HAND (MOTOR) MOVEMENT IMAGERY CLASSIFICATION OF EEG USING TAKAGI-SUGENO-KANG FUZZY-INFERENCE NEURAL NETWORK

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Biomedical Engineering

by

Rory Larson Donovan

June 2017

COMMITTEE MEMBERSHIP

TITLE:	Hand (Motor) Movement Imagery Classification of EEG Using Takagi-Sugeno-Kang Fuzzy-Inference Neural Network
AUTHOR:	Rory Larson Donovan
DATE SUBMITTED:	June 2017
COMMITTEE CHAIR:	Xiao-Hua Yu, Ph.D. Professor of Electrical Engineering
COMMITTEE MEMBER:	Robert Szlavik, Ph.D. Professor of Biomedical Engineering
COMMITTEE MEMBER:	Jamey Eason, Ph.D. Lecturer of Biomedical Engineering

ABSTRACT

Hand (Motor) Movement Imagery Classification of EEG Using Takagi-Sugeno-Kang Fuzzy-Inference Neural Network

Rory Larson Donovan

Approximately 20 million people in the United States suffer from irreversible nerve damage and would benefit from a neuroprosthetic device modulated by a Brain-Computer Interface (BCI). These devices restore independence by replacing peripheral nervous system functions such as peripheral control. Although there are currently devices under investigation, contemporary methods fail to offer adaptability and proper signal recognition for output devices. Human anatomical differences prevent the use of a fixed model system from providing consistent classification performance among various subjects. Furthermore, notoriously noisy signals such as Electroencephalography (EEG) require complex measures for signal detection. Therefore, there remains a tremendous need to explore and improve new algorithms. This report investigates a signal-processing model that is better suited for BCI applications because it incorporates machine learning and fuzzy logic. Whereas traditional machine learning techniques utilize precise functions to map the input into the feature space, fuzzy-neuro system apply imprecise membership functions to account for uncertainty and can be updated via supervised learning. Thus, this method is better equipped to tolerate uncertainty and improve performance over time. Moreover, a variation of this algorithm used in this study has a higher convergence speed. The proposed two-stage signal-processing model consists of feature extraction and feature translation, with an emphasis on the latter. The feature

extraction phase includes Blind Source Separation (BSS) and the Discrete Wavelet Transform (DWT), and the feature translation stage includes the Takagi-Sugeno-Kang Fuzzy-Neural Network (TSKFNN). Performance of the proposed model corresponds to an average classification accuracy of 79.4 % for 40 subjects, which is higher than the standard literature values, 75%, making this a superior model.

Keywords: Brain-Computer Interface (BCI), Electroencephalography (EEG), Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN), Discrete Wavelet Transform (DWT).

ACKNOWLEDGMENTS

I would like to give a special thank you to my wife, Alba Tapia, for keeping me out of trouble and keeping my eyes on target. Also, a thank you goes to my father, John, for showing me the value of hard work and supporting me through my journey. I would also like to thank my mother, Judy, for providing the food and shelter to get where I am. Lastly, thank you Dr. Yu for kindling my fervor for computational intelligence. Under your instruction, I have found this wonderful passion to supplement my incredible college experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1: Statement of the Problem

Roughly, 185,000 amputations are performed annually in the United States, primarily caused by diabetes and peripheral heart disease [1]. Nerve injuries correspond to an estimated $150 billion in healthcare expenditure [2]. Currently, a Brain Computer Interface (BCI) systems, a hardware-software integrated communication systems intended to interface the human brain or cranial epidermis, poses as a viable solution because this device is capable of treating, restoring, and augmenting peripheral nervous system and arm function disabled by amputation or other neuromuscular disorders [3]. For example, an amputee can regain the ability to grasp an object using a neuroprosthetic arm [4]. However, contemporary neuroprosthetic systems are not readily available on the market, likely because their recordings are unreliable [5] or have high associated risk (as a class III medical device) [6]; and are thus are more difficult to receive regulatory approval. Furthermore, they are designed exclusively for one individual at a time making it difficult to receive proper quality control [7]. As a result, the current state of this medical treatment is not reaching its full potential improvements are needed in the areas of information transfer speed, accuracy of detection, and adaptability to make these devices of sufficient quality to be extended to the total available market [8]. With these improvements, it is possible to provide a greater breadth of treatment to the various subjects affected by these neurological disorders.

1.2: List of Terms

This study focuses on the signal processing steps in BCI, which includes feature extraction from neural data and the translation of those features into an output classification decision. The process begins with the acquisition of event-related potentials, recorded as electromotive forces. These values are collected from instruments such as electroencephalograms (EEG) and electrocorticograms (ECoG), recorded from the surface of the scalp and intracranial, respectively [9]. The primary difference between these two electrode placement systems being the intrusiveness and biocompatibility, refer to ISO 10993 [10]. The electroencephalograph (EEG) is a prime contender for electrophysiological measurement because its minimal cost and low level of invasiveness is highly appealing to researchers. The EEG is used in this study due to the accessibility of the data.

EEGs retain spatiotemporal information [11], while they are relatively inexpensive and non-invasive. Furthermore, EEGs are capable of transmitting neuronal signal to an external device in order to drive a prosthetic arm [12]. However, EEGs are notoriously noise [13] and therefore require extensive signal processing to a degree that is not currently achievable. Thus a multistep data analytical procedure for noise reduction is utilized, comparing and contrasting various filtration methods and feature translation methods in Chapter 2 to reduce the noise and the size of the feature space. Blind source separation (BSS) methods, including Automated Artifact Rejection (AAR) and Independent Component Analysis (ICA), reduce the noise. This method is advantageous over those that do not use find new projections of data because it reduces information that is contaminating the data results.

The Discrete Wavelet Transform (DWT) is a method of transforming data into the frequency domain, similar to the Discrete Fourier Transform (DFT). It is often utilized to extract features for spectral analysis. For example, spectral energy can be calculated at various frequency bandwidths from sets of detail and approximation coefficients [14], which are relatable to transient points on a compressed time scale. This method is advantageous because it improves the balance between transient and frequency precision [15]. Therefore, the extracted feature more accurately reflects the original waveform than, for example, the sinusoidal waveforms of the Fast Fourier Transform (FFT). This method may also produce better results other methods such as the Hilbert Huang Transform (HHT), which produces inconsistent bandwidths for analysis [14]. Thus spectral analysis of the transient properties is inconsistent in reference.

The Takagi-Sugeno-Kang Fuzzy-Neural Network (TSKFNN) is a network that incorporates fuzzy logic principles and machine learning. Inputs feed into predefined functions such as the Gaussian distribution to determine activation strength. The signal feeds forward and is met by connective resistance. The signals are linearly combined then transformed once in the premise layer, then once again in the consequence layer. The fuzzy inference mechanism, known as Takagi-Sugeno-Kang (TSK) fuzzy inference [16], allows precise inputs to be formulated degrees of uncertainty; yet precision to be achieved in the output. Machine makes guesses to reduce error classifications of future input learning with each new sample.

The proposed method employs a hybrid back propagation algorithm and least squares error to learn and Takagi-Sugeno style fuzzy inference for increased convergence speed [17]. Wherein the standard learning techniques that use the back propagation are

slow to converge and are unstable, the hybrid method updates the consequence layer using the LSE. This ensures maximal classification accuracy on the training data by manipulating the consequence parameters alone and hastens the speed of convergence. The premise layer parameters are updated using the back propagation algorithm, which retains the online learning mechanism of the neural network. Thus, together these allow for adaptive learning as well as speedy convergence.

1.3: Purpose of Study

This study investigates a state-of-the-art BCI signal-processing model for the purpose of increase classification accuracy of imagined hand motor movement of Electroencephalograms (EEG) recorded from a surface electrode. This implementation is part of the groundwork needed for improving treatment of amputations in a clinical setting. With the proposed implementation of fuzzy logic, the intended system is more tolerant for imprecision [17] and thus better suited for BCI adaption than current methods. Together, fuzzy-logic and machine learning make for a more robust classification system for the purpose of imagined motor movement identification than previous models such as the Artificial Neural Network (ANN) and the Support Vector Machine (SVM).

The proposed approach consists of two stages, feature extraction and feature translation. The first stage incorporates Automated Artifact Rejection (AAR) and Independent Component Analysis (ICA) to separate the signal into independent sources, or components, based on statistical variance of the data. Component elimination increases the signal to noise ratio, making this model less prone to erroneous classification over the other methods. This process is followed by analysis of the Event Related (de) Synchronization (ERD/ERS) activity by means of the Discrete Wavelet Transform (DWT) to extract features for the machine-learning stage or feature translation stage. In this stage, the extracted features become the input, and binary identifiers are calculated via Takagi-Sugeno-Kang Fuzzy Inference Neural network (TSKFNN). These classification performance values are compared to the standard literature value to determine the superiority of this robust model.

CHAPTER 2: LITERATURE REVIEW

2.1: Previous Methods in Literature

Various transformations are used to decode Electroencephalograms (EEG) characteristics in transient time course in the Brain-Computer Interface (BCI) literature. The Fast Fourier Transform (FFT) is utilized most often [18-20] because it is robust and has been successfully employed in the past. More recently, methods such as the Discrete Wavelet Transform (DWT) and the Hilbert Huang Transform (HHT) with Empirical Mode Decomposition (EMD) are making their way into the biological research community because of their basis functions, or lack thereof, and their ability to handle non-stationary, stochastic data [18][21]. Although the FFT has worked in the past, these new methods may be better suited for physiological signals such as the EEG for various reasons.

First, the DWT is better able to reconstruct a wave, which is better for spectral estimation because the temporal representation more accurately models the original signal [22]. DWT also uses predefined basis functions, however that can be sharp or smooth, orthogonal or bi-orthogonal, symmetric or asymmetric, incorporating all sorts of properties [23]. Thus, this method is a better alternative for reconstructing waveforms with sharp peaks for decomposition than the FFT (in which the signal is always smooth and orthogonal). However, with this method, proper selection of the mother wavelet determines how accurate the reconstruction is going to be.

2.1.1 Multivariate Empirical Mode Decomposition

The Hilbert-Huang Transform (HHT) is making its way into the biological research community, in addition to the Discrete Wavelet Transform (DWT), as a solution to the problems faced by using a priori assumptions of the data [21]. The authors of [19] are able to successfully implement a noise-assisted multivariate version of the Empirical Mode Decomposition model (NA-MEMD), achieving higher mean classification performance. They obtain superior results from NA-MEMD than from the synchronized wavelet transform (SST), continuous wavelet transform (CWT), and the Butterworth Filter (BF). In the study, mu and beta waves (those pertaining to the somatosensory cortex) are analyzed and neural activity is discriminated using the Common Spatial Patterns (CSP) algorithm.

For decomposition, Empirical Mode Decomposition (EMD) determines time-frequency localized information by decomposing the wave into basis functions to be superimposed on top of each other. Two envelopes are calculated, $e_u(t)$ and $e_l(t)$, by interpolating the maxima and minima values. Intrinsic mode functions (IMF) denoted as $c_i(t)$ are obtained from the mean of the two local extrema envelopes subtracted from the previous IMF, $c_{i-1}(t)$, where *I* is the order of the IMF [24].

$$\bar{m}(t) = (e_l(t) + e_u(t))/2 \tag{1}$$

$$c_i(t) = \tilde{v}(t) - \bar{m}(t) \tag{2}$$

This process is iteratively repeated until the stopping criterion is met. The signal then becomes a monotonic residual signal and the original signal is reconstructed using Eq. (3), where *M* is the number of IMFs [24].

$$v(t) = \sum_{i=1}^{M} c_i(t) + r(t) \tag{3}$$

The Hilbert Huang Transformation (HHT) can be applied to obtain a time-frequency spectrogram as indicated in (4), where symbol *P* indicates the Cauchy principle value. The signal is then described using (5). The variables $a_i(t)$ and $\theta_i(t)$ denote the *i*'th amplitude and phase functions [25].

$$H(c_i(t)) = \frac{1}{\pi} P \int_{-\infty}^{\infty} \frac{c_i(t')}{t - t'} dt' \tag{4}$$

$$V(t) = \sum_{i=1}^{M} (c_i(t) + jH(c_i(t))) = \sum_{i=1}^{M} a_i(t) e^{j\theta_i(t)} \tag{5}$$

Through differentiation of the phase functions, the instantaneous frequency can be obtained using $w_i(t) = d\theta_i(t)/dt$.

One major disadvantage for EMD is that the local frequency changes, in a process known as mode mixing. Mode mixing is instigated by time lag and noise and results in temporal property discrepancies and randomness between channels of EEG [17]. Therefore, frequency locked information would be difficult to achieve even if temporal properties information of the sources is known *a priori*. This is important for analyses in which *a priori* frequency information is known and amplitude estimation of the sources from a particular band is desired. Although the EMD method alone is precluded from being applied independently to multiple channels, the authors in this study [18] present a solution to these problems by suggesting the multivariate approach.

In this study, features of the ensemble data analysis, with Gaussian white noise channels, are incorporated with bivariate empirical mode decomposition (BEMD) to achieve the noise-assisted multivariate EMD (NA-MEMD). This representation creates

similar bands amongst the data. In this form, the noise resides in a subspace separate from the original signals and is used to enforce a filterbank structure. The IMF subspace is retained while the noise subspace is eliminated which improves the ability of the algorithm to tolerate the noise in the original data and distinguish the trends. It is important to note that an increase in the number of channels results in a better estimation of envelopes and therefore better identifies common activity [26].

The Common Spatial Patterns algorithm is used to extract features by maximizing the variance in one class of signals while minimizing the variance in another class. This allows the determination of ERD and ERS activity based on subtle changes in power in the EEG data [27]. The features were translated using the support vector machine (SVM) with a Gaussian kernel. The PhysioNet data was divided into 32 input vectors for training and 13 input vectors for testing out of the original 45 input vectors [18]. The classification was repeated 100 times with random sampling. Classification performance 77.7% for the NA-MEMD, whereas 64.9% for the synchronized-squeeze transform (SST), 72.3% for the continuous wavelet transform, and 74.8% for the fifth-order Butterworth Filter. The 11 channels selected for analysis include “FC3,” “FC4,” “C3, “CZ,” “C4,” “C5,” “C6,” “T7,” “T8,” “CP3,” and “CP4. The Morlet mother wavelet is selected for the CWT in the analysis. The CWT and SST scales are reconstructed to obtain the band-pass filter signals. Two noise channels are added to the data for decomposition and the Hilbert-Huang spectra is calculated.

Overall, this mathematical technique is advantageous for multiple reasons. First, the empirical mode decomposition method is advantageous in that it does not require a priori knowledge of the data to decompose a biological signal for classification; it is fully

data driven. Thus, there is no need for selection of a predefined basis function to achieve accurate results. In addition, it obtains highly localized time-frequency estimation as each intrinsic mode function captures different temporal scale information that is intrinsic to the data. Furthermore, EMD can be modified for spectral estimation, presenting the signal as decomposed time-energy-frequency deterministic functions. Lastly, EMD with the HHT poses the ability to handle non-linear, stochastic transient time course for time-frequency localization [29].

However, the time-frequency localization fluctuation also poses as a disadvantage for multichannel biological signals such as EEG because each signal produces a different number of IMFs and various frequency bands [19]. Furthermore, these signals change their properties with respect to time posing as an obstacle for multiple signal analysis. The first issue can be addressed through techniques such as noise-assisted analysis, but the analyses are not temporally fixed to a specific band. The Discrete Wavelet Transform (DWT) is better for a reliable decomposition with fixed frequency bands across all of the channels. Thus, DWT may remain a better option when extracting spectral features because, in the case of EEG, *a priori* the frequency properties are known.

2.1.2: Wavelet and Artificial Neural Network

The Discrete Wavelet Transform (DWT) is a powerful tool used to process spectral information. Much like the Discrete Fourier Transform (DFT), it can be used to represent signals in the frequency domain. However, the wavelets unique basis functions better reconstruct a signal with varying properties. Additionally, noise outside of the designated frequency can be eliminated in the frequency domain to improve event discrimination. With signal reconstruction, frequency fixed features can be extracted as previously demonstrated [29]-[31]. From the DWT, energy and various characteristics are calculated from *details* and *approximations* coefficients, see Chapter 4.2. These coefficients represent time points on a compressed scale, approximately half of the original size. Various calculations, such as those listed in Table 2, are performed on these values to obtain these features.

First, the DWT decomposes the wave into a set of detail coefficients, cD, and approximation coefficients, cA. The first variable represents time points on half of the spectrum corresponding to the higher frequencies, whereas the second variable represents time points for the lower frequency band. This process is repeated on each approximation each time decomposing the signal into smaller segment details and approximation, as demonstrated in Fig. 1.

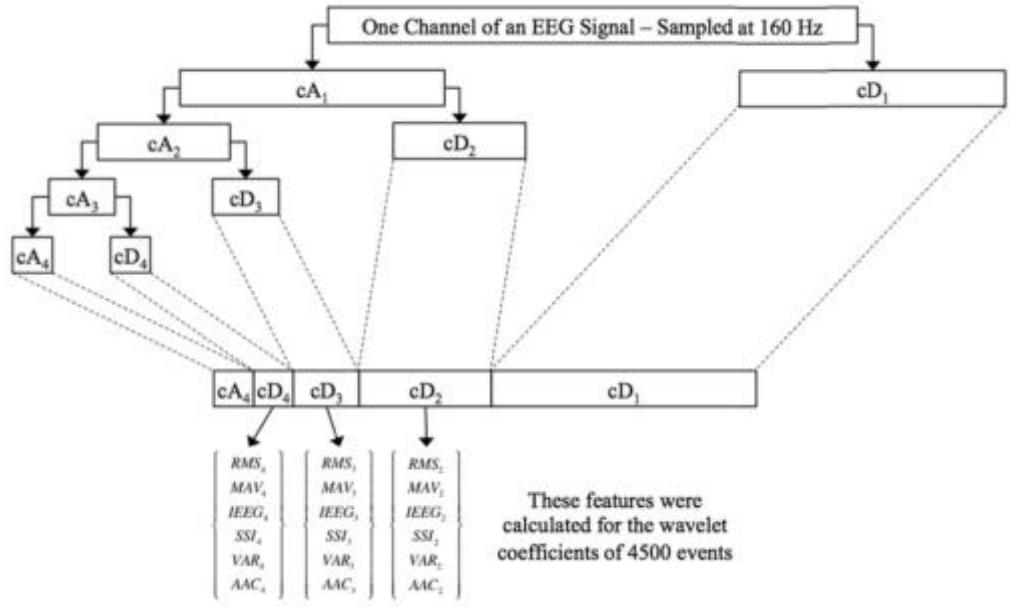


Fig. 1. DWT wavelet decomposition obtained [30]

Table 1. Detail and approximation decomposition for frequency bands

Signal Component	Frequency Range
cD_1	40 – 80 Hz
cD_2	20 – 40 Hz
cD_3	20 – 20 Hz
cD_4	5 – 10 Hz
cA_4	0 – 5 Hz

A 160 Hz signal is broken down to a 0 – 80 Hz band and an 80-160 Hz and an 80 Hz band is broken down to a 0 – 40 Hz and 40 – 80 Hz band. The detail and approximation coefficients from decomposition are then used in computation of the spectral features using (6-11) in Table 2.

Table 2. Characteristics used for measurement of EEG related time course data on detail and coefficients cD_i [30]

Name	Formula	
Root Mean Square (RMS)	$\sqrt{\frac{1}{N}\sum_{n=1}^{N} cD_i^2(n)}$	(6)
Mean Absolute Value (MAV)	$\frac{1}{N}\sum_{n=1}^{N} \lvert cD_i(n)\rvert$	(7)
Integrated EEG (IEEG)	$\sum_{n=1}^{N} \lvert cD_i(n)\rvert$	(8)
Simple Square Integral (SSI)	$\sum_{n=1}^{N} \lvert cD_i(n)\rvert^2$	(9)
Variance of EEG (VAR)	$\frac{1}{N-1}\sum_{n=1}^{N} \lvert cD_i^2(n)\rvert$	(10)
Average Amplitude Change (AAC)	$\frac{1}{N}\sum_{n=1}^{N} \lvert cD_i(n+1) - cD_i(n)\rvert$	(11)

Together, the DWT and the Artificial Neural Network (ANN) can used to classify Motor Movement Imagery (MMI) data using the electrodes multiple electrodes.

