic task (bottom graph).

The network architecture used for our experiments is shown in Figure 8. The circles rep-

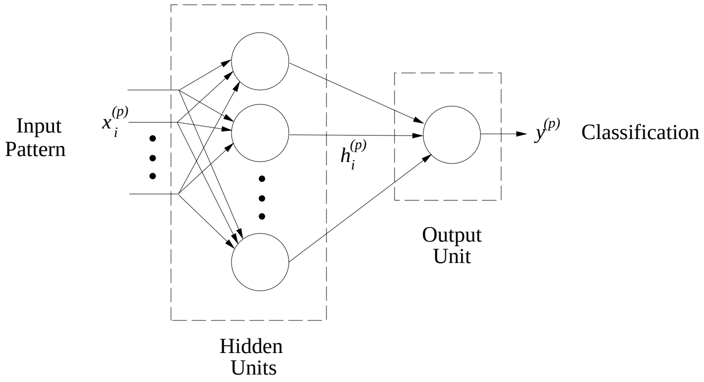

Figure 8: Two-layered neural network used in all experiments reported here. Only the number of hidden units and the input representation was varied.

resent the computational units of the network. The interconnections represent scalar values passed as input to each unit. Each unit has a unique vector of weights corresponding to its input vector. The computation performed by the units is a weighted sum of their inputs and a nonlinear *squashing* function that restricts the range of the output to be between 0 and 1. We used the typical *sigmoid* squashing function. Let the inputs to a unit be x_i, the weights be w_i, and the output of the unit be y. The weighted sum and sigmoid function are combined to produce the unit's output:

$$y = \frac{1}{1 + e^{-\sum_i x_i w_i}}.$$

The network consists of two layers of units. The units in the first layer are called *hidden units*, because the outputs of these units are used internal to the network to transform the input into another representation for the *output unit*. The output of the output unit is taken as the classification of the current input pattern.

To train the network, a set of training patterns and corresponding correct outputs is repetitively presented to the network. After each pass through the training data, called an *epoch*, the weights are adjusted to reduce the error between the correct output and the actual output of the network. To determine how to adjust each weight, we applied the error backpropagation algorithm [RHW86]. This algorithm calculates the gradient of an error function with respect to each weight, then adjusts the weights in the negative gradient direction to reduce the error. The error function is the squared error summed over all training patterns:

$$E = \sum_p (z^{(p)} - y^{(p)})^2$$

where p is an index into the set of training patterns, $z^{(p)}$ is the correct output for pattern p, and $y^{(p)}$ is the output of the network for pattern p. For the experiments reported here, $z^{(p)}$ is set to 0.1 for the baseline task and 0.9 for the mental arithmetic task. This determines how we interpret the output of the network as a classification: if $y^{(p)} > 0.5$, pattern p is said to be classified as being from a mental arithmetic task; if $y^{(p)} < 0.5$, the pattern is classified as a baseline task.

The gradient of E with respect to the weights results in the following expressions. Let $h_i^{(p)}$ signify the output of hidden unit i when the network receives input pattern p. To change the weights of the output unit, we sum the following $\Delta w_i^{(p)}$'s

$$\Delta w_i^{(p)} = c(z^{(p)} - y^{(p)})h_i^{(p)}$$

over all training patterns and, at the end of the epoch, add the result to the weights in the output unit. The constant c is a scale factor that is chosen empirically to produce weight changes that are not too large nor too small. If too large, the network would converge quickly to a suboptimal local minimum; if too small, the time required for the network to converge would be impractically long. Similarly, we sum the $\Delta w_{ij}^{(p)}$'s for hidden unit j, given by

$$\Delta w_{ij}^{(p)} = c_h(z^{(p)} - y^{(p)})h_j^{(p)}(1 - h_j^{(p)})x_i^{(p)}$$

where $x_i^{(p)}$ is a component of the input pattern give to the network. The hidden units have their own scale factor, c_h.

2.7 The Overfitting Problem

Although this algorithm is designed to minimize the squared error over a training set, the true goal of this procedure is to find a set of weights for which the squared error is minimized over a novel set of data, i.e., data to which the network was not trained. Only if the algorithm is able to find a model (the network and weights) that generalizes well to this *test set* can we say that the model captures the regularities present in the data. This problem has been called the overfitting problem—the network too closely matches the training data and does not interpolate and extrapolate well to novel data. The is usually tested by dividing the data into a *training* and a *testing* set. The network is trained to convergence on the training set, after which the error on the testing set is calculated and used as an estimate of how well the network will perform on novel data.

To limit the amount of overfitting, one may decrease the complexity of a trained network through pruning, or by limiting the growth in complexity during training, or by terminating the training when the network begins to overfit. Each is described below.

A trained network that overfits may be pruned by removing weights and units that have minimal effect on the network's error. This may be performed by sequentially setting each weight to zero and testing the error of the resulting network. Methods that require less computation rely on measures of utility for each weight. For example, the Hessian of the error with respect to the weights for a given input pattern is the sensitivity of the error with respect to each weight [LDS90]. Other measures of utility include [MS89] and [RP92].

The complexity of the function learned by a neural network is related to the number of hidden units and the number of weights of magnitude significantly different from zero. One way to limit the growth of complexity during training is to add an error term that biases the gradient search for weight values to regions in weight space of low-magnitude weights. Weigend, Rumelhart, and Huberman [WRH91] developed a term that penalizes large magnitude weights and applied this to several time series prediction problems.

The third method for controlling overfitting is more dependent on the data than the previous two. The data is divided into three parts, two for training and testing, and the third for determining when the network is overfitting. The third part is called the *validation* set. After each pass through the training data, the error on the validation set is calculated. Initially, the error on the training and validation set decreases. As the network begins to overfit, the error on the validation set begins to increase. At this point, prior to convergence on the training data, training is halted and the error on the testing set is calculated as a measure of the network's performance. Weigend [Wei94] used this technique on a time-series prediction problem and found it to be useful even for very small networks.

For the experiments described here, we applied the validation set method for early stopping. We divided the data into 10 distinct subsets, corresponding to the 10 recording sessions for one subject performing two tasks. One subset was selected to be the validation set, another subset was the testing set, and the remaining eight subsets comprised the training data. Thus, there are 90 possible combinations of training, validation, and testing sets. Notice that training, validation, and testing always were performed on data from distinct recording sessions. All 90 combinations were used and classification performance (on the testing sets) were averaged over the 90 runs.

3. Programming Technique

The data windowing, K-L transform, and frequency analysis were performed with a combination of Unix shell scripts and C code and run on a Sun Sparc 10. The eigenvector and frequency analysis were performed using implementations from [PFTV88].

The classification experiments were performed on a CNAPS Server II from Adaptive Solutions, Inc.. Our CNAPS system is a parallel, SIMD architecture with 128, 20 MHz processors, upgradable to 512 processors. It can be programmed at three levels, using assembly language, C with parallel programming extensions, or a library of routines that implements error backpropagation. Assembly language can be used to write highly efficient code, but the library of backpropagation routines is by far the most efficient in terms of development time. The C language level is intermediate, giving the programmer the ability to write efficient code in a familiar language. We chose to use the existing library to implement the error backpropagation algorithm. Specifically, we used Adaptive Solutions' `BPfolded` routine, which distributes the load even when the network is larger than the number of processors.

The library consists of routines to set algorithm parameters, to identify the source of the data, and to execute the algorithm. The following is an outline of the C program that we used to call the backpropagation routines. All steps, except step 9, were performed on the CNAPS server; step 9 was implemented in C. The C program was compiled and run on a Sun Sparc 10.

1. Connect to CNAPS server and specify the training algorithm.
2. Specify the algorithm's parameters, including the number of inputs, outputs, and hidden units, the number of training and validation patterns and the names of files containing them, the maximum number of epochs to train, the error criterion at which to terminate training, the learning rate constants for output and hidden layers.
3. Specify names of training and validation data files.
4. Set control modes of CNAPS to log the validation error by epoch.
5. Generate random initial weight values between -0.2 and 0.2.
6. Initiate execution on the CNAPS. Upon completion, results are returned in a predefined C structure.
7. Identify epoch number at which error on validation set was lowest.
8. Retrain, starting with same initial weights, stopping at epoch found in previous step.