The Artificial Neural Network (ANN) is used to decode the data in the method for the second comparative study [30]. An ANN imitates the structure of the brain modeling the neuron as a node. In Frank Rosenblatt's perceptron model, an ANN model for information storage and organization, given in Fig. 2, the neurons mathematical representation consists of a linear combination of input values that undergo a non-linear

operation; this is the functional unit of an artificial neural network. This representation allows for the solution of non-linear classification decisions (such as the XOR benchmark test). This is accomplished using weighted combinations of the input, subjected to connective resistance. Then, a classification decision is made for each input vector. In this network, the first layer is the input layer and the last layer is the output layer, whereas all of the layers in between constitute the hidden layers.

(a) (b)

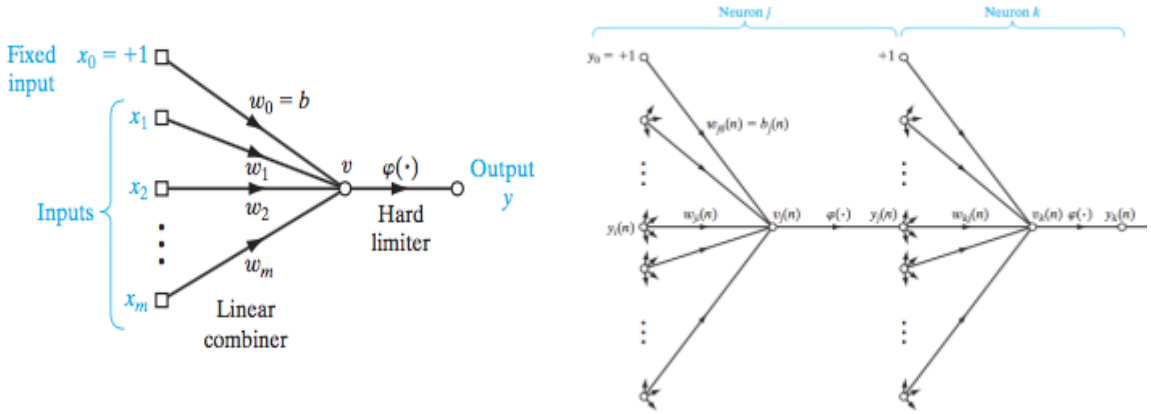


Fig. 2. Perceptron model as seen in (a) and representation of an ANN with and input and an output layer in (b) [32]

The back propagation algorithm allows the network to improve classification accuracy over time. The weights are updated by means of the gradient descent, minimizing the means square error of the output. There are two stages to the back propagation algorithm: feed forward and back propagation. In the forwarding feeding stage, the signals are conducted through the network meeting connective resistance. The output is calculated at each node until the signal reaches the outputs of the output layer, given in Algorithm 1.

Algorithm 1 The Back Propagation Algorithm (BP)

1. Layer 1 (Input Layer) - Let the *i'th* input be described by the variable x_i and the connective resistance be described by w_{ji} denoting that the signal is feeding from the input neuron i to j. The output can be describe as the weighted sum of the input undergoing a non-linear transformation $\varphi(\cdot)$ as given in the equation below,

$$y_j^1(N) = f\left(\sum_{i=0}^{m} w_{ji}^1(N) x_i(N)\right) \tag{12}$$

where the variable m is the number of neurons in the input layer, with the exception of the bias. The signal propagates into the *j'th* neuron of the first hidden layer.

2. Layer 2 (Output Layer) - Next, in the output layer, the mathematical formula of the single neuron output is the given as the linear combination of the weighted sum of the hidden neurons represented as

$$y_k^2(N) = O(N) = \sum_{i=1}^{n} w_{kj}^2(n) y_j^1(n), i = 1, \ldots, l \tag{13}$$

where *l* represents the number on neurons in the hidden layer and *k* represents the output layer.

After the output has been calculated from the input and the weight according to algorithm 1, the output is compared to a target value. The error signal is then conducted back through the network changing weights according to the delta rule. On-line learning requires recursively obtaining a gradient vector in which the elements are derivatives of the error with respect to the parameters of the network. For the proposed algorithm, which uses the supervised gradient descent method, the energy function is described as,

$$E = \frac{1}{2}(d_k - y_k)^2 = \frac{1}{2}e_k^2 \tag{14}$$

where *d* is the desired output, *y* is the actual output, *k* is the neuron, and e is the error term.

Algorithm 2 Back Propagation Algorithm

---

1. Layer 2 (Output Layer) - Then the learning algorithm is described as follows. According to the chain rule of calculus, the error signal calculated passing through the linear transformation function is as follows

$$d_k^2 = \frac{\partial E(N)}{\partial v_k^2(N)} = - \frac{\partial E}{\partial e_k(N)} \frac{\partial e_k(N)}{\partial y_k^2(N)} \frac{\partial y_k^2(N)}{\partial v_k^2(N)} \tag{15}$$

and the weights are calculated according to the delta rule, which can be expressed in the following way,

$$\Delta w_{kj}^2 = - h \frac{\partial E(N)}{\partial w_{kj}(N)}$$

$$= - h \frac{\partial E(N)}{\partial y_k(N)} \frac{\partial y_k(N)}{\partial w_{kj}(N)}$$

$$= h d_k^2 y_k^2 \tag{16}$$

where η is the learning rate parameter. The weights are updated according to the following equation:

$$w_{kj}^2(N+1) = w_{kj}^2(N) + \Delta w_{kj}^2(N) \tag{17}$$

The error propagates through the neurons with a non-linear activation function in this layer.

$$d_j^1 = j'_j(v_j(N)) \frac{\partial E(N)}{\partial y_j^1(N)}$$

$$= j'_j(v_j(N))e_k(N)\frac{\P e_k(N)\P v_k(N)}{\P v_k(N)\P y_j^1(N)} \quad (18)$$

Using the delta rule again, the equation for the updated weight becomes

$$w_{ji}^1(N+1) = w_{ji}^1(N) + h d_j^1 y_j^1 \quad (19)$$

Once sufficient performance on the training data has been achieved from updating the weights, the test data is fed to the network to determine the classification performance.

The authors of this study [30] compare the classification accuracy for the Coiflet, Symlet, and Debauchees wavelets. They use three detail coefficients, which correspond to cD_2, cD_3, and cD_4 (5-10 Hz, 10-20 Hz, and 20-40 Hz, respectively) and three electrodes, C3, CZ, and C4. Thus, there were 9 features in each input vector corresponding to 3 channels with 3 levels of details (3 channels x 3 details = 9 features). With 100 subjects with 45 trials, of event related time courses, for each subject, this corresponds to a total of 4500 inputs to the neural network. They achieve a maximum classification of 74.97% using an ANN with 20 hidden layers, mean absolute value, and a Coiflet with 4 vanishing moments. However, this process was repeated by the authors of [31] who were able to achieve 89.11% for the ANN with mean absolute value feature and Coiflet wavelets as their maximum classification performance. In this study, the AAR with the Blind Source Separation (BSS) algorithm of the EEGLAB toolbox was used in the MATLAB environment to automatically remove electromyographic (EMG) and electrooculargraphic (EOG) artifact.

In summary, ANN's with back propagation have been making headway in research in which precise mathematical model or the probabilistic classifier is not known. The physiology of the brain varies for different people; therefore it is difficult to find one mathematical model that will distinguish EEG data pertaining to a motor imagery event for a large amount of people, using other conventions. In the proposed ANN black box approach, an ANN learns to make classification decisions in a supervised learning manner without any a priori knowledge about the system. Additionally, it is better able to create reconstruct highly ordered polynomial equations to solve complex, non-linear classification problems.

ANN with the DWT method has advantage over others, such as Naïve Bayes Classifier, *k*-nearest neighbors, and even Radial Basis Functions (RBF) and Support Vector Machines (SVM) that are not adaptive, or do not have the ability to update their parameters to improve performance over time. Using supervised learning, and learning rate parameterization, speed of learning can be adjusted to improve performance quicker or more stably [32]. Performance must increase over time by definition according to the gradient descent method. As a result, the back propagation is an adaptive model capable of online-learning, constantly updating to new inputs.

However, there is a drawback to the standard ANN with BP, as the learning algorithm learning rate parameterization can greatly affect stability and convergence speed. Poor parameterization will lead to poor classification performance [32]. To add to this, this method of hard computation does not account for uncertainty, therefore the model (such as activation function) has to be known.

2.1.3: Support Vector Machine

Another classification technique that is often employed in BCI literature is the Support Vector Machine (SVM). By means of the kernel trick, a much larger class of functions can map non-linear functions onto a high-dimensional space. In this space, the ability of separation is likely increased [34]. The data set does not need to be linearly separable in the feature space, although it can be. A margin of separation is created from the hyper planes to the support vectors; through this arrangement the classification performance is optimized, as demonstrated in Fig. 3.

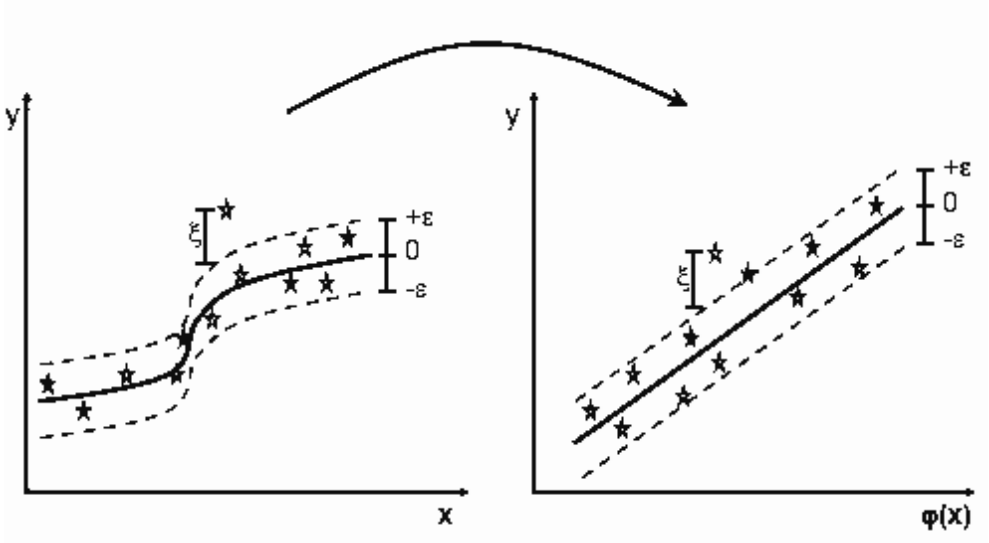


Fig. 3. Support Vector Machine classifier [36]

To achieve separation, the SVM relates a frequent pattern recognition problem involving the statistical optimization of a cost function for the data. The goal is reduce the probability of erroneous classification of future points based on the training data. For cases that are not linearly separable, a soft margin is imposed on the data so that it that it can learn a classification rule. The cost function is given as (20) [36],

$$\Phi(w,\xi) = C\sum_{i=1}^{N}\xi + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{20}$$

where C is a user-defined parameter, ξ is the slack parameter, and **w** is the weight vector. The dual representation of this formula is given in (21) [36].

$$Q(\alpha) = \sum_{i=1}^{N}\alpha - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{21}$$

where α is the Lagrangian multiplier and d is the adjustment factor for adjusting the width of the margin between the hyper plane and the support vectors. which is subject to the following limitations [36],

i. $\sum_{i=1}^{N}\alpha_i d_i = 0$

ii. $0 \le \alpha \le C$ for $i = 1, 2, ..., N$

The discriminant function $g(\mathbf{x})$, which separates the data, is written in the feature space as (22) [36],

$$g(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i d_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{22}$$

For $g(\mathbf{x}) > 0$, **x** is in class 1 ω_1

For $g(\mathbf{x}) < 0$, **x** is in class 2 ω_2

Selection of the kernel and the kernel parameters and soft margin parameter determines the effectiveness of the SVM method. The parameters are often selected with growing exponential sequences of the parameters. The optimal results are selected via cross validation.

The RBF types Mercer kernel for SVM always satisfies Mercer's theorem. It is given by (23) where s^2 is the width common to all kernels.

$$k(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2s^2} \| \mathbf{x} - \mathbf{x} \|^2 \right) \tag{23}$$

The main advantage of SVM is that it projects the data onto a higher space; therefore, it is more likely able to separate the data [34]. The presence of the higher dimensional space also decreases the risk of over fitting data, thereby reducing the classification error [37]. Although this higher dimensional feature space has desirable properties, this method has more neurons in the feature space, and therefore it increases the computational difficulty. Furthermore, the multi-class SVM is often reduced to several binary problems, which adds to the amount of computations that need to be performed [38]. Another problem is that the parameters that are obtained are difficult to comprehend [39].

In [40], the SVM machine is used to classify the PhysioNet MMI data from signals FC3, FC4, C3, C4, CP3, and CP4. The Debauchees mother wavelet is utilized with 4 vanishing moments. The SVM classifier with Gaussian kernel (RBF) is trained with the 360 training sets and tested with 90 sets for 10 subjects. Classification accuracy is reported as 75%.

2.2: Proposed Improvement to Methods in Literature

There are various advantages and disadvantages of the aforementioned methods, see of Motor Movement Imagery (MMI) classification, which prompted the proposal of the method that is used in this report, refer to Table 3. First, for feature translation techniques, the NA-MEMD method has mode mixing and, therefore, contains different information in each band, which isn't useful for a frequency fixed analysis. Because there is plenty of research behind neural oscillations and event related potentials, *a priori* knowledge of the system already exists. The frequency bands are already known and researchers know that they are looking for synchronization activity. Consequentially, the Discrete Wavelet Transform (DWT) is better for spectral analysis and specifically for determining spectral Event Related (De) Synchronization (ERS/ERD) because the frequency localizations are locked.

Next, for feature extraction, although the selection of the mother wavelet is very important for the accuracy of the analysis, there has been research to suggest that a few of the wavelets such as Symlet and Debauchees are better for decomposition than the Morlet and Coiflet wavelets used in the previous studies. Therefore, the proposed method uses a Debauchee wavelet with 2 vanishing moments, as is it demonstrates better results in [41][42]. Also, the DWT is selected as a better alternative better than the FFT because these wavelets can achieve and are better able to construct a waveform with sharp peaks and asymmetric properties.

Furthermore, the SVM method is problematic because it is only trained once. This means that this application would not achieve improved online performance. Also, a

Table 3. Summary of the Various Feature Translation Algorithms Used in EEG

	Method	Advantages	Disadvantages
Feature Extraction	MEMD	Fully data-driven; good time-energy-frequency analysis; good for stochastic signals	Mode mixing
	FFT	Lot of research to support use of various methods	Sinusoidal waveforms; does not do well with peaks and asymmetry
	DWT	Good for spectral analysis	A priori knowledge of the waveform needed for accurate analysis
Feature Translation	SVM	Higher classification performance	Can only be trained once
	ANN with BP	Accuracy increases with time	Slow convergence; may converge to local minimum
	TSKFNN with Hybrid learning	Tolerance for uncertainty; learning improves performance; quicker convergence	May converge to local minimum

multi-class problem complexity is increased for two-category classification scenarios, which is undesirable because it is inefficacious. The SVM can be interpreted as a model that statistically separates the data using Gaussian kernels for the RBF type SVM, however the Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) has the ability to account for uncertainty in the modeling of the premise layer. Probabilistic functions can be used to determine clusters centers for a neural network, through Subtractive Clustering or other clustering methods, to represent the data in a smaller dimensional feature space. Furthermore, the membership functions can be updated via back propagation to improve the performance. Table 3. compares and contrasts the various strengths and weaknesses of the methods discussed in Chapter 2.1.

CHAPTER 3: BACKGROUND

3.1: Electroencephalography

Electroencephalograms (EEGs) are measurements of physiological signals of neural origins recorded from the surface of the scalp as a degree of electromotive force. These signals are ideal for BCI applications due to their low cost, ease of use, non-invasiveness, and ability to extract information with high spatial-temporal resolution [11]. Despite these advantages, neuronal signals have to propagate through the Dura Mater and skull to get to the surface of the skin, and therefore have been more attenuated than other more invasive methods that use intracranial chronic electrodes. Furthermore, EEG signals are often more contaminated with electromyograms (EMG), electrooculargrams (EOG), power line, and environmental artifacts and have a very low signal to noise ratio [43][44].

However, not only does implantation of the electrodes directly into the brain introduces the risk of infection [45]. The mechanical mismatch between the implant and the brain, such as Young's Modulus (elastic modulus) inconsistency, instigates glial cell encapsulation that prevents chronic electrodes from conducting a signal through these scars within a couple weeks to months [45], which makes intracranial methods of recording less appealing for research. However, the high level of invasiveness comes with higher signal-to-noise ratio, which is important for being able to discriminate types of neural oscillations.

Despite these limitations, methods such as spectral analysis have been developed to remove sources of noise using frequency properties inherent to them. Through transformation into the frequency domain, elimination of frequency bands, and decomposition of the waveform in the time course can give information such as Variance

of EEG (VAR), Mean Absolute Value (MAV), or Root Mean Square (RMS) for each frequency band, Table 3. Furthermore, information such as energy percentage at each bandwidth or the power spectral density can be calculated at each band. The frequency bands that are referred to in this study are listed in Table 3 and demonstrated in Fig. 4. This study utilizes the Discrete Wavelet Analysis (DWT) for feature extraction. The details will be discussed in Chapter 3.2.

Table 3. Rhythmic activity bands in EEG

Bands	Frequency (Hz)	Associated With
Theta	4-8	Drowsiness
Alpha	8-13	Relaxation
Beta	13-30	Active thinking
Gamma	30-100	Cross-modal sensory processing

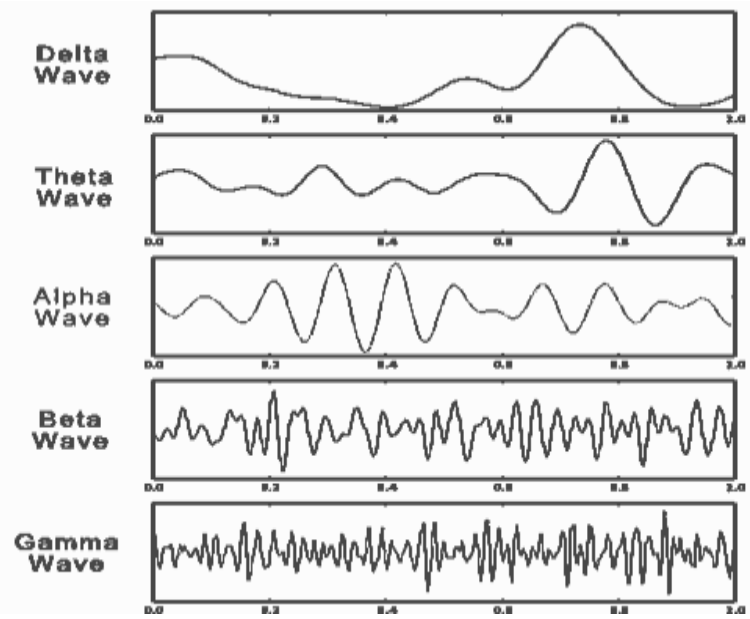


Fig. 4. EEG Rhythmic Activity Bands [46]

Sensorimotor rhythms (SMRs) and slow cortical potentials (SCPs) are analyzed in this report. Of these two, SMRs, Fig. 5, are more efficacious for neuroprosthetic control in BCI because they offer a higher level of control, in the form of degrees of freedom [47]. Moreover, SMRs exist in both healthy and afflicted subjects such as amputees and have been demonstrated efficacious in motor movement imagery tasks [48]; thus making it is an area of focus for studies involving the brain through methods such as Event Related Desynchronization (ERD). Sensorimotor rhythms appear over the sensorimotor cortex and are detected in the beta/alpha wave band.

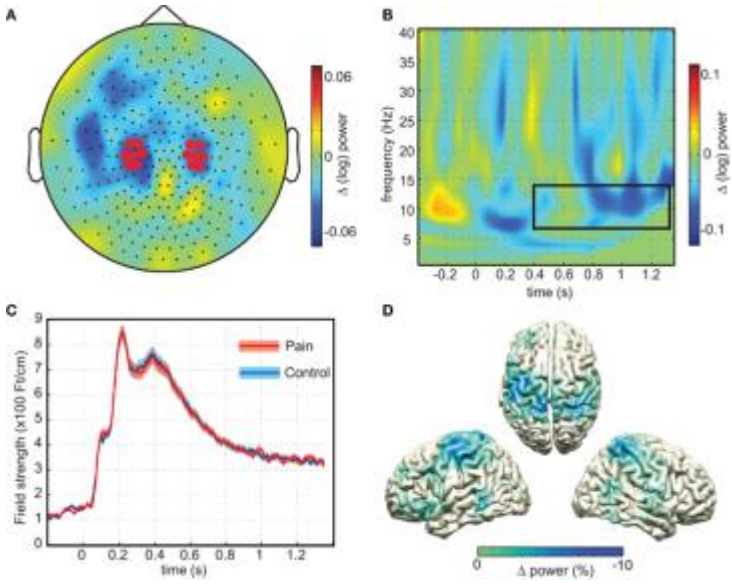


Fig. 5. Suppression of pain response causal relationship with sensorimotor alpha waves [49]

SCPs, on the other hand, originate from large cell assemblies in the upper cortical layer. They may be externally triggered or self-induced and are present during states of behavioral or cognitive preparation [50]. Activity of a thalamo-cortical-striatal circuit

encompassing the prefrontal cortex [51], primary, and supplementary motor areas [52], posterior parietal cortex [53], anterior cingulate cortex and thalamic nucleus [54] instigate contingent negative variation [55] in EEG. SCPs can be detected at the frequencies of theta and mu neural oscillations as negative shifts in transient time course. Regulation of SCPs appears to be attenuated for motor preparation [56].

In EEG, electrodes are usually placed according to systems best capture the signals of interest. SMRs are located most nearly in the motor cortex (PC3, PCZ, PC4) or somatosensory cortex (C3, CZ, C4) for hand imagery in ERD. For Movement Related Cortical Potentials (MRCP), electrodes are often used in both locations, Fig. 6.

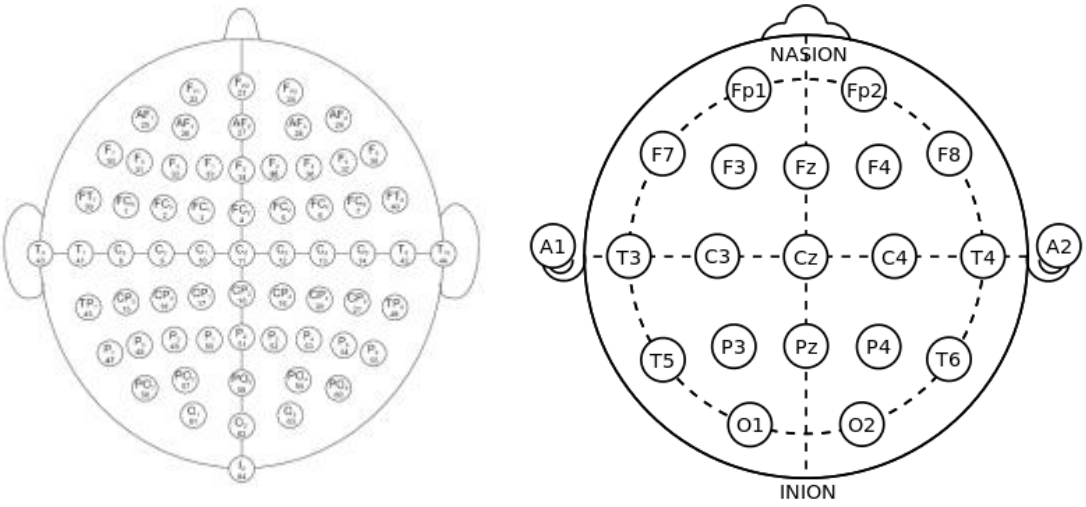


Fig. 6. Internationally recognized methods for EEG placement. This figure demonstrates (a) the high-density 10-10 system [12] and, (b) the low density 10-20 EEG [57]

Various EEG placement systems have been adopted with various densities selected, including 10-10 and 10-20. Less complex models such as 10-20, in Fig. 6 (a), can use 21 electrodes and is, therefore, less computationally demanding. High-density

placement systems such as the 10-10 international system, in Fig. 6 (b), which uses 64 electrodes placed throughout the scalp and can provide higher resolution at the cost of computational complexity. Electrode systems derive their names from the distance between the adjacent electrodes. The numbers 10 and 20 indicate the distance between the electrodes front to back, and left to right; for example, 10 % or 20 % of the total distance of the skull, correspondingly. Both of these electrode placement systems will appear later in Chapter 3 and in Chapter 4.

3.2: PhysioNet Motor Movement & Imagery Database

The EEG data in this research comes from Motor Movement and Imagery (MMI) Database available online at PhysioNet [12]. The data consists of more than 1,500 EEG recordings acquired from 109 subjects; recorded at 160 Hz using the BCI2000 with 64-channel EEG, performing a total of 14 experimental runs. Tests are included in Table 4:

Table 4. PhysioNet EEG MMI task list executed by each person, 3 x

Number	Tasks
1	1 min. baseline, eyes open
2	1 min. baseline, eyes closed
3	2 min. task 1 (open and close left or right fist)
4	2 min. task 2 (imagine opening and closing left or right fist)
5	2 min. task 3 (open and close both fists or both feet)
6	2 min. task 4 (imagine opening and closing both fists or both feet)

Baseline tests were performed one time per test subject for a one-minute period, whereas tasks 1 through 4 are performed three times for two-minute periods each. For Task 4, a target appears on either the left or right side of the screen, annotated T1 or T2 respectively, which prompts the subject to open and close the corresponding fist until the target disappears after approximately 4.1 seconds (this is the event-related time course used in the study for the DWT energy calculation). Each trial was followed by 4.1 seconds of rest, corresponding to 656 time points per electrode on each trial [8][12]. Overall, the left and right fist activity was engaged in a total of 15 trials for three separate periods.

3.3: Feature Extraction

3.3.1: The Discrete Wavelet Transform

Much of the noise in electroencephalograms (EEG) existing outside of the 0.01-100 Hz can be removed using high-pass and low-pass filters [58]. Artifacts caused by sweating, drift in impedance, muscle contraction, aliasing, and power lines contribute significantly to recordings outside of this range. Sweating and drifts in electrode impedance gradually change the measured voltage leading to saturation in amplitude and distortions over event-related time course [59]. Additionally, muscle contraction artifacts, primarily electrooculargraphic (EOG) and electromyographic (EMG) exist in frequencies and above the 0.01-100 Hz neuronal EEG range [60], [61]. Also, a natural phenomenon called aliasing causes lower frequency artifacts that may require a little more attention, but in many cases can be filtered out using a low pass filter [62]. Lastly, power line noise occurs at 60 and sometimes 50 Hz and can be filtered out using a notch filter [63]. Once these noises are removed, the neuronal signals can more easily be discriminated.

Two classes of filters analyzed for investigation, in this study, which include the Fast Fourier Transform (FFT) and the Discrete Wavelet Transform (DWT). Early EEG filtration utilizes FFT, however, this approach limits the spectrum of the EEG signal to four frequency bandwidths, including delta, theta, alpha, and beta [64]. The later development of DWT and numerous other methods has increased the popularity of the wavelet transform, as computation has become easier and more accurate. In recent literature, the wavelet has been a common method used for feature extraction. Based on its recommendation from various sources and the performances in [29][65], the DWT is used in the feature extraction phase of this analysis.

The DWT more accurately reconstructs higher order polynomial functions algebraically with its unique waveform properties. Based on a multi-resolutions operation, the wavelet characteristics of an adjustable window allow the extraction of all the components for every position by scaling and shifting. Consequentially, this achieves better representation of the original signal than would be possible with sinusoidal based waveforms as in the case of the FFT [22]. Additionally, the DWT retains the spectral analysis capabilities with of the FFT because features can be the signal can be converted into the frequency domain where it is reduced. Then frequency fixed information can be extracted from details and coefficients, such as energy, of the waveform representing the transient time course on a compressed scale. Thus the DWT is highly effective for feature extraction and signal representation, as is demonstrated in Chapter 4.

Accurate reconstitution of bio-signals, however, requires proper selection of the mother wavelet. Shifting and scaling different wavelets will produce different representations of the signal, and therefore different resolution. The most popular class, the Debauchee wavelet, has properties that are ideal for a lot of applications. Based on high performance in the literature review, in [29][66], a Debauchee mother wavelet with 2 vanishing moments was selected (‘db2’) as the wavelet; although, the Symlet family (rather intuitively, based on properties of asymmetry, orthogonality, and biorthogonality) is another good choice because and it has comparable results in literature [29][67]. The Coiflet family even produced higher classification accuracy in some studies.

The Wavelet Transform analysis performs scaling and shifting a function called the mother wavelet to produce the daughter wavelets and is represented as follows [68]:

$$T(a,b) = \frac{1}{\sqrt{a}} \cdot \int_{-\infty}^{+\infty} x(t) y^{*}\left(\frac{t-b}{a}\right) dt \tag{24}$$

where ψ^* is the complex conjugate of the mother wavelet $\psi_{a,b}$ is the daughter wavelet, $a^{-1/2}$ is the constant of energy normalization, l is the location factor, and a is the dilation factor. We use the convolution theorem to express the integral as a product in Fourier space.

The Continuous Wavelet Transform scales continuous data, whereas setting the dilation and translation factor to discrete integer $(a, b \geq 1)$, produces the Discrete Wavelet Transforms [68]

$$\psi_{m,n}(t) = \frac{1}{\sqrt{a_0^m}} \psi\left(\frac{t - nb_0 a_0^m}{a_0^m}\right) \tag{25}$$

where m and n are the wavelet dilation and translation. The variable $k, t\ (\in Z)$ and t is the time localizations. The dyadic grid wavelet is obtained by substituting the constants $a_0 = 2$ and $b_0 = 1$ into (25).

$$\psi_{m,n}(t) = 2^{-m/2} \psi(2^{-m} t - n) \tag{26}$$

The discrete wavelets are chosen to be orthonormal and as such they are normalized to have unit energy,

The formulation of the DWT gives way to the Multi-Resolution Analysis (MRA), which is schematically shown in Fig. 7,

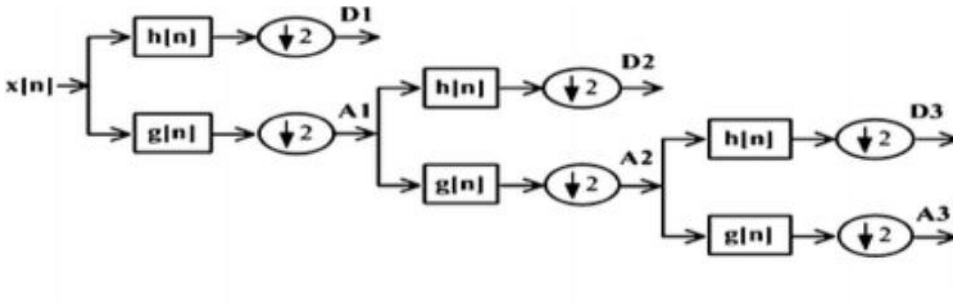


Fig. 7. DWT sub-band decomposition [69]

Signal decomposition begins with specification of the waveform in terms of the low-pass filter h, which satisfies the quadrature mirror filter condition [17],

$$H(z)H(z^{-1})+H(-z)H(-z^{-1})=1 \tag{27}$$

where $H(z)$ is the z-transform of h. Its associated high pass-filter can be represented as,

$$G(z)=zH(-z^{-1}) \tag{28}$$

Consequentially, a sequence of filters is constructed with increasing lengths,

$$H_{i+1}(z)=H(z^{2^i})H_i(z) \tag{29}$$

$$G_{i+1}(z)=G(z^{2^i})H_i(z), i=0,1,...,I-1 \tag{30}$$

The initial condition satisfies $H_0(z) = 1$, and upscale filtration is by a factors of 2, denoted $[.]_{2\uparrow}$, with equally sampled discrete times, k.

$$h_{i+1}(k)=[h]_{\downarrow 2^i}h_i(k) \tag{31}$$

$$g_{i+1}(k)=[g]_{\downarrow 2^i}h_i(k) \tag{32}$$

The normalized wavelet and scale basis functions are defined as,

$$j_{m,nl}(t)=2^{-m/2}h_i(2^{-m}t-n) \tag{33}$$

$$y_{i,l}(t)=2^{-m/2}g_i(2^{-m}t-n) \tag{34}$$

where $2^{i/2}$ is the inner product normalization, i is the translation parameter, and l is the translation and scale parameters, respectively. Thus,

$$a_i(l)=x(k)j_{i,l}(k) \tag{35}$$

$$d_i(l)=x(k)y(k) \tag{36}$$

where a and d are the approximations and detail coefficients. Thus concludes the derivation.

3.3.2: Blind Source Separation

The composition of source activities at the same location of the scalp depends on orientation, distance, and amplitudes of cortical origins across the cortex [70]. As a result, area restricted brain activity can be identified using these pseudo-spatial filtration techniques as such. By assuming that the channels are combinations of transient signals traveling through the brain, skull, and scalp, artifacts such as those electrooculographic (EOG) and electromyographic (EMG) existing in the 0.01 to 100 Hz frequency range of EEG, these sources can be removed [43][44]. A sufficient starting point for the discussion of spatial filtration of EEG is a review of the collection of algorithms under the title of Blind Source Separation (BSS).

The objective of BSS is to remove the correlation from the data and produce separation among signal sources. Let $\mathbf{s}(t) = \{s_1(t), \ldots, s_n(t)]^T$ be the signal sources and $\mathbf{x}(t) = [x_1(t), \ldots, x_m(t)]^T$ be the m observed mixtures of signals at the electrodes, both on the time course $t=1,2,\ldots,L$, then the signal at the j'th electrode can be modeled as [72]:

$$x_j(t) = \sum_{i \in \Gamma_{EMG}} a_{ji} s_i(t) + \sum_{i \in \Gamma_{EEG}} a_{ji} s_i(t) \tag{37}$$

where a_{ij} is the i'th source transfer coefficient to the j'th electrode and Γ is the set of all electromyographic and electroencephalographic indexes, respectively. The equation can be simplified in the following way [71],

$$\mathbf{x}(t) = \mathbf{x}_{j,EEG}(t) + \mathbf{x}_{j,EMG} = \mathbf{A}_{EEG}(t)\mathbf{s}_{EEG}(t) + \mathbf{A}_{EMG}(t)\mathbf{s}_{EMG}(t) \tag{38}$$

where **A** is a matrix of the transfer coefficients. In this representation, the neuronal components can be separated from the non-neuronal if the artifacts are located.

Automated Artifact Removal (AAR) is a subdivision of BSS that indirectly spatial filters EEG and may be used for automated EOG and EMG artifact correction. It involves finding a new projection **W** of the data, called an *unmixing matrix*, that maximizes the correlation between sources, as is demonstrated by [72]:

$$\mathbf{WX} = \mathbf{A}_{EEG} \tag{39}$$

After multiplication with the original data, **X**, **W** reveals the matrix time course **A** of neuronal components. The purpose of this transformation is to map the data into another space.

Contemporary AAR methods such as Second Order Statistics (SOS), including the Second Order Blind Identification algorithm (SOBI), assume that the original source signals are uncorrelated and time-lagged covariance matrices. SOBI, for example, uses a non-zero time delay auto correlation (used in this study for EOG removal based on its performance in [43]). Under this assumption, the mixing matrix is diagonal with a set of *p* cross-correlation matrices [73]:

$$R(t_i) = E[x(t)x(t - t)^T] \tag{39}$$

where $i = 1, 2, ..., p$ and $E[\cdot]$ is the expectation operator. Using Sevcik's algorithm, the waveform coordinate (x_i, y_i) are mapped into a unit square by $x_i^* = x_i / x_{\max}$ and $y_i^* = (y_i - y_{\max}) / (y_{\max} - y_{\min})$ where $x_{\max}$ is the maximum x_i and $y_{\max}$ and $y_{\min}$ are the maximum and minimum of y_i, respectively.

The first step of SOBI begins with decomposition of correlative data windows (with windows greater than 0.25xm^2, as per [43], where *m* is the number of electrodes). This separation allows the analysis to tolerate the non-stationary nature of the signal.

Afterwards, components are identified using fractal dimension (FD) (to measures complexity). FD is calculated as [73]:

$$FD = 1 + \frac{\ln(l)}{\ln(2(n - 1))} \tag{40}$$

where l is the total length of the waveform and n is the number of sample points of the signal. EEG waveforms are flatter and have a wide spectrum, and thus have a higher FD, whereas, EOGs have lower FD, and thus discrepancies may be discerned and artifacts may be removed.

The electromyography (EMG) artifacts can be eliminated through means of a second AAR algorithm designated the Canonical Correlation Analysis (CCA) which attempts to recover the original signal sources in $\mathbf{S}(t)$ by setting a time lag to ensure that each electrodes time source signals are related to each other [74]. With each successive new pair of variables, the pair is maximally correlated with each other and uncorrelated with the first pair. Each consecutive variate yields time courses that have autocorrelation among all possible linear combinations of time courses and are uncorrelated with the previously obtained time course. Because muscle artifacts yield more properties having less autocorrelation, EMG and EOG are expected to be present in the lowest auto correlated CCA sources, setting these sources to zero removes the noise.

For CCA, first it is assumed is that the sources are mutually uncorrelated and maximally auto correlated and that the mixing matrix is square. Therefore, $\mathbf{Y}$ is selected to be a temporarily delayed version of the original data matrix $\mathbf{X}$, $\mathbf{Y}(t) = \mathbf{X}(t\text{-}1)$. Next, the average of each row is subtracted; producing two vectors whose projection correlation onto the basis vectors are mutually maximized according to [44]:

$$\mathbf{u} = w_{x_1}\mathbf{x}_1 + \ldots + w_{x_k}\mathbf{x}_k = \mathbf{w}_x^T\mathbf{X} \tag{41}$$

$$\mathbf{v} = w_{y_1}\mathbf{y}_1 + \ldots + w_{y_k}\mathbf{y}_k = \mathbf{w}_y^T\mathbf{Y} \tag{42}$$

The weight vectors $\mathbf{w_x} = [w_{x1}\ldots,w_{xK}]^T$ and $\mathbf{w_y} = [w_{y1}\ldots,w_{yK}]^T$ maximize the correlation ρ between the variables **u** and **v**:

$$\max r(\mathbf{u},\mathbf{v})$$

$$\mathbf{w}_x\mathbf{w}_y = \frac{E[\mathbf{uv}]}{\sqrt{E[\mathbf{u}^2]E[\mathbf{v}^2]}}w$$

$$= \frac{E[(\mathbf{w}_x^T\mathbf{X})(\mathbf{w}_y^T\mathbf{Y})]}{\sqrt{E[(\mathbf{w}_x^T\mathbf{X})(\mathbf{w}_x^T\mathbf{X})]E[(\mathbf{w}_y^T\mathbf{Y})(\mathbf{w}_y^T\mathbf{Y})]}}$$

$$= \frac{\mathbf{w}_x^T C_{xy}\mathbf{w}_y}{\sqrt{(\mathbf{w}_x^T C_{xx}\mathbf{w}_x)(\mathbf{w}_y^T C_{yy}\mathbf{w}_y)}} \tag{43}$$

where $\mathbf{C_{xx}}$ and $\mathbf{C_{yy}}$ are the auto covariance matrices and $\mathbf{C}_{xy}$ is the cross covariance matrix, of **X** and **Y**. By setting the derivatives of (43) to with respect to $\mathbf{w_x}$ and $\mathbf{w_y}$ equal to zero, after some algebraic manipulation, the following equation is presented:

$$C_{xx}^{-1}C_{xy}C_{yy}^{-1}C_{yx}\hat{\mathbf{w}}_x = r\hat{\mathbf{w}}_x$$

$$C_{yy}^{-1}C_{yx}C_{xx}^{-1}C_{xy}\hat{\mathbf{w}}_y = r\hat{\mathbf{w}}_y \tag{44}$$

which is used to find the maximum correlation.