9. Using the weights found in the previous step, calculate the mean square error over the testing set.

10. Save results to a file and repeat after changing the learning rates, number of hidden units, or training and validation sets or input representation (by specifying different data files).

All data for a given representation is stored in a file with one pattern per line and alternating between baseline and mental arithmetic tasks. To produce the training, validation, and testing data files, this file is split into ten equally-sized pieces and each piece is converted into the binary format required by the CNAPS server. This step is performed using the conversion tool `cv` from Adaptive Solutions. The assembly from these 10 parts into training, validation, and testing sets was implemented in the Tcl scripting language. The Tcl/Tk toolkit and shell is a public domain, interactive programming environment for creating graphical user interfaces [Ous94] and general utilities. We chose Tcl to implement this for two reasons. First, we plan to develop a graphical user interface to facilitate the development and visualization of different representations of EEG signals. We will also add components to the GUI to allow us to control the running of the CNAPS server and analyze the results. The second reason we chose Tcl is the ease of programming in its interactive environment.

4. Results

Table 1 shows the results of all classification experiments as the average percent of test patterns classified correctly, out of 158 patterns. This table includes 90% confidence intervals, based on 90 repetitions (see Section 2.7).

Hidden Units	Unprocessed (250 Hz)	K-L	Frequency-Bands
40	53.2 ± 0.6	51.7 ± 0.6	73.9 ± 0.7
5	52.8 ± 0.6	51.8 ± 0.7	72.9 ± 0.6
1	52.1 ± 0.5	50.4 ± 0.5	73.1 ± 0.6

Table 1: Percent of test patterns classified correctly for different sized networks and different signal representations.

Clearly the best classification accuracy is achieved with the frequency-band representation, giving an average accuracy of about 74% for a network with 40 hidden units, though the accuracy varies little for other network sizes, including a network with a single hidden unit. Performance with the other representations is significantly lower, ranging from 50% to 53%. The effect of the number of hidden units is slightly more significant for the unprocessed and K-L representations than for the frequency-band representation. These results suggest that the energy within standard frequency bands is more useful in discriminating the two mental tasks than is the unprocessed data, or dimensionally-reduced data. This hypothesis must be tested by further experimentation. It appears that the performance of the unprocessed and K-L representations is increasing with network size, but the differences are not statistically significant. We have not yet tried networks with more than 80 hidden units.

To determine which frequency bands or asymmetry ratios were most useful, the weights to which the neural network converged must be examined. The weight magnitudes provide some information about which inputs are most relevant. More informative would be the partial derivatives of the network output with respect to each input component, showing which input components have the greatest effect on the output for a given input pattern.

4.1 Benefit of Parallel Implementation

To obtain the results reported here, each network was trained a number of times to approximately optimize the training algorithm's parameters—the learning rates for the output and hidden layer. After finding good parameter values, 90 runs of 1,000–5,000 epochs were made with each network to calculate the averages in Table 1. The use of the parallel CNAPS server made the scope of this study practical.

To estimate the actual benefit of using the CNAPS server, we observed the execution time of training various sized networks for 1000 epochs using the unprocessed data representation. Results are graphed in Figure 9 as the minutes of execution time versus number of hidden units. Execution time increases approximately linearly in the number of hidden units for the serial

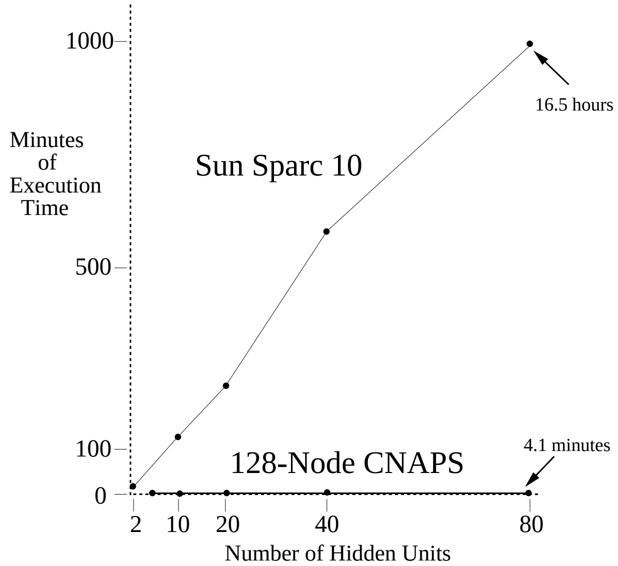

Figure 9: Minutes of execution time for increasing network size for the parallel CNAPS server and a serial Sun Sparc 10. All runs were for 1000 epochs.