Lastly, Independent Component Analysis (ICA) and is one of the more successful contemporary BCI signal processing [75]. ICA also applies BSS to remove artifact and improve SNR (neuronal vs. non-neuronal) of tasks related EEG signals, and it may even be used to aid in electrode selection. ICA works to remove the correlation of the data using a mixing matrix, just like the Principle Component Analysis (PCA) and the AAR

methods previously described, however a key difference is that SOS methods use second order statistical methods whereas ICA uses all order statistics and ICA makes no assumptions about the mixing matrix.

ICA seeks projections (utilizing the same unmixing matrix from the BSS problem), $\mathbf{W}^{-1}$, of the data that maximizes their statistical independence. Every Independent Component (IC) is a weighted sum of the signals recorded at all scalp channels and every channel signal is a weighted sum of the projected activities of the ICs:

$$X=W^{-1}A_{EEG} \tag{45}$$

As with AAR, ICA attempts to reconstruct a multi channel source using a mixture of those sources to allow for the separation of muscle artifacts, such as ocular. This method, however, uses unsupervised learning to solve the BSS mixing matrix problem (therefore, it is important to note that depending on learning rate, error stopping criteria, and various other factors, ICA performance may change).

There are many ICA methods available, with different measurements statistical independence, including the following: INFOMAX, JADER, and FASTICA. In this report, 23 channels are kept for spatial filtration, and due to INFOMAX's stability and JADER and FASTICA's limitations, INFOMAX is the superior choice. FASTICA computes individual components 1-by-1 and is therefore slower than INFOMAX. In addition, JADER can only handle roughly 50 channels for storage reasons [76], putting it at a disadvantage for high density.

3.3.3: Neural Oscillations Analysis

Event-Related Potentials (ERP) is a term often used in literature to describe the measured brain responses to sensory, cognitive, and motor events and covers the subsequent analyses which includes the following: Event-related synchronization and desynchronization (ERS/ERD) and Movement Related Cortical Potentials (MRCP). The first (ERS/ERD) analyzes neuronal activity correlated to locations C3, CZ, and C4 of the scalp (near the somatosensory cortex [65]), whereas the latter (MRCP) often contains activity from these three electrode placements as well as C1, C2, FC3, FCZ, and FC4 [77], above the motor cortex as well.

ERD/ERS quantifies motor events through changes in rhythms of the mu (8-13Hz) and beta (13-30 Hz) frequency bands when motor activity is preformed or imagined. Frequency locked EEGs decrease amplitude prior to and during execution of movement in a process referred to as desynchronization, see Fig. 8. After the movement is concluded, the beta region cortical potentials increase in a process recognized as synchronization [78]. Left hand movement corresponds to alpha and beta ERS/ERD activity in the right hemisphere of the brain, whereas right hand movement corresponds to the left hemisphere. This poses a starting point for feature extraction. Tracking changes in the rhythm fluctuation, for example, through energy percent calculation at each of the frequency bandwidths, generated from different sources is hypothesized to aid in the differentiation of the left and right hand motor movement.

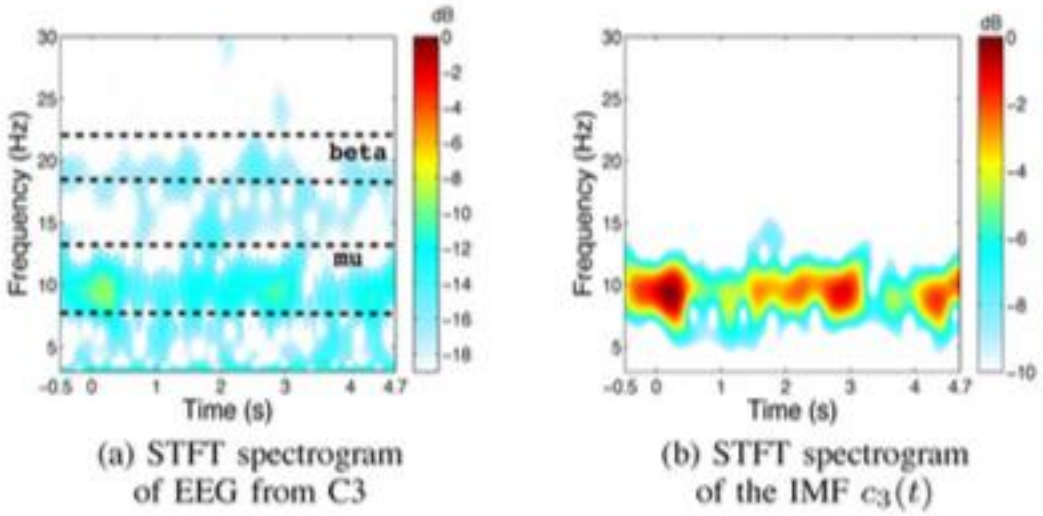


Fig. 8. Changes in time frequency power distribution calculated for ERD activity [80]

Movement Related Cortical Potentials (MRCP) refers low frequency (0 – 5 Hz) negative shifts that occur 1.5-2 seconds before the onset of voluntary movement [80]. In order to extract more features for the neural network, the same electrodes may be used as to capture the cortical activity corresponding to motor movement imagery, which happen to be located above the primary motor cortex, near the somatosensory cortex. Using ERD/ERS analysis in addition to MRCP electrode placements with energy percentage calculated from wavelet coefficients through the DWT (for mu and beta waves), it is assumed that the features produced from the characteristics can be implemented to successfully classify the motor events using the TSKFNN.

3.4: Feature Translation

Neuro-fuzzy systems have been emerging amidst the development of soft computing methods [81]. Methods such as fuzzy logic, genetic algorithms (or other constituents of soft computing) are able exploit the tolerance for imprecision to achieve a low cost, tractable solution, making these systems extremely robust [82]. The use of fuzzy logics uncertainty principles allows for tolerance to imprecision, based on the formulation of uncertainty into rules of calculus. This can be of special use in non-linear, multivariate systems when exact output can be generated for linguistic like expressions [83]. This allows fuzzy logic and neural networks to synergize and machine to be taught classification expertise, thus making it a superior tool for feature translation.

3.4.1: Takagi-Sugeno-Kang Fuzzy-Inference

Fuzzy inference systems are governed by a set of linguistic rules. Statements such as *IF A THEN B* in conjunction with mathematical principles assist in the formulation of decisions that would be otherwise difficult to quantify. Based on fuzzy principles, or uncertainty in decisions, networks can produce output to respond to a situation such as:

If color is blue and object is round, then object is blueberry

where *blue* and *round* are not indefinite descriptions. Based on an experienced set of rules, precise output is calculated [83]. Thus, this linguistic method is thus applicable to the treatment of amputation.

The Takagi-Sugeno-Kang fuzzy inference model emerged to fulfill the need for a simple linguistic model for multivariable control in which the dimensionality is large (accurate or ‘defuzzified’ output). First a membership function is defined to represent the

accuracy of description of a particular subject, or feature. Degree of membership of a particular instance to another is defined, such as degree of an object to the color blue. Combining a set of instances of an object belonging to different categories of data, a general classification statement can be made. The Sugeno style model is defined such that fuzzy implication *R* is of the format:

$$R\text{: If } f(\text{x}_1 \text{ is } A_1, \ldots, x_k \text{ is } A_k) \text{ then } y = g(x_1, \ldots, x_k) \tag{46}$$

The Takagi-Sugeno method defuzzifies the output so that the consequence part of the equation is quantifiable, and from this organization algorithms such as TSKFNN have emerged for machine learning application in output devices, each with their own set of advantages.

The TSKFNN consists of two stages: feed forward and weight update. In the feed forward period, the inputs undergo fuzzification as degrees of membership to multiple mathematical functions are calculated as follows. Assuming that the membership can be modeled as a normal distribution:

$$M_i^j(\cdot) = \exp\left\{ - \frac{(x_i - m_j)^2}{s^2} \right\} \tag{47}$$

For the case of an asymmetric membership function, the equation becomes piecewise. Using the *AND* logic represented by element-by-element multiplication, the rule layer is developed to compare different conditions. The Takagi-Sugeno style inference mechanism performs a linear combination of the consequent parameters $\{c_0, c_1, \ldots, c_n\}$:

$$f_1 = T_k = c_0 y_0 + c_1 y_1 + \ldots + c_n y_n \tag{48}$$

(where the bias is in the first term and *n* is the number of inputs), performs product *AND* in the consequent layer so that the input is a linear sum with scalars representing the consequent parameters. The output layer defuzzifies the input, where the connective resistance going into the node constitutes firing strength [16], and generates a crisp output or exact solution.

3.4.2: TSK Fuzzy Neural Network

The Takagi-Sugeno-Kang Fuzzy Neural Network updates the network to produce the desired output based on error feedback to classify the segments from the features extracted in the previous stage [16]. Adaptive Neural Fuzzy Inference System (ANFIS), Fig. 9, is another name for the type of TSKFNN that is implemented in the Fuzzy Logic application package of MATLAB that is employed for this study. In the ANFIS model, the input features arrays feed through the network changing their strengths based on membership to the membership functions, and then the value is compared to the target output and error feeds back through the network and the weights are modified accordingly.

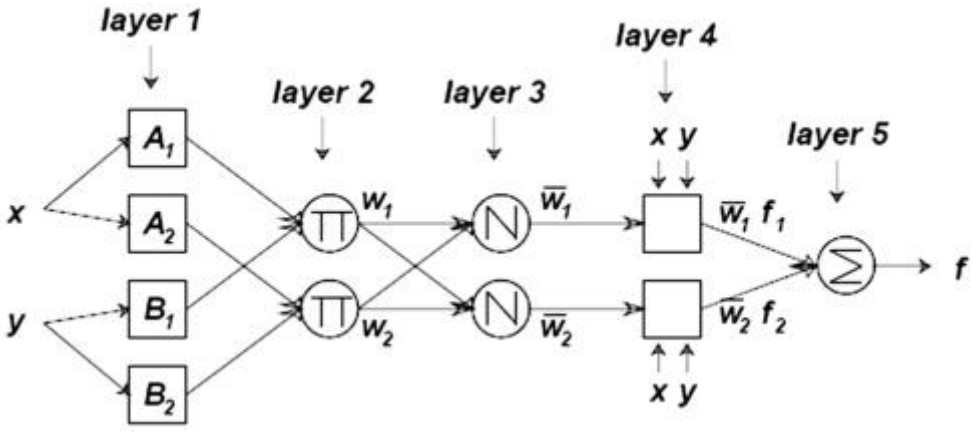


Fig. 9. ANFIS Takagi-Sugeno-Kang Style Inference Neural Network [18]

A. Back Propagation, Feed Forward [16], [17].

In the forward feed process, the output is computed from the Takagi-Sugeno-Kang Fuzzy Reasoning. The premise layer uses product AND to combine the terms and the consequence layers uses product *IMPLICATION*. The consequence parameters are updated through the LSE algorithm.

1) Layer 1 (Input Layer): The node input and output for the *i*'th node are represented as the following, in the input layer,

$$y_i^1(n) = x_i(n), i = 1,..., q \tag{49}$$

where $x_i(n)$ the input at the *n*'th iteration, and *q* represents the number of inputs.

2) Layer 2 (Membership Layer): In the membership layer, the degree of membership to each function is calculated using the Gaussian distribution.

$$y_j^2(n) = M_i(y_i^1(n)) = \frac{1}{1 + \left[\frac{y_i^1(n) - m_j}{s_{ij}^2} \right]^2 b_i}$$

$$= \exp\left\{ -\left(\frac{y_i^1(n) - m_j}{s_{ij}^2} \right)^2 \right\}, j = 1, 2,..., s \tag{50}$$

where m_j is the membership function center, *s* is the number of membership functions multiplied by *q* (inputs), and σ_{ij}^2 is the variance of the Gaussian membership curve.

3) Layer 3 (Rule Layer and Consequent Parameter Formulation): Every node in the rule layer multiplies the incoming signals and sends the product out. The *k*'th node formula is as follows,

$$y_k^3(n) = \tilde{O}_{j=1}^{l} w_{jk}^3 y_j^2 \tag{51}$$

where w_{jk}^3 is the weight of the membership layer to the rule layer, and is equal to 1. Each node is indicative of firing strength, normalized as,

$$\bar{y}_k^3(n) = \frac{y_k^3(n)}{\sum_{i=1} y_k^3(n)} \tag{52}$$

The TSK type fuzzy inference layer is designed by linearly transforming the input signals with the consequent parameters, denoted c_{ik} as described below.

$$T_k(n) = \left(\sum_{i=0} c_{ik}(n) y_i(n) \right) \tag{53}$$

The variable c_{0k} is the bias, which can be set to 0. T_k is the output of the linear combination.

4) Layer 4 (Consequence Layer): The consequence layer is formulated as the product of the output of the normalized rule layer with the linearized input signal.

$$y_k^4(n) = \bar{y}_k^3(n)\left(T_k(n)\right) = \frac{y_k^3}{\sum_i y_k^3} \left(\sum_i c_{ik}(n) y_i^1(n) \right) \tag{54}$$

5) Layer 5 (Output Layer): This single node acts as the defuzzifier and is mathematically represented in the following equation:

$$O_i^5 = \sum_{k=1} w_k^5 y_k^4 \tag{55}$$

Least Squares Estimate (Consequent Layer Update): As part of the hybrid learning routine, the consequent layers update according to the LSE method described in equation (72) and (73).

B. Back Propagation Using Delta Learning [16], [17]

Premise parameters update via propagation of the error term, where the error function E is defined below. The output of the neural network is a value that is compared

to a binary value (+1 or -1) that would be used upon learning to determine whether the movement corresponds to left hand or right hand. The error is calculated using least square error.

$$E = \frac{1}{2}\left(w - O_i^5\right)^2 = \frac{1}{2}e^2 \tag{56}$$

The value ω is the desired output and target forward propagation, respectively. The update is given as follows.

1) Layer 5: The local gradient is the only term calculated:

$$\delta_0^5 = -\frac{\partial E}{\partial y_0^5(n)} = -\frac{\partial E}{\partial e}\frac{\partial e}{\partial y_0^5(n)} \tag{57}$$

2) Layer 4: The error propagation can be written in the following way, in the consequence layer,

$$\delta_k^4 = -\frac{\partial E}{\partial y_k^4(n)} = -\frac{\partial E}{\partial y_0^5}\frac{\partial e}{\partial y_k^4(n)} = \delta_0^5 w_k^5 \tag{58}$$

wherein the weights are fixed.

3) Layer 3: The consequence parameters do not update in this layer. However, the local gradient is derived as:

$$\delta_k^3 = -\frac{\partial E}{\partial y_k^3(n)} = -\frac{\partial E}{\partial y_0^5(n)}\frac{\partial y_0^5(n)}{\partial y_k^4(n)}\frac{\partial y_k^4(n)}{\partial y_k^3(n)} = \delta_k^4 T_k(n) \tag{59}$$

4) Layer 2: In this layer, the error term is calculated with the following equation:

$$\delta_j^2 = -\frac{\partial E}{\partial y_j^2(n)} = -\frac{\partial E}{\partial y_0^5(n)}\frac{\partial y_0^5(n)}{\partial y_k^4(n)}\frac{\partial y_k^4(n)}{\partial y_k^3(n)}\frac{\partial y_k^3(n)}{\partial y_j^2(n)} = \delta_k^4 \sum_k \delta_k^3 y_k^3 \tag{60}$$

The premise parameters $\{\sigma_j, m_j\}$ are updated using the chain rule, with learning rates η_m and $\eta_{\sigma j}$ for membership function average and deviation. The learning rate, η, is updated according to the generic parameter, α, and can be derived as follows,

$$\Delta a = -\eta \frac{\partial E}{\partial a} \tag{61}$$

Therefore the membership center and standard deviation update respectively, according to the following formulas:

$$\Delta m_j = -\eta_m \frac{\partial E}{\partial m_j} \tag{62}$$

$$= -\eta_m \frac{\partial E}{\partial y_0^5(n)} \frac{\partial y_0^5(n)}{\partial y_k^4(n)} \frac{\partial y_k^4(n)}{\partial y_k^3(n)} \frac{\partial y_k^3(n)}{\partial y_j^2(n)} \frac{\partial y_j^2(n)}{\partial m_j(n)} \tag{63}$$

where Δm_j represents the change that the membership function center undergoes upon error feedback each iteration and,

$$\Delta \sigma_j = -\eta_{\sigma_j} \frac{\partial E}{\partial \sigma_j} \tag{64}$$

$$= -\eta_\sigma \frac{\partial E}{\partial y_0^5(n)} \frac{\partial y_0^5(n)}{\partial y_k^4(n)} \frac{\partial y_k^4(n)}{\partial y_k^3(n)} \frac{\partial y_k^3(n)}{\partial y_j^2(n)} \frac{\partial y_j^2(n)}{\partial \sigma_j(n)} \tag{65}$$

$\Delta\sigma_j$ is the change of the standard deviation of the membership function curve. The equation for update for each is given in the following computation:

$$m_j(n+1) = m_j(n) + \Delta m_j \tag{66}$$

$$\sigma_j(n+1) = \sigma_j(n) + \Delta \sigma_j \tag{67}$$

The learning rate can be written in terms of the step size, *k*, also known as is the length of each gradient transition.

$$h = \frac{k}{\sqrt{\sum_a \left(\frac{\partial E}{\partial a}\right)^2}} \tag{68}$$

The learning rate of the membership function centers is derived below.

$$h = \frac{k}{\sqrt{\sum_a \left(\frac{\partial E}{\partial y_0^5(n)} \frac{\partial y_0^5(n)}{\partial y_k^4(n)} \frac{\partial y_k^4(n)}{\partial y_k^3(n)} \frac{\partial y_k^3(n)}{\partial y_j^2(n)} \frac{\partial y_j^2(n)}{\partial m_j(n)}\right)^2}} \tag{69}$$

$$h = \frac{k}{\sqrt{\sum_a \left(\frac{\partial E}{\partial y_0^5(n)} \frac{\partial y_0^5(n)}{\partial y_k^4(n)} \frac{\partial y_k^4(n)}{\partial y_k^3(n)} \frac{\partial y_k^3(n)}{\partial y_j^2(n)} \frac{\partial y_j^2(n)}{\partial s_j(n)}\right)^2}} \tag{70}$$

The weights parameters are updated according the hybrid LSE-BP method described in 3.4.3. Thus concludes the derivation of the Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN).

3.4.3: Hybrid Learning Algorithm

A key distinction between the various frameworks of fuzzy neural networks is the premise and consequent layer parameter. There are several different procedures for updating these parameters including Back Propagation (BP), Least Square Estimate (LSE), or a combination of these and various others. Between the BP and LSE, there are multiple ways of updating the parameters including (descending in complexity):

1. Updating all parameters via gradient descent;
2. Applying LSE once at the very beginning to get the initial values of the consequent parameters, then applying gradient descent to update all of the parameters;
3. Applying LSE to update the consequent parameters and gradient to update the premise parameters; and
4. Applying the Kalman filter algorithm to update all parameters using approximate LSE only.

The gradient descent method is slow and frequently gets trapped in local minimums. However, the LSE method can be computational inefficient. Furthermore, the hybrid LSE-gradient descent method permits quick convergence, but other methods may be used to hasten convergence. In [16], the authors use a convergence analysis with the delta rule based on the discrete type Lyapunov function to counter poor convergence. BP has very low complexity in contrast to the other methods and is, therefore, the most efficacious. It recursively updates the weights according to the delta rule allowing

convergence to a local minimum, assured by steepest descent, to achieve the highest classification performance at the lowest extrema.

One deficiency of this method, however, in addition to getting caught at the wrong local minima, is that instability results from improper learning rate parameterization [32]. With small learning rates, the online-learning system can approximate gradient descent and remain stable. However, if the learning rate parameter is too small, the system will not learn quickly and will perform poorly for small training sets. If it is too big, the system will vacillate unstably. These issues will cause the system to perform poorly on the test data points. As a result, a correct learning rate needs to be selected for optimal results.

Let the consequence parameters $\{c_1, c_2, \ldots\}$ be X^T and A be the output of the previous layer feeding into the consequence layer. Let y be the output, then, AX = B. So $X = A^{-1}B$. However this is an over determined solution because the data is usually much larger than the number of features. So instead we use a least squares estimator X^* and the equation become $AX^* = `B$. This formulation of the solution is often recognized as the pseudo-inverse. It often appears in textbooks and is written as [84]:

$$X^* = (A^T A)^{-1} A^T B \tag{71}$$

where *A* contains a row for every pattern of training data, *X* has consequence parameters, in the hybrid case and *B* is target output from the training data. These matrices are of sizes *(PxM), (Mx1), (Px1),* respectively, where *P* and *M* are the number of training sets and the number of consequence parameters. Sequentially, the formula can be represented online as [82]:

$$X_{i+1} = X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \tag{72}$$

$$S_{i+1} = \frac{1}{\lambda}\left[S_i - \frac{S_{i+1} a_{i+1} a_{i+1}^T S_i}{\lambda + a_{i+1}^T S_i a_{i+1}} \right] \tag{73}$$

where λ is the forgetting factor placing less emphasis on older values, S is the covariance matrix, and a_i and b_i correspond to the i'th row of A and B respectively. The forgetting factor increases adaptability, allowing online learning approximation. This allows for samples misrepresenting the data lose strength as the new test data becomes available [24]. The initial conditions are of the form $X_0 = 0$ and $S_0 = kI$, where k is a large positive constant and I is the identity matrix with rows and columns of size M. It is important to note that too little decay causes instability.

Hybrid LSE-gradient decent uses back propagation to tune the hidden layers parameters (consequent parameter) and LSE to identify the output layer parameters (premise parameters), demonstrated in Table II.

Table 5. Summary of the hybrid algorithm [17]

	LSE	BP
Premise Parameters	Fixed	Gradient Descent
Consequent Parameters	Least Squares Estimate	Fixed
Signals	Node Outputs	Error Rates

Fine-tuning the membership functions is another option to consider. Linguistically speaking, fine-tuning allows for the adaption of subjective description for ill-defined concepts, as is done with the asymmetric membership functions in the convergence analysis [80]. In most applications, the use of fuzzy logics linguistic structure necessitates only subtle knowledge of the background of the presented situation and therefore enhanced representation may not be reflective of the data set, in which case the membership functions should be fixed constant. However, fine-tuning the membership functions can make the model more adaptive and perform better on new data and give more precise feedback.

3.4.4: Subtractive Clustering Algorithm

One of the problems with the TSKFNN is what is often denoted the *curse of dimensionality*. This is a term often used to describe the condition wherein a small increase in the number of input in the input layer leads to an exponential increase in the number of nodes in the neural network and therefore computational complexity [85]. Each new input must be compared with a variety of other. The addition of a few more nodes increases the number of possible connections quite drastically. In order to reduce the number of calculations, a few methods have been proposed including subtractive clustering and fuzzy c-means clustering. These methods all cluster the data to find a pattern between groups of data.

Membership a function are constructed with a degree of probability of all of the input features (essentially a point) being part of one cluster (another point) as opposed to one input (a feature) being a degree of probability of being of the type of one class (feature of class). This significantly decreases the number of nodes in the premise layer and, therefore, the total number of calculations. The main difference between these methods is the method of determining the clusters. Wherein subtractive clustering requires that cluster centers be a minimum distance [86], fuzzy c-means clustering lets the user select a number of clusters and the centers and distance does not matter [87]. Subtractive clustering is used in this study because it has been demonstrated to have higher performance on most data types [86][88].

Subtractive clustering finds the highest density of points and creates one cluster center. First, the points are scaled to [0,1] in each dimension. Next, each point $z_j=(x_j,y_j)$ is assigned a potential P_j that is determined based on its distance from all of the other points

[90]:

$$P_i^* = \sum_{j=1}^{n} e^{-\alpha\|x^i - x^j\|^2} \tag{74}$$

where $\alpha = \gamma / r_a$ between the *i'*th and the *j'*th data point, *x* is the data point, γ is a variable commonly set to 4, and r_a is the positive cluster radius. When points are close together, the potential is high and the probability of being a cluster center increases. The highest potential point becomes the cluster center $c_1 = (d_1, e_1)$. Subsequently, the potential is recalculated for all of the other points excluding the first cluster center, in [89],

$$P_i^* = P_i^* - P_k^* \sum_{j=1}^{n} e^{-\beta\|x^i - c^k\|^2} \tag{75}$$

where $\beta = 4 / r_b^2$ and $r_b = r_a * \eta$.

The next cluster center has to be a minimum distance from the last (defined by user chosen radii), and is selected based on probability highest probability (density). The cluster becomes a cluster center if it has the highest potential and fulfills the given limitation [90]:

$$\frac{d_{\min}}{r_a} + \frac{P_k^*}{P_1^*} \geq 1 \tag{76}$$

The variable $d_{\min}$ is the minimum distance between c_1 and the preceding cluster centers. If the following criterion is met, the clustering ends [91]:

$$P_k^* < \varepsilon P_i^* \tag{77}$$

where ε is the rejection ratio. Finally, the membership function is described as [91]:

$$\mu_j^{ik} = e^{-\alpha\|x_j^i - c_j^k\|^2} \tag{78}$$

An illustration of the concept is given in Fig. 10.

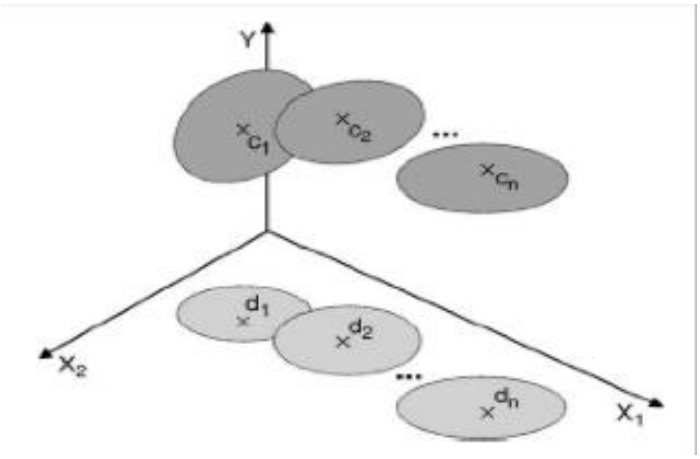


Fig. 10. Subtractive clustering schematic [90]

It is demonstrated that the subtractive clustering method decreases dimensionality of the feature space because a multidimensional space is calculated from one feature as opposed to many. This allows for an analysis of a multidimensional feature space that would otherwise be too complex to calculate.

CHAPTER 4: APPROACH & RESULTS

The focus of Chapter 4 is the methods, Fig. 11, used and the results that are achieved. First the Finite Impulse Response (FIR) with Hamming windowing is used to eliminate noise with a band pass filter. Next, the Automated Artifact Rejection (AAR) and Independent Component Analysis (ICA) algorithms isolate signal components of interest. The signals are then decomposed using the DWT. Finally, the energy percentage of each band is calculated to produce features used in the Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN). This network translates the input into to a crisp value that is processed using a threshold function to classify hand contraction.

Graphical Abstract:

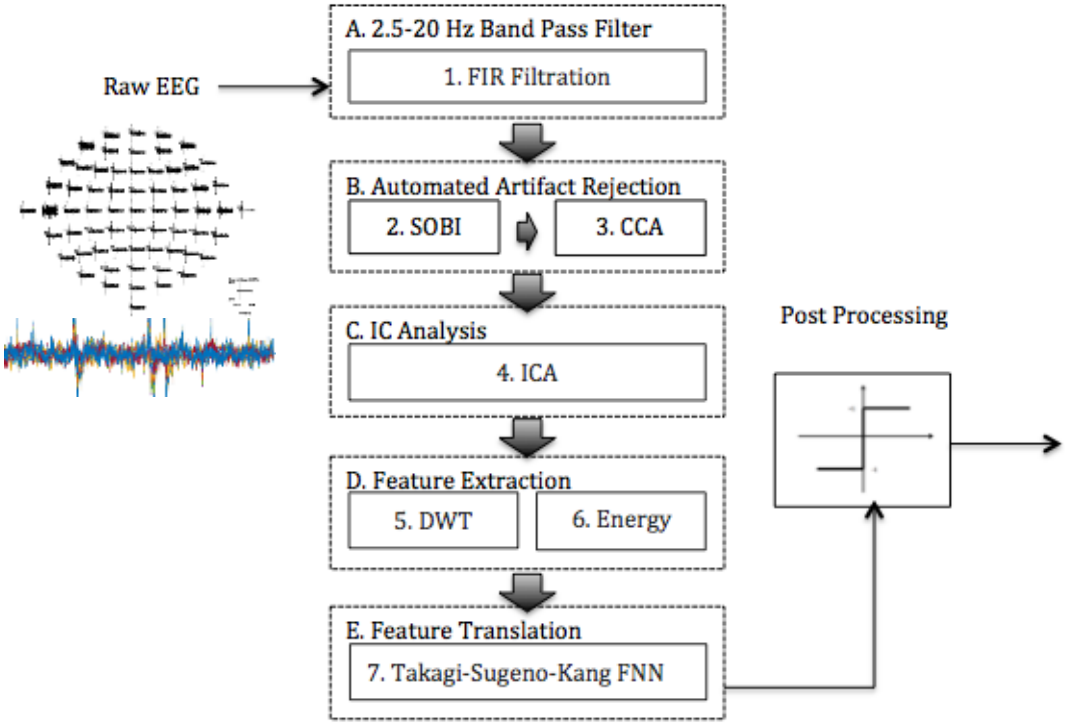


Fig. 11. This figure illustrates the process through which the noise is reduced and the signal is recognized.

First, Chapter 4 begins with a brief introduction of EEGLAB and the various functions that will be used. Each function will include the inputs and outputs. Consult Appendix B for demonstrations on how to implement the code.

4.1: EEGLAB Introduction

The EEGLAB software is an interactive MATLAB toolbox for processing electrophysiological data. This data may be either continuous or event-related time course and has tools for Independent Component Analysis (ICA), time-frequency analysis, and other methods of signal processing such as Automated Artifact Rejection (AAR) [90]. EEGLAB contains an interactive graphical user interface (GUI) for users to process their data as well as command line scripting functions for advanced programmers. Because electroencephalographs (EEGs) are notoriously noisy, this study makes extensive use of EEGLAB for signal processing.

A few plugins were installed to take advantage of the various computational capabilities of user developed software. The plugins that were downloaded for this study include the following: aar_master, and Biosig3.0.5, described below.

> ***BIOSIG3.0.5*** – import/export a wide variety of formatted data including European data format (EDF).
>
> ***AAR*** – implement state-of-the-art methods for automatic removal of ocular and muscular artifacts.

To obtain functionality, the folders were downloaded into the plugins folder of EEGLAB. After the plugins have been loaded, the following functions were used with the input indicated in parentheses:

pop_importdata()

> Description: file imports data from the specified format and creates a structure for storing information about the signal.

Inputs – 'data format': format of data (i.e. 'matlab,' string)

- 'nbchan': number of channels' (i.e. ['nchan',0] for all)
- 'data': file location (['data','file_path'], string)
- 'srate': sampling rate (i.e. ['srate',160] for this data, units Hz)
- 'pnts': how many time points to include (i.e. ['pnts',0] for all)
- 'xmin': first time point (['xmin',0] for initial)

Outputs – EEG: structure containing the previously specified information; rows are channels whereas columns are time points.

eeg_checkset()

Description: check the data to verify that all data in file has been updated

Inputs – EEG: structure containing *EEG.data* time course data

Outputs – EEG: structure containing *EEG.data* time course data

** Note – best to use after every function to make sure that the set updates

pop_chanevent():

Description: add event channel from one of the channels in *EEG.data* (i.e. channel 65 is annotated channel that has been replaced with values corresponding to the current event at each point in time)

Inputs – EEG: structure containing annotated *EEG.data* channel

- channel: the annotated channel number
- 'edge': edge corresponding to event region (i.e. ['edge','leading'])
- 'edgelen': length of the event region (i.e. ['edgelen',0] to include all event related time course for an epoch)
- 'duration': include duration (i.e. ['duration','on'], string)

Outputs – EEG: structure containing the old EEG information with newly specified information

pop_chanedit():

Description: edit channel information such as locations file; used for 2D or 3D viewing;

Inputs – EEG: structure containing the *EEG.data* time course data and channel information

- 'lookup': find the file containing the files to read channels (i.e. ['lookup','path'], string)
- 'load': load previous manually entered channel locations (for automation)
- 'plotrad': specify how much of channels to be observed from topographical map (i.e. ['plotrad',0.5] for purely topographical view)

Outputs – EEG: structure containing the old EEG information with newly specified information

pop_spectopo()

Description: Create log power spectrum chart

Inputs – EEG: structure containing the *EEG.data* time course data and channel information

- dataflag: process component channels/activations, 1/0 (i.e. 1)
- time range: epoch time range, in ms (i.e. [0 4093.75] because 4.1 seconds)

- process: work on either the mean single-trial 'EEG' spectra, the spectrum of the trial-average 'ERP', or plot 'BOTH' the EEG and ERP spectra (i.e. ‘EEG’)
- ‘percent’: data to be included in the computation of the spectra (i.e. [‘percent’,15], saves computation time)
- ‘freqrange’: x-axis, frequency range (i.e. [‘freqrange’,[0 50]], units Hz)

Outputs – spectopo_output: graphical information

pop_eegfiltnew()

Description: automatically filter the single epoched time course data

Inputs – EEG: structure containing the *EEG.data* time course data and channel information

- ‘nbchan’: number of channels’ (i.e. [‘nbchan’,0], for all)
- lower: high frequency cutoff (i.e. [2.5], units Hz)
- upper: low frequency cutoff (i.e. [20], units Hz)

Outputs – EEG: structure containing the old EEG information with filtered *EEG.data* time course

pop_reref()

Description: re-reference the window to a new mean

Inputs – EEG: structure containing the *EEG.data* time course data and channel

- ref: vector of new reference number (i.e. [] for average reference)

Outputs – EEG: structure containing the old EEG information with newly specified information

** Note: A 2.5 to 20 Hz band pass filter is utilized because it captures the Movement Relate Cortical Potentials (MRCP) and Event Related Desynchronization while diminishing the effect of the noise on Independent Component Analysis (ICA).

pop_autobsseog()

Description: Automated Electrooculargraphic (EOG) artifact correction method employing Blind Source Separation (BSS).

Inputs – EEG: input EEG data structure

- 'nbchan': number of channels' (i.e. ['nbchan',0], for all)
- win_length: analysis window length, ms (i.e. [EEG.xmax])
- win_size: shift analysis windows, ms (i.e. [EEG.xmax])
- bss_alg: name of the BSS algorithm being used (i.e. 'sobi')
- bss_opt: option to pass the BSS algorithm (i.e. {'eigratio', [1000000]}
- crit_alg: criterion name for rejecting components (i.e. 'eog_fd')
- crit_opt: criterion special options (i.e. 'eog_fd', {'range',[2 21]}

Outputs – EEG: structure containing the old EEG information with EOG artifact correct EEG signal.

**Note: Too large of an eigenvalue ratio will result in low removal rate of principle components, whereas too low will result in the contrary [43]. An eigenvalue ratio of 1.0E6 is arbitrarily chosen in this study because the developers of EEGLAB have selected it as an appropriate tradeoff default balance. A range of 2 to 21 Hz is selected because it contains the desired 2.5 to 20 Hz region.

pop_autobssemg()

Description: Automated Electromyographic (EMG) artifact correction method employing BSS.

Inputs – EEG: input EEG data structure

- 'nbchan': number of channels' (i.e. ['nbchan',0], for all)
- win_length: analysis window length, ms (i.e. [81.9188])
- win_size: shift analysis windows, ms (i.e. [81.9188])
- bss_alg: name of the BSS algorithm being used (i.e. 'bsscca')
- bss_opt: option to pass the BSS algorithm (i.e. {'eigratio', [1000000]}
- crit_alg: criterion name for rejecting components (i.e. 'emg_psd')
- crit_opt: criterion option (i.e. {'ratio', [10],'fs', [160],'femg', [15], 'estimator',spectrum.welch({'Hamming'}, 80),'range', [0 32]}, for a power ratio of 10, sampling frequency of 160 Hz, boundary for EEG and EMG of 15 Hz, Hamming window for power spectral estimation and 0 to 32 Hz correction of EMG artifaction)

Outputs – EEG: structure containing the old EEG information with EOG artifact correct EEG signal.

**Note: Again, An eigenvalue ratio of 1M is arbitrarily chosen in this study because the developers of EEGLAB have selected it as an appropriate tradeoff default balance. Too large of an eigenvalue ratio will result in low removal rate of principle components, whereas too low will result in the contrary. The *emg_psd* selects components for rejection based on average power in EEG and EMG bands. It is chosen because it is the only

currently available option [43]. Hamming window is used because it has the sharpest cutoff of the various filter options.

pop_epoch()

Description: this data epochs the data with stored even information

Inputs – EEG: input EEG data structure

- event: selected event for epoching (i.e. 1 or 2 for this study)
- 'newname': give the dataset a new name (i.e. ['newname','S1T1.set'])
- 'epochinfo': save epoch information (i.e. ['epochinfo','yes'])

Outputs – EEG: structure containing newly epoched data

pop_rmbase()

Description: this file removes the baseline from the epoched data

Inputs – EEG: input EEG data structure

- timerange: epoch time range to remove baseline mean from (i.e. [0 4093.75], units ms)

Outputs – EEG: baseline removed EEG structure

pop_runica()

Description: runs ICA decomposition of the data

Inputs – EEG: input EEG data structure

- ' extended' : run ICA on all data

Outputs – EEG: EEG structure with stored ICA weights.

4.2: 2D Plotting ERP and Channel Activity

Electroencephalography (EEG) channels were added to the EEG data structure using the *pop_chanedit*() function in MATLAB. The EEG data files were imported as European Data Format (EDF) .edf files from the PhysioNet Motor Movement Imagery data (eegmmidb) using the *BIOSIG* plugin, aforementioned in subsection 4.1. The channel locations were read using the *chanedit*() with spherical location Standard-10-5-Cap385_witheog.elp. The files were saved to a .ced format to be referenced for all for automation. Fig. 12 demonstrates illustrates the 2 dimensional plot of the channel locations file.