Sparc 10, but the execution time for the parallel CNAPS server increases from 3.8 minutes for two hidden units to only 4.1 for 80 hidden units. A network of 80 hidden units takes about 240 times longer on the Sparc 10 than on the CNAPS server. Running 90 repetitions of the training procedure for a 40 hidden unit network took approximately 6 hours on the CNAPS Server; on the Sparc 10 this would require 37 days.

4.2 Program Development Effort

Most of the implementation effort for this study was devoted to the extraction of the binary EEG data from tape, separating and gathering the data for each experiment, dividing it into segments, then normalizing and converting to the binary format for CNAPS machine. Writing the programs for running on the CNAPS server took one graduate student approximately one month to learn how to use the library. This effort is detailed below:

Retrieving data from tape: This required discussions with Keirn to fully understanding the format of the data on the tape. One graduate student wrote a C program to convert the data to ASCII and extract the data we needed to run each program. Effort: 2 weeks.

Generating training and testing data: The data had to be segmented into quarter second intervals, assigned a correct classification value, normalized, divided into training and testing sets, and converted into the binary form required by the CNAPS machine. Two graduates students worked on this phase. Effort: 3 weeks.

Implementing the eigenvector analysis: The Numerical Recipes algorithm [PM92] was embedded into a C program that performed the eigenvector decomposition and projected all data vectors onto the highest ranked eigenvectors. Effort: 1 week.

Implementing the frequency analysis: A C program was written to compute the Burg AR coefficients using the `evlmem` and `memcof` C code from [PM92]. Effort: 1 week.

Implementing backpropagation: One student taught himself how to program the CNAPS machine using the library of backpropagation routines. He wrote a C program that accepts a number of command line arguments, calls the appropriate CNAPS routines to initialize the machine and the backpropagation algorithm, and then calls the routines to perform the training and saving of results. Effort: 1 month.

Interpretation of results: Awk [AKW88] scripts were written to calculate means and standard deviations of the results from multiple runs. Effort: 1 day.

5. Conclusion

Two conclusions can be drawn from this work. The first is related to the results, the second to the method. The results strongly suggest that the frequency-based representation produces significantly more accurate classification than do the unprocessed or K-L representations. The size of the neural network appears to have little effect on the classification accuracy. This means that the signal representations we have considered do not bear significant relationships much beyond the near-linear relationships that are possible with networks having a single hidden unit.

This conclusion must be supported by further experimentation. We are currently investigating the effect of averaging the output of the network over successive quarter-second segments. Preliminary results show that by doing so the classification accuracy can be increased up to 90% correct by averaging over segments from the full 10 second recording period. This, of course, would be impractical for the real-time control of a wheel chair. We are also considering other representations, such as wavelets. Wavelets represent both frequency and time features of a signal. For this reason, periods over which a signal is nonstationary are more accurately represented with wavelets than with a strictly frequency-based representation.

The second conclusion from this study is the utility of the parallel implementation of the error backpropagation algorithm. A much greater number of network sizes and initial weight vectors could be evaluated on the CNAPS server than could be completed in a comparable amount of time on a serial machine. The experiments reported here would have required over one month to complete on a Sun Sparc 10.

The utility of this parallel implementation for a portable, real-time, EEG pattern recognizer is currently not known. If networks with only one or two hidden units suffice, then a parallel implementation of the neural network classifier may be unnecessary. However, a parallel implementation of the Burg algorithm for computing the frequency-based representation might significantly reduce the overall response time. Parallel algorithms for computing the FFT (Fast Fourier Transform) on SIMD architectures are well-known [KV87, JJS86, Swa84], but the Burg method, based on an AR model, produces a smoother spectrum than does the FFT when applied to noisy signals, such as EEG. Incremental methods, based on Kalman filter techniques, exist for calculating the coefficients of an AR model as new samples are received. We are investigating parallel implementations of these algorithms as efficient means for computing the frequency-based representations. However, the primary bottleneck we currently face in a real-time implementation is the apparent requirement of averaging over a number of successive time segments to gain sufficient classification accuracy. Even if the classifier provides a real-time response, several seconds of EEG samples must be processed before a confident task identification can be made. Thus, further experimentation with different representations is required.