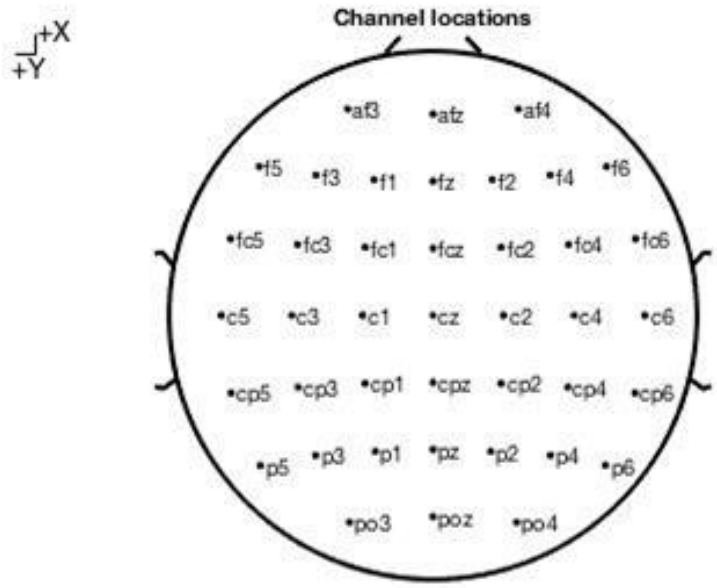


Fig. 12. 2D illustration of the channel locations file

The electrodes are not all located on the scalp, as some of the electrodes wrap around the scalp to the side of the head. For example, electrodes T9 and T10 are both located underneath the ears (acting as reference channels). The 'plotrad' value, set to 0.5,

provided a purely topographical view. Thus, only 41 of the 64 channels are shown in figure 12.

Transient channel data scrolls are illustrated in Fig. 13 with the addition of the .ced channel locations file. The electrodes are labeled on the y-axis. The time (seconds) is labeled on the x-axis. Although Fig. 13 is relatively free of Electromyographic (EMG) and Electrooculargraphic (EOG) interference, this is – more often than not – not the case. The signals often contain noise that can increase the magnitude of the EEG tenfold.

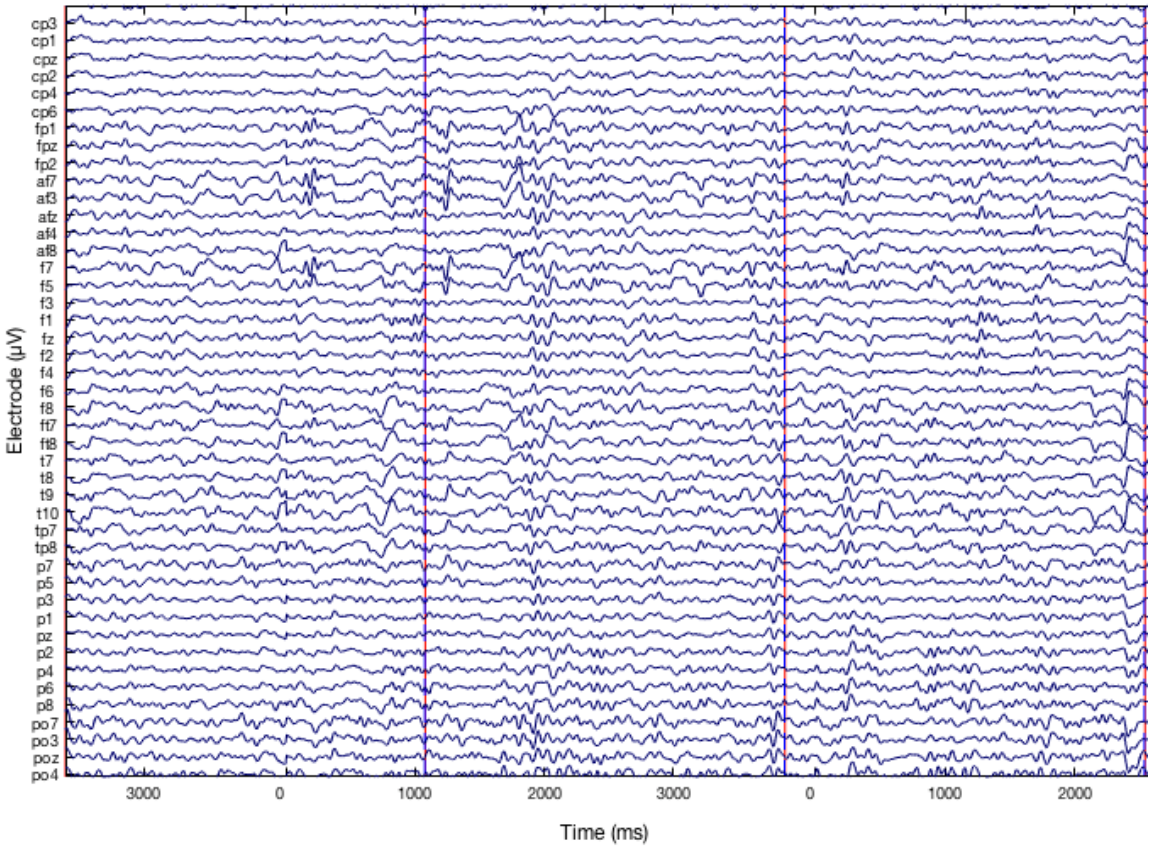


Fig. 13. Labeled 64 channel data scroll

4.3: Feature Extraction Approach

4.3.1: FIR with Hamming Window as Band-pass Filter

Noise exists in the signal that is outside of the range typically used for the analysis of ERD (as well as inside of this region). Furthermore, power-line corruption is present in the activation spectrum of electrode Cz – refer to Fig. 12 – at 60 Hz, Fig. 14. Additionally, electromyographic (EMG) and Electrooculargraphic (EOG) information exists outside of the bandwidth of interest and need to be removed from the signal for discrimination. The Finite Impulse Response (FIR) filter was utilized to remove noise below 2.5 and above 20 Hz. The band-pass filter decentered the signal and reduced the probability of recognition of components in Independent Component Analysis (ICA) outside of the bandwidth of interest.

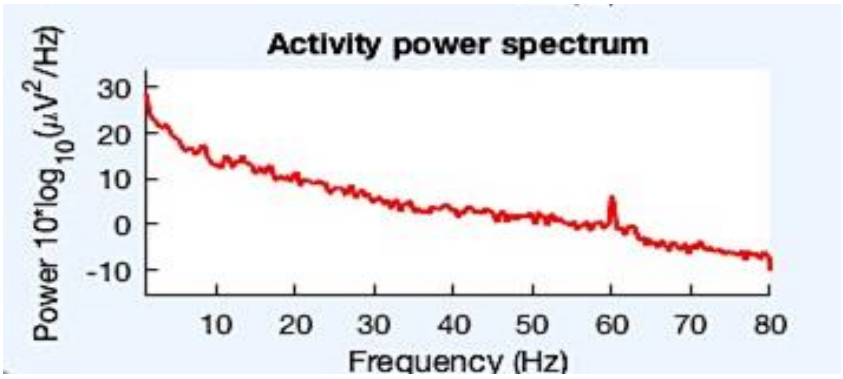
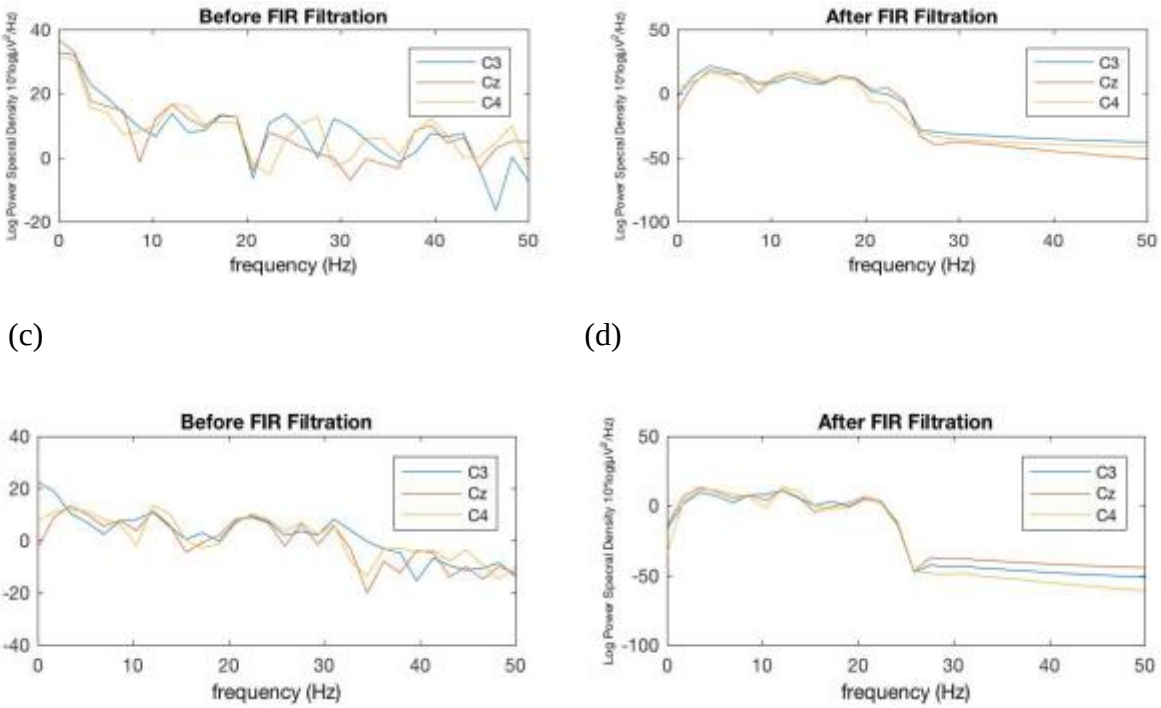


Fig. 14. Activity spectrum demonstrating power-line corruption visible at 60 Hz on electrode Cz removed with other artifact above the 60 Hz.

As a result of filtration, the EMG and EOG artifacts within the spectral range of interest are more easily identified and removed. Fig 15 demonstrates characteristic trends of every signal processed in this study.

(a) (b)



(c) (d)

Fig. 15. Activity spectrum demonstrating Hamming windowed FIR filtered data

There is visible reduction in artifact above the 20 Hz range and minimization of the artifact below 2.5 Hz. There is a sharp cutoff in the power spectrum above 20 Hz, which decreases the influence of these regions.

4.3.2: AAR Using the SOBI and CCA Algorithms

After the prospective neural oscillations are isolated, the Blind Source Separation (BSS) algorithms, which include Second Order Blind Identification (SOBI) and Canonical Correlation Analysis (CCA), remove electromyographic (EMG) and electrooculargraphic (EOG) artifact with the EEGLAB software. First the signals were referenced in order to remove excessive noise. An average reference was arbitrarily selected because there is a lack of research to substantiate that one method is statistically better than another.

Automated Artifact Rejection (AAR) through the use of SOBI and CCA reduce extent of EOG and EMG artifact by flattening disturbances along their ranges of their principal direction. As a result, the event related time course become more deterministic. EOG and EMG artifact have higher amplitude peak. EOG artifact is located more closely to the frontal lobe, whereas EMG can be more scattered throughout. EMG interference demonstrates similar characteristics as the EOG artifact, but is located at various locations along the scalp. First, SOBI automatically removes the EOG artifact via *pop_autobsseog()* function. The inputs used include,

- Eigenvalue ratio of 1.0xE6,
- Bandwidth for fractal dimension of 2 to 21 Hz.

The eigenvalue ratio was set to 1.0E6 by default. This ratio is a good balance for rejection; too high of a ratio results in low rejection ratio whereas too low results in too high of a rejection ratio, refer to subsection 4.1.

Next, automated CCA was used to address EMG corruption via the *pop_autobssemg*() function. The inputs used include,

- Eigenvalue ratio of 1.0xE6
- Welch estimator with a ratio set to 10
- Range is set to 0 to 32 Hz.

Once again the Eigenvalue ratio was set to 1.0E6 by default. The power ratio was set to 10 demonstrated to have good results, refer to subsection 4.1 for more information. Upon completion of these two processes, the EEG signals demonstrate reduced EMG and EOG infiltration.

Disturbances in the waveforms are present, in Fig. 16, in channels FP1 through AF8 (essentially the front of the scalp, above the eyes) between 10 and 11 sec. This is reduced in severity by the AAR processing. The flattening of the peaks demonstrates reduced EOG influence.

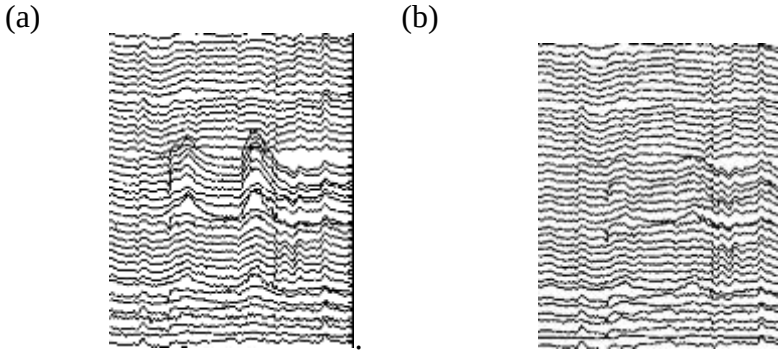


Fig. 16. Demonstration of (a) before and (b) after SOBI and CCA noise reduction of EOG and EMG artifact.

It is necessary to eliminate the noise from all of the channels before removing the principle components because otherwise additional artifact is identified in the principle components that are otherwise naturally removed. Thus, less of the already significant

amount of noise remains in the signal after decomposition with Independent Component Analysis (ICA).

Fig. 17 and 18 illustrate stacked signals before and after Automated Artifact Rejection (AAR). They demonstrate magnitude reduction cause by noise suppression. They depict 4.1s x 15 epochs x 2 (rest for every epoch) x 3 sessions or 369 seconds of right hand imagined motor contraction.

(a) (b)

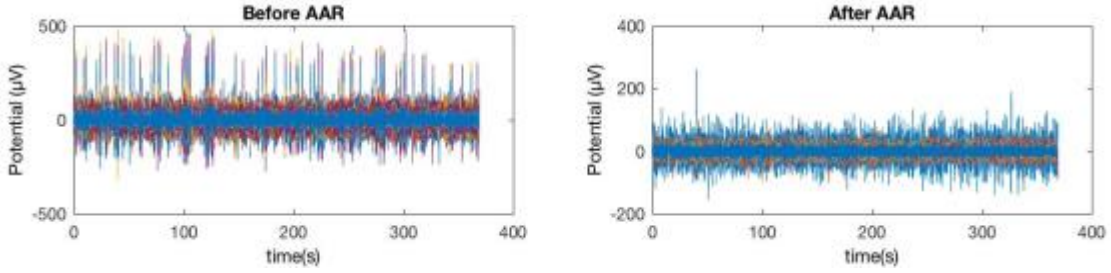


Fig. 17. Demonstration of imagined right hand clenching for 4.1 seconds (a) before and (b) after SOBI and CCA noise reduction of EOG and EMG artifact.

(a) (b)

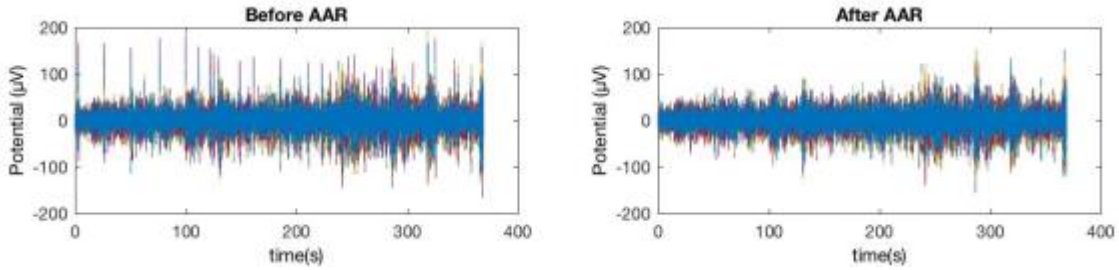


Fig. 18. Demonstration of imagined right hand clenching for 4.1 seconds (a) before and (b) after SOBI and CCA noise reduction of EOG and EMG artifact.

Every signal demonstrates suppression in signal magnitude because the EMG and EOG artifacts perturbations have been reduced. In Fig. 17 the signal was reduced from

approximately 500 μV to 100 μV, which is near the 100's of μV maximum typically demonstrated by neuronal signals [58].

EEG signal amplitudes have an average of 50μV magnitude with a maximum of 100μV. On-the-other-hand, the amplitude of an EOG signal is generally in the range [50, 200] μV and for an EMG signal, it is usually in the range [20, 200] μV, but can reaches as high as 1.5 mV [63]. Fig. 18 demonstrates similar results with reduction from 200 μV to 50 μV. This was consistent with the rest of the data. Fig. 19 demonstrates reduction of 4.1 seconds of right hand clenching for the specific electrodes of interest.

(a) (b)

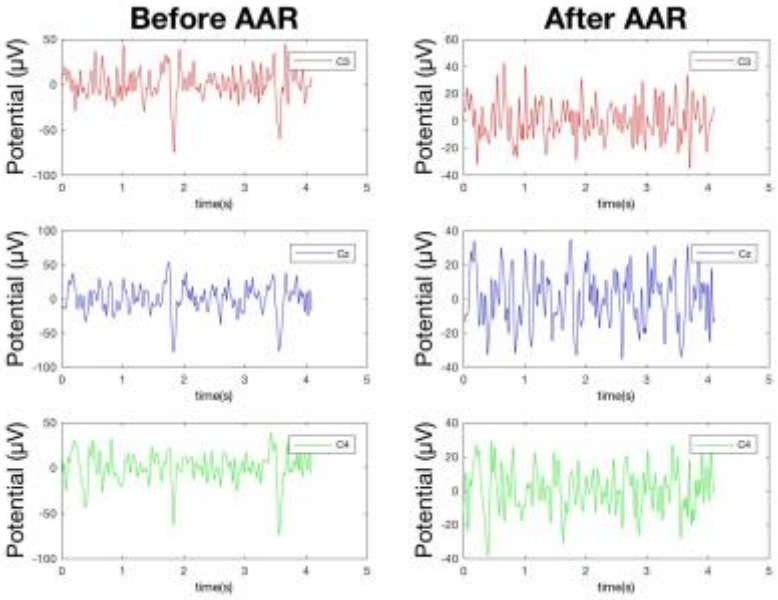


Fig. 19. Demonstration of imagined right hand clenching for 4.1 seconds (a) before and (b) after SOBI and CCA noise reduction of EOG and EMG artifact for the electrodes of interest.

In general there is less noise infiltration of noise into the electrodes of interest. But disturbances are still prominent a visible to the naked eye. For example, the subject

engages in an ocular contraction at approximately 2 seconds and 3.5 seconds of the transient signal. This blink is no longer visible after AAR.

4.3.3: Independent Component Analysis

Independent Component Analysis (ICA) further reduces the pervasiveness of non-neuronal components of the signal. The Infomax version of the ICA algorithm, implemented in MATLAB using EEGLAB's *runica()* function, helps identification of spectral activity using BSS in a similar fashion to Automated Artifact Rejection (AAR) (refer to subsection 4.1 for more details on the function and information on the parameters that are used in this study). Artifacts may be rejected in an automated or semi-automated fashion.

First, the data was first separated, or epoched, into short segments. The relatively short elapsed continuous data and was separated into its 4.1 sec. event time intervals (the same size segments as the original data trials, or event related time course). The data was then concatenated so that all of the event trials (T1 or T2, for left versus right fist) for an individual were merged to form one dataset. Thus, one concatenated segment of data included 45 epochs for left and right hand data.

ICA was used to determine the new components of the data for all of the data for single subject at a time, Fig. 20, using the runica() function in EEGLAB. The process was repeated for all 40 subjects, producing a total of 40 datasets of left and right hand data.

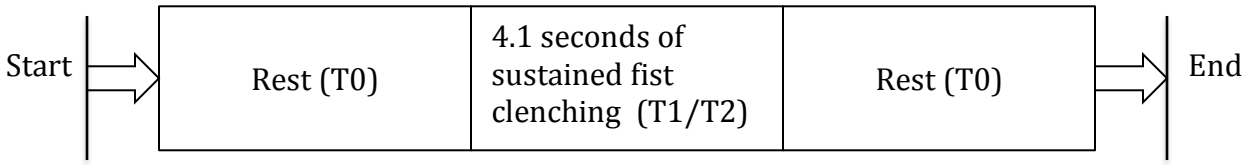


Fig. 20. Schematic of trial epochs extracted from the concatenated data

Decomposition of the data using ICA requires approximately k x n^2 time points in order for the neural network to learn the (n x n) weights. Furthermore, there were 45 trials per person of 4.1-second intervals that were sampled at 160 Hz. Therefore, 29,520 (45x4.1sx160Hz) time points were used. For high-density electrode placement systems, k is set to 25 [70]. The maximum number of channels $n_{\max}$ is calculated by (79),

$$n_{\max} = \sqrt{\frac{t}{25}} \tag{79}$$

Therefore, 24 channels or 3 more electrodes than the 10-20 international system were available for ICA. Thus, the 10-20 system was utilized in the addition of the PC3 and PC4 localizations (above the motor cortex), Fig 21, were used for the identification of independent components.

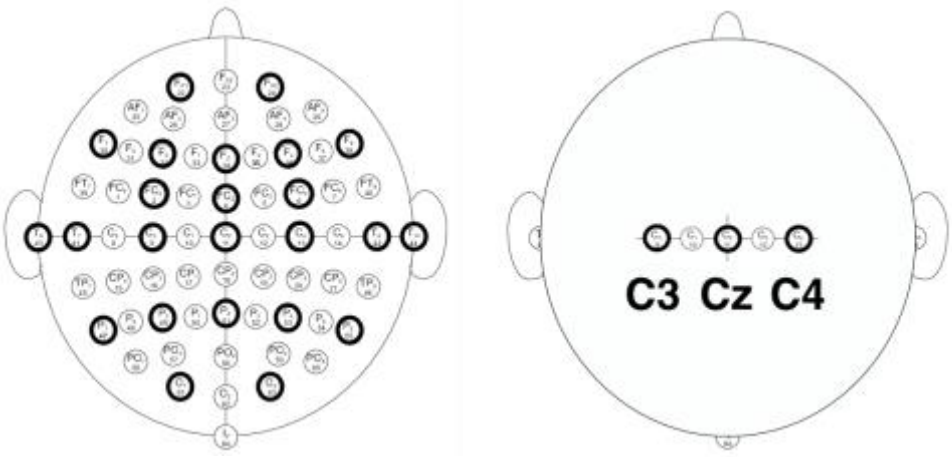


Fig. 21. This illustration depicts the electrodes that were used for (a) ICA as indicated by the dark circles versus (b) DWT-Energy analysis. Adapted from [8]

EEGLAB's *runica*() function, described in subsection 4.1., decomposed the 24 channels of the merged (T1 and T2) datasets. The training error stopping parameter was selected to be 1.0E-6. This value was selected arbitrarily because the author could not find a significant difference between this value and others.

Each simulation capturing two sets of values, ICA weights and spheres (these values calculate the weight matrix and the inverse weight matrix – the pseudo-inverse of the product of the ICA weight matrices and the ICA sphere matrices). The weight matrix is simply the product of these same two matrices and is can be used stored for future use on new time course. The weights and spheres values range from -1 to +1 and are of the size (n x n), where n = 24 is the total number of components.

$$W = EEG.icaweights' \ EEG.icaspheres \tag{80}$$

$$W^{-1} = pinv(EEG.icaweights' \ EEG.icaspheres) \tag{81}$$

The topographical maps are arranged in MATLAB by decreasing order of the variance. There are high levels of EMG and EOG infiltration in the first independent components, which is identified as red along the edge of the scalp map. The component maps and component spectra plotting features of EEGLAB are produced via EEGLABs *pop_specto*() function in Fig. 21. The input is the AAR reduced waveform with all electrodes; a full epoch was used. The three electrodes (C3, Cz, and C4) are marked from left to right in the illustration. 10 Hz (mu/alpha region) was selected as the frequency to be used for analysis.

The red regions in Fig. 22 indicate areas of high power, whereas, the blue regions represent areas of low power. Yellow represents the middle value between these two. Anything else represents a scale of between these three values. Power, in a sense, is

related to energy wherein the signals are continuous over time. Thus, a region of red is indicative of high energy over the entire EEG time course.

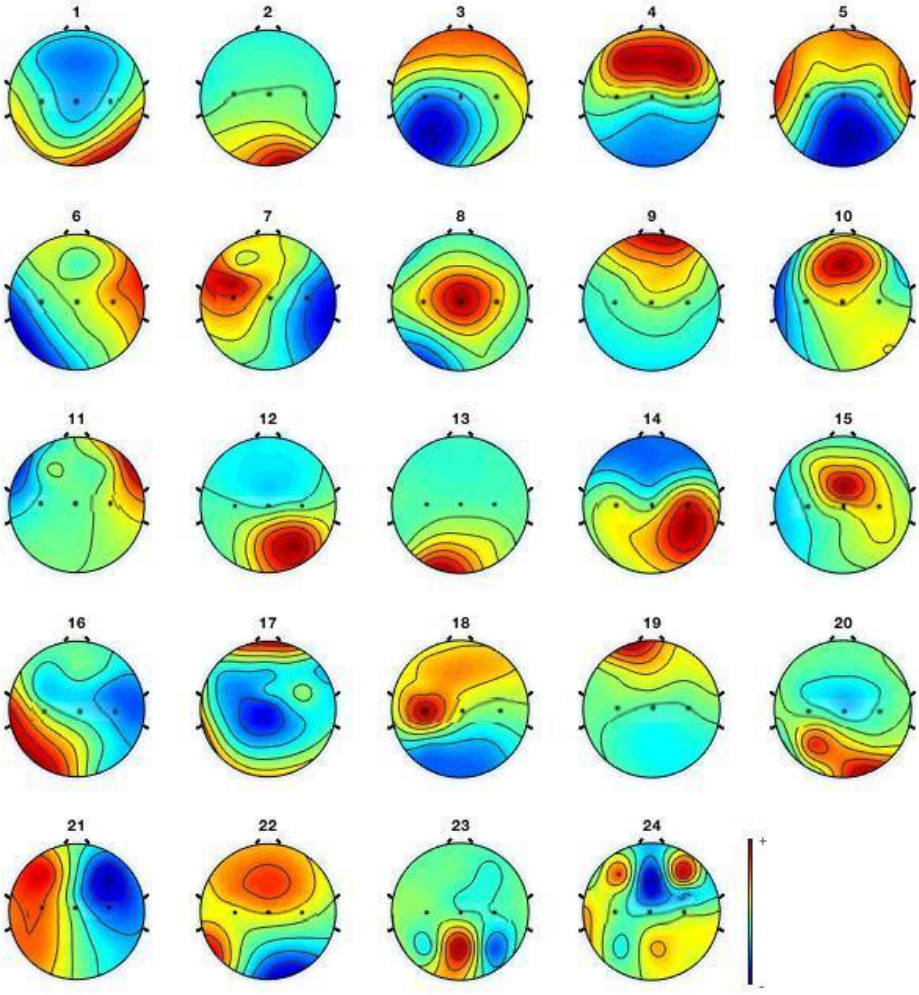


Fig. 22. Topographical mapping of the components for imagination of fist clenching for one of the test subjects.

EOG and EMG artifacts demonstrate a smooth gradual decrease in power with respect to frequency in the activity power spectrum. In map #9, EOG activity is identifiable because the power is being generated from the front of the scalp, whereas in

map #2, EMG activity is observed. Power is entering from the back of the scalp. Fig. 23 illustrates EOG artifact captured from map #9. There is a red region indicating high spectral energy towards the front of the scalp.

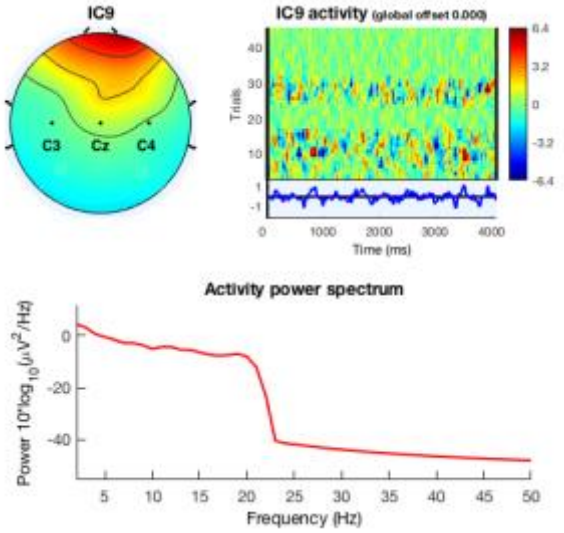


Fig. 23. Demonstration of EOG artifact in the activity spectrum, seen decreasing smoothly in the bottom panel.

There is also a gradual decrease in the energy with respect to frequency with a sharp decrease at approximately 20 Hz. The power is lowest at 22.5 Hz. Anywhere beyond 25 Hz the power remains constantly at -40 ($10*\log(\mu V^2/Hz)$).

In Fig. 24 EMG activity is demonstrated in map 2. It is visible through in the spectrogram where there is a notable red region toward the back of the scalp. Again, this indicates high-energy throughout the time course. There is also a gradual decrease power as the frequency decreases. The EMG components were removed.

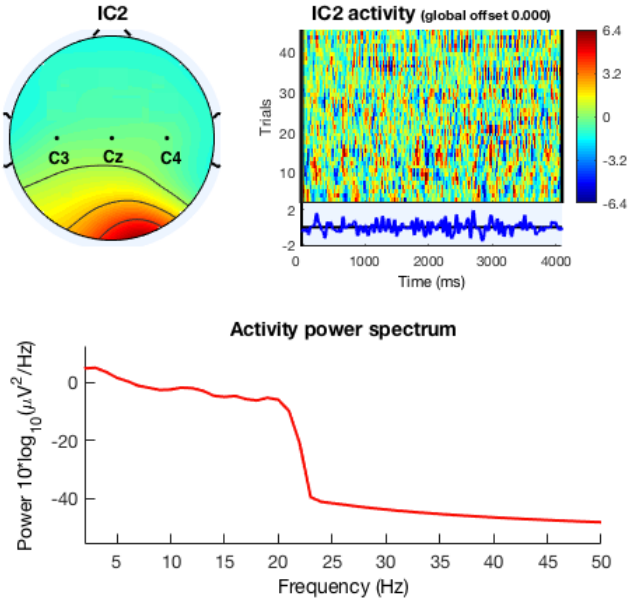


Fig. 24. ICA component for clenching of the right fist for one of the test subjects. This figure depicts the power spectrum typical of EEG contaminated with EMG components.

Fig. 25 demonstrates activity that is characteristic of mu/alpha brain activity. There is a notable increase in power at in approximately 12 Hz. This indicates that Event Related Desynchronization or Synchronization (ERD/ERS) are taking place and is confirmed by looking at the spectrogram in Fig. 26.

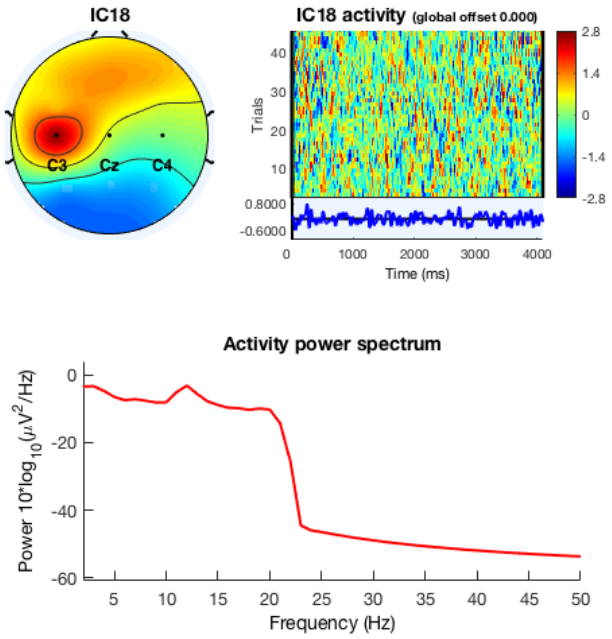


Fig. 25. Individual component activity spectrum for clenching of the left fist for one of the test subjects.

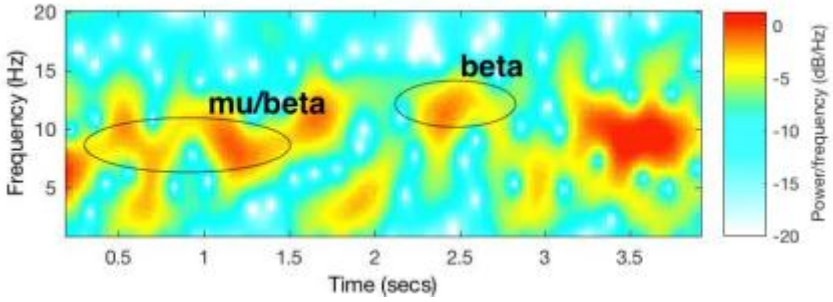


Fig. 26. Spectrogram of the mu/beta activity observed using the Fast Fourier Transform (FFT) with MATLAB's *spectrogram*()

The removal of artifactual components in a semi-automated fashion is illustrated in Fig. 27. Components not demonstrating characteristic activity of the motor cortex were rejected. This includes the aforementioned EMG and EOG artifacts, which are indicated with red highlight above the maps. The components that are retained are indicated with

green highlight above the map. Again, the red in the map indicates high energy over the full time course, whereas blue indicates low energy emission. Data epochs are normally rejected subsequently using windows and spectrum threshold values; however, this step was omitted for the integrity of the study.

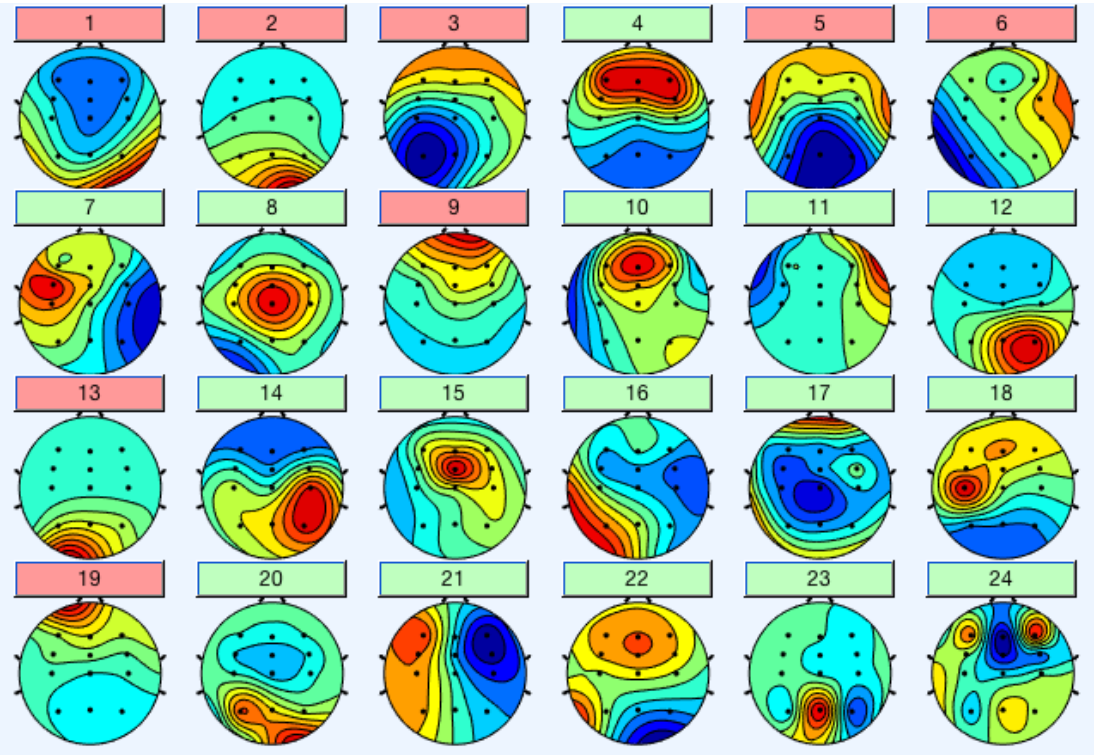


Fig. 27. Topographical map of transient time-course for high-density international 10-10 system. Demonstration of the second pass of ICA using the *runica* in EEGLAB

ICA was then run on the cleaner epochs to better identify the neural components. Component map #5 in Fig. 27 represents activity from the right hand. Map 18 of Fig. 27 portrays activity of the somatosensory cortex. This activity is verified by looking at the component spectra map, Fig. 28, where there is a noticeable spike in the power spectrum

at approximately 12 Hz and again at 27 Hz. This activity is characteristic of Motor Movement Imagery (MMI) synchronization and desynchronization activity.

(a) (b)

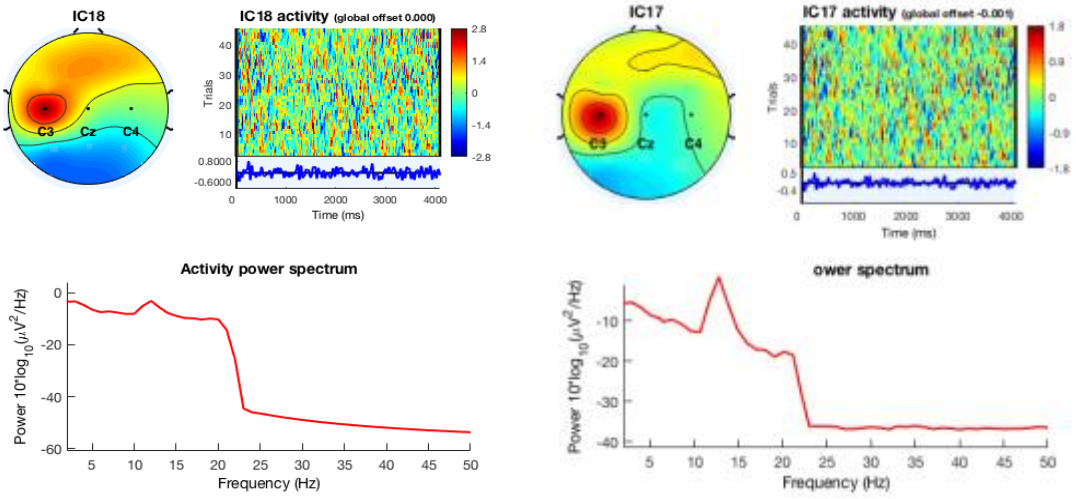


Fig. 28. (a) Before and (b) after iterative ICA decomposition. This figure demonstrates alpha and beta in the activity spectrum seen as rounded peaks in the curve.

The signals should now demonstrate a higher signal to noise ratio as is apparent from the component maps and activity spectrum. Energy spectrum analysis was then performed at the designated bands. Energy was used to analyze the desynchronization activity of the neural oscillations. Desynchronization results in decreased amplitudes in the transient signals as the transient signals becomes out of phase and the signals transition from positive to negative values rapidly. Desynchronization can be observed as an percent energy change.

4.4: Spectral Analysis

Next, spectral analysis was performed on the data using energy percentage of each frequency band. The Discrete Wavelet Transform (DWT), via MATLAB's *wavedec*() function, decomposed the signals into 6 level. A debauchees mother wavelet with 2 vanishing moments was selected (subsection 3.3.1). See below for information about the discrete wavelets transform's *wavedec* function. This produced 6 vectors of detail coefficients and 1 vector of approximation coefficients, Table 6. The approximation was not used. Details 1 through 3 are omitted as well because they were outside the range of interest in this study.