Acknowledgements

The authors would like to thank Jorge Aunon and Zachary Keirn for providing the EEG data, Ed Orosz for installing and interpreting the data, and Andreas Weigend for his helpful discussions concerning the analysis of multivariate time series. This research was funded by the National Science Foundation through grant IRI-9202100.

References

[AKW88] A. V. Aho, B. W. Kerninghan, and P. J. Weinberger. *The AWK Programming Language*. Addison-Wesley, 1988.

[DOG74] J. C. Doyle, R. Ornstein, and D. Galin. Lateral specialization of cognitive mode: II EEG frequency analysis. *Psychophysiology*, 11:567–578, 1974.

[EW80] H. Ehrlichman and M. S. Wiener. EEG asymmetry during covert mental activity. *Psychophysiology*, 17:228–235, 1980.

[FD88] L. A. Farwell and E. Donchin. Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70:510–523, 1988.

[Gev87] Alan S. Gevins. Overview of computer analysis. In A. S. Gevins and A. Rémond, editors, *Methods of Analysis of Brain Electrical and Magnetic Signals*, volume 1 of *EEG Handbook*, chapter 3, pages 31–83. Elsevier Science Publishers Biomedical Division, 1987.

[GO73] D. Galin and R. E. Ornstein. Hemispheric specialization and the duality of consciousness. *Human Behavior and Brain Function*, 1973.

[Jas88] H. Jasper. The ten twenty electrode system of the international federation. *Electroencephalographic Clinical Neurophysiology*, 10:371–375, 1988.

[JJS86] Leah H. Jamieson, Philip T. Mueller Jr., and Howard Jay Siegel. Fft algorithms for simd parallel processing systems. *Journal of Parallel and Distributed Computing*, 3:48–71, 1986.

[KA90] Zachary A. Keirn and Jorge I. Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical Engineering*, 37(12):1209–1214, December 1990.

[Kei88] Zachary A. Keirn. Alternative modes of communication between man and machine. Master's thesis, Purdue University, 1988.

[KMLW84] R. H. Kraft, O. R. Mitchell, M. L. Languis, and G. H. Wheatley. Hemispheric asymmetry during six- to eight-year old performance of piagetian conservation and reading tasks. *Neuropsychologia*, 22:637–643, 1984.

[Koh84] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[KV87] Donald E. Kirk and Jacques G. Verly. An algorithm for distributed computation of ffts. *Computing & Electrical Engineering*, 13(2):83–96, 1987.

[LDS90] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, page 598. Morgan Kaufmann, San Mateo, CA, 1990.

[LTL93] Shiao-Lin Lin, Yi-Jean Tsai, and Cheng-Yuan Liou. Conscious mental tasks and their EEG signals. *Medical & Biological Engineering & Computing*, 31:421–425, 1993.

[MS89] M. C. Mozer and P. Smolensky. Skeltonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky, editor, *Advances in Neural Information Systems*, volume 1, pages 107–115. Morgan Kaufmann, San Mateo, CA, 1989.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit*. Professional Computing. Addison-Wesley, Reading, Massachusetts, 1994.

[PFTV88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[PM92] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing*. MacMillan, New York, 1992.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. W. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and The PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. Bradford, Cambridge, MA, 1986.

[RP92] S. Ramachandran and L. Pratt. Information measure based skeletonisation. In *Advances in Neural Information Systems*, volume 4. Morgan Kaufmann, San Mateo, CA, 1992.

[Swa84] Paul N. Swartztrauber. Fft algorithms for vector computers. *Parallel Computing*, 1:44–63, 1984.

[Wei94] Andreas S. Weigend. On overfitting and the effective number of hidden units. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 335–342, 1994.

[WRH91] Weigend, Rumelhart, and Huberman. Generalization by weight-elimination with application to forecasting. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 875–882. Morgan Kaufmann, 1991.