[C,L] = *wavedec*(EEG,level,m_wave)

Description: function for decomposing data with DWT

Inputs – EEG: input EEG data, with electrodes in rows and time points in columns

- 'level': number of leves for decomposition (6)
- 'm_wave': mother wavelet used ('db2')

Outputs – C: decomposition vector

- L: bookkeeping vector

Table 6. Wavelet decomposition for 160 Hz EEG signal

Level/Approx.	**Frequency (Hz)**	**Level**
D6	2.5-5	6 (delta, theta)
D5	5-10	5 (theta, alpha)
D4	10-20	4 (alpha, beta)

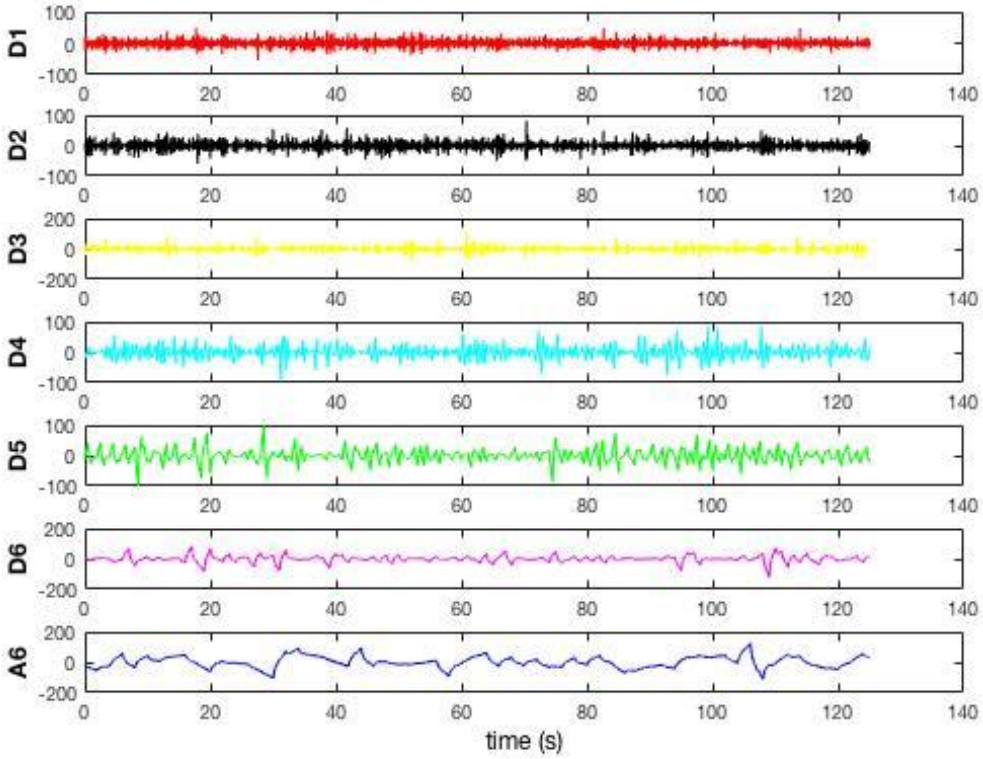


Fig. 29. The illustration depicts levels D4 through D6 of the 6 level decomposition (subject 9) with the DWT function.

The energy percentage at each of the perspective levels was calculated from the detail coefficients via formula (83), which was adapted from Parseval's theorem (82).

$$ED_i = \sum_{j=1}^{N} |D_{ij}|^2 \tag{82}$$

In this equation, the variable *i* corresponds to the *i'th* level *l*, as *j* to the value of the *j*'th index of the level, N_i and *O* to the number of detail and approximation coefficient indices (4.1 seconds x 160 Hz = 656 time points for the reconstructed waveform), and D_{ij} to the detail.

$$EP_i = \frac{ED_i}{E_{total}} \times 100, i = 1,...,l \tag{83}$$

The energy percentage of one frequency band is the proportion of its energy to the total energy of all of the bands. Energy was calculated from the square of the coefficients which are obtained via the *detcoef*() function in MATLAB; with the decomposition vectors, bookkeeping vectors, and decomposition levels in the input.

To capture the grasping motion of the hand, Event Related Potentials (ERPs) recorded from electrodes C3, Cz, and C4 above the motor cortex. These electrodes were required for Event Related (De) synchronization (ERD/ERS) and Movement Related Cortical Potentials (MRCP) analyses, subsection 3.3.3. Correspondingly, activity in the theta and beta rhythmic spectra indicated as D4, D5, and D6 were all extracted features. These features become the input to the Takagi-Sugeno-Kang Style fuzzy inference neural network (TSKFNN), Table 7.

Table 7. TSKFNN inputs with 3 electrode locations, 3 energy percentage bandwidths for 4.1 second event-related time course

Property	Feature	Frequency Bandwidth
Energy Percentage	1	2.5-5 Hz (C3)
(Electrodes C3, C4, Cz)	2	5-10 Hz (C3)
	3	10-20 Hz (C3)
	…	
	9	10-20 Hz (Cz)

The proposed model has 3 electrodes with 3 energy percentages corresponding to a total of (3x3) or nine input features. These are calculated using the coefficients in the 2.5-20 Hz frequency range (2.5 – 5 Hz, 5 – 10 Hz, and 10 – 20 Hz), calculated from electrodes

C3, Cz, and C4, using the DWT. Thus, the input to the neural network is a (9 x 1) vector of the input features above.

4.5: Takagi-Sugeno-Kang Fuzzy Neural Network

MATLAB's Adaptive Neuro-Fuzzy Inference System (ANFIS) was used as he Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) in this study. This neural network translates features into classification decisions for peripheral systems control. The extracted features are inputs to the neural network. A classification decision is made based on weighted firing strengths of neurons in the network. Then performance is increased using supervised learning. The learning algorithm used is illustrated in Fig. 30.

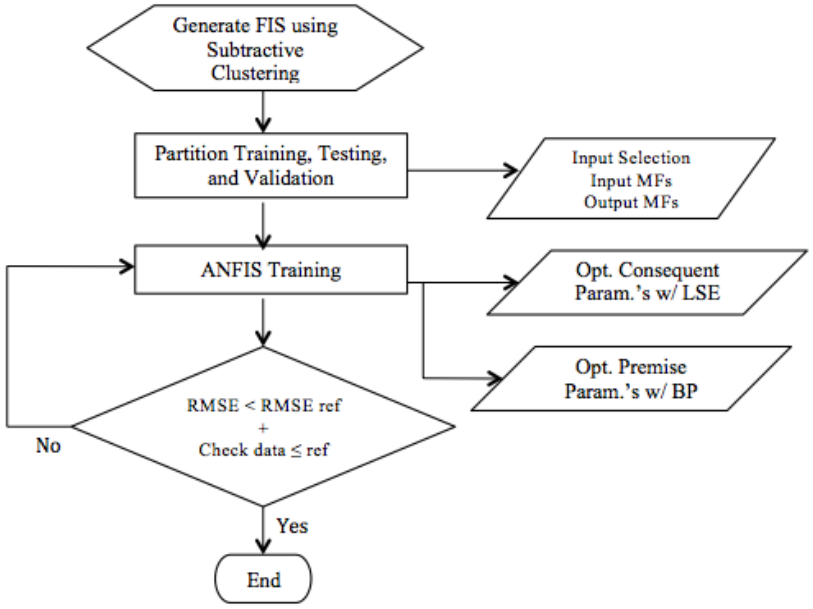


Fig. 30. ANFIS learning method flowchart

To train the network, first 70% (1260) of the total input data was arbitrarily selected using the divide rand function in MATLAB. This consisted of 9 by 1800 feature vectors extracted from the 4.1 s. time segments (40 subjects x 45 movements) for training with all of the subjects at once. For the training sessions of the individual subjects, this corresponded to 70 % of 45, or 31. Targets -1, for the left, and +1, for the right hands, were specified as the output goals.

Next, the Fuzzy Inference System (FIS) was generated via the *genfis*() function. This function generates a system given the clustering method to provide the initial conditions for ANFIS training. The 'SubtractiveClustering' was the input method selected; refer to subsection 3.4.4 for information on subtractive clustering. Additionally, training and target data was entered as input.

```
opt = genfisOptions('SubtractiveClustering');
tks_fis  = genfis(p(1:size(p,1),trainInd)',t(trainInd)',opt);
```

The symmetric Gaussian function was selected due to its normal shape, or nice transition between certainty and uncertainty. Thus, the FIS network system was generated as output for use in the ANFIS.

The output FIS was then trained using the *anfis*() function in MATLAB. The ANFIS also required the same input training data. However, checking data is also required to prevent over training.

```
TRNOPT= [10,NaN,0.0009,0.999,1.001]
[anfis_fis,error,~,~,chkerror] =
anfis(trnData,tks_fis,TRNOPT,DISPOPT,chkData,hybrid);
```

An initial step size of 0.001 was specified in the input and the step sizes increased and decreased by ±10% of the previous value each step, or 0.9 and 1.1 respectively. Lastly, The network was trained using the hybrid LSE-BP learning algorithm, specifying with a Boolean +1 for true and -1 for false, indicating right and left hand, respectively.

The consequent parameters were optimized in the feed forward stage using Least Squares Error (LSE), with fixed premise parameters. Next the output was compared to

the target value and the error signal propagated backwards, updating the premise parameters along the way, with fixed consequent parameters. While the checking data performance decreased and the Root Mean Square Error (RMSE) decreased, then the learning algorithm continued. Otherwise, the training was terminated and the best performing network parameters were retained.

After training the neural network for the single subject event related time course (4.1 seconds of fist 1 or fist 2), generally the network was 10 to 35 fuzzy rules in size. Because the subtractive clustering algorithm groups the data, the quantity of membership functions and parameters was is determined by the data; thus the quantities vary each training session. Example of a single subject training versus all of the subjects trained together, are given Table 8.

Table 8. Demonstration of a Training Session for ANFIS

	Single Subject Trained	All Subjects Trained
Number of Fuzzy Rules	29	11
Number of Linear Parameters	290	110
Number of Non-linear Parameters	522	198
Total Number of Parameters	812	308
Number of Neurons	592	232
Number of Training Data Pairs	31	1259
Number of Checking Data Pairs	7	270

Optimal performance on the testing data is achieved when the 2-norm, or Root Mean Square Error (RMSE), and the validation performance is at its optimal value. This is obtained at approximately 300 training epochs for the single subject trials and 100 for all of the subject trials combined.

The number of fuzzy rules in Table 8 is the number of logical rules developed from the FIS network. Each cluster that has influence has a rule that defines a criterion for selection of the final classification state. An example of the fuzzy rules generated from an ANFIS training session is given Rule (1-*N)*.

Rule 1) IF (in1 is in1cluster1) and (in2 is in2cluster1) and (in3 is in3cluster 1) and …

Rule 2) IF (in1 is in1cluster2) and (in2 is in2cluster2) and (in3 is in3cluster 2) and …

Rule 3) IF (in1 is in1cluster3) and (in2 is in2cluster3) and (in3 is in3cluster 3) and …

…

Rule N) IF (in1 is in1clusterN) and (in2 is in2clusterN) and (in3 is in3cluster N) …

THEN (out1 is cluster [1,2,3,…,*N*])

In this example, if all of the input has eigenvalues that place the point in cluster 1, then the output is cluster 1. If all of the input has eigenvalues that place the point in cluster 2, then the output is cluster 2, and so on. The signal is then defuzzified into a crisp output value that is used to make the classification decision.

As for the other parameters, the number of linear parameters is equivalent to the number of input (9) plus a bias (+1) times the number of fuzzy rules. These are the consequent parameters that are updated without performing a non-linear transformation, see Fig. 9; they are depicted as square nodes. Next, the non-linear terms are those of the

activation functions that are updated in the premise layer. The membership function is near-symmetric Gaussian (slight rounding error due to float); therefore, they include the membership centers and the variance. There are 2 membership variables multiplied by the number of features and the number of fuzzy rules.

The number of neurons in the network was computed as follows. There was one node for each input feature in the input vector (m = 9). Also, there was one bias node, one output computation node, and one output node (+1 +1 +1 = 3). There was one node for each cluster or fuzzy rule and a node for weighting the firing strengths; thus there were 2 nodes for each respective fuzzy rule (+2 x the number of *F.R.*). Lastly, the inputs multiplied by the fuzzy rules gave the number of nodes in the premise layer and the consequent layer (+ 2 x the number of *F.R.* x the number of inputs). Thus for the example in Table 8, there 9 + 3 + (2 x 11) + (2 x 11 x 9) total nodes.

The output of the TSKFNN is classified using a forcing function (binary) to distinguish left from right hand data. The target value for the data is set to +0.9 and -0.9, to ensure that the network remains stable in the case of rounding floating digit error. Output values of less than 0 and a value greater than or equal 0 to +1 correspond to left and right hand classification correspondingly. Zero was selected arbitrarily to represent a value of +1.

$$f(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \tag{84}$$

4.6: Performance Results

The Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) and variations such as Adaptive Neuro-Fuzzy Inference system (ANFIS) have been used on other databases [31][92]. On-the-other hand, the results have not been compared and have also not been compared to the (naturally noisy) PhysioNet Data. Contemporary methods more often utilize the Fast Fourier Transform (FFT) and power for feature extraction along with feed forward neural networks (NN) updated via back propagation (BP) or Support Vector Machine (SVM) for feature translation. Unfortunately, there were not any references that could be found by the author that grouped the data together so that the data results could all be compared. The performance of this algorithm on large datasets cannot be directly compared to other methods at this time for this objective.

However, two cases were used to test the performance of the Adaptive Neuro-Fuzzy Inference Systems (ANFIS) on the PhysioNet data [8]. In the first case, the networks were trained for a single test subject each time comparing their performances. The ANFIS was trained 20 times for each subject and the performance was averaged. In the second case, the networks were trained with data from all of the subjects randomly permutated. This networks was trained 5 times starting with arbitrary weights and the performance was averaged. In both cases, the data was randomly divided into three groups, including training (70 %), checking (15%), and testing (15%). The neural network was never trained with a test data or checking data.

Specificity, sensitivity, selectivity, and accuracy were all calculated to test the performance in the ANFIS, in (84)-(87). In these formulas, true positive (TP) represents class +1 (right hand) data being accurately classified as class +1, whereas true negative

(TN) represents class -1 (left hand) being accurately classified as class -1. On-the-other-hand, false negative (FN) and false positive (FP) correspond to inaccurate classifications of the classes, respectively.

$$Specificity = TN/(TN + FP) \tag{85}$$

$$Sensitivity = TP/(TP + FN) \tag{86}$$

$$Selectivity = TP/(TP + FP) \tag{87}$$

$$Accuracy = (TN + TP)/(TN+TP+FN+FP) \tag{88}$$

The best indicator of overall performance is the accuracy of the system. Higher accuracy means high classification performance on both categories of data, left and right hand. Thus high accuracy means that the medical device will be better able to obtain the proper output control.

4.6.1: Case One (Individual Simulations)

In case one, a single Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) was trained with arbitrary starting weights 20 times. The performances were averaged across these simulations. This was repeated iteratively for each of the first 40 subjects in the naturally noisy PhysioNet Motor Movement Imagery Data [8]. There were 20 x 40 = 800 simulations. The data was divided into 3 sets for each TSKFNN, which included training, checking and testing data. Training sets correspond 70 % (31) of the original 45 datum, whereas checking and testing corresponded to 15 % (7 respectively). Classification was performed on the 9 by 1 vector of features extracted the 4.1-second segments of Event Related Potentials (ERPs), on each of the 3 electrodes (C3, Cz, and C4), refer to subsection 4.4. The training results are given in Table 9.

Table 9. Mean training classification performance by subject for 20 sessions on the naturally noisy 4.1-second EEG left hand and right hand fist contraction data from the PhysioNet MMI database

Subject	Specificity (%)	Sensitivity (%)	Selectivity (%)	Accuracy (%)
1	100	100	100	100
…	…	…	…	…
40	100	100	100	100
Ave	100	100	100	100

The average training performance is very high for all of the subjects. The RMSE approaches 0 for almost all of the subjects as the training performance is 100 %. Higher

performance is expected on the training data; however, this high of a performance is likely due poor performance on the checking data with low number of subjects on high variability of the data. The highest performance was achieved with an initial learning rate of 0.05.

The best performance achieved is at approximately 100 epochs for (a-d) in Fig. 30. It usually occurs at the global maximum of the RMSE curve for the checking error. This is the point at which the network starts overtraining. Table 10 demonstrates the results of using these weights on the testing data.

Table 10. Mean testing classification performance by subject for 20 sessions on the naturally noisy 4.1-second EEG left hand and right hand fist contraction data from the PhysioNet MMI database

Subject	Specificity (%)	Sensitivity (%)	Selectivity (%)	Accuracy (%)
1	52.17	59.83	48.83	53.57
2	52.08	75.50	65.75	64.29
3	73.75	48.42	61.57	66.43
4	80.00	91.25	89.83	87.50
5	91.33	85.67	84.58	88.57
6	89.25	93.50	92.08	91.43
7	78.58	79.58	83.58	79.29
8	92.75	88.25	93.25	90.71
9	63.25	91.67	74.92	78.57
10	76.67	87.50	85.79	85.00

11	87.54	82.11	83.80	85.83
12	69.58	74.33	78.17	71.43
13	58.33	74.92	64.45	65.00
14	68.67	50.33	56.58	58.57
15	90.83	96.67	94.17	94.29
16	84.38	77.08	86.46	83.33
17	71.33	72.00	64.17	70.71
18	85.42	98.17	95.32	95.00
19	95.25	88.58	92.92	90.71
20	84.75	73.42	80.83	80.71
21	84.50	75.50	77.50	77.14
22	74.17	75.33	73.00	73.57
23	79.56	84.08	80.67	79.17
24	94.17	62.17	92.50	77.14
25	67.92	90.25	78.92	82.50
26	77.00	78.16	75.35	77.86
27	71.25	82.67	80.20	77.86
28	87.75	80.25	89.00	84.29
29	84.75	74.33	82.25	77.86
30	72.08	47.33	60.83	55.00
31	84.44	91.37	86.31	87.78
32	82.11	89.47	81.36	85.00
33	89.25	93.42	88.17	90.00

34	76.17	79.58	74.33	79.29
35	70.25	56.25	59.50	64.29
36	80.56	88.89	82.22	85.71
37	86.67	72.69	88.83	79.29
38	70.58	88.25	76.50	79.29
39	95.42	91.42	95.00	91.67
40	86.37	97.89	86.90	92.14
Ave	79.02	79.70	79.66	79.44

Subject 18 achieves the highest performance with 95.0 % accuracy. However, the average accuracy was 79.44 %. These results were consistent with the results in literature recorded in Table 11.

Table 11. Comparison of the DWT-Energy-TSKFNN to Various Other Methods in Literature on the PhysioNet Data

Author	Method	Mean Accuracy (%)	Max. Accuracy (%)
Cheolsoo *et al* [19]	MEMD & SVM	77.7	97.4
Mohamad *et al* [31]	DWT & ANN I	77.6	89.1
Ayman *et al* [32]	DWT & ANN II	64.2	71.6
Mahdiyeh *et al* [40]	DWT & SVM	75.0	NA
Donovan, et al.	***DWT & TSKFNN***	***79.4***	***95.0***

There the authors report accuracies averaging in the high 70's and maximum accuracies in the 70's to high 90's. The mean individual accuracy for the proposed method, ***Donovan, et al***, was slightly higher on average than the best reported findings [18].

4.6.2: Case Two (40 Subject Simulations)

In case two, all 40 subjects data trials are merged grouped together. A single Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) was trained with arbitrary starting weights 5 times. The data, again, was divided into three sets of data (70 % training and 15 % checking and testing each). Results were compared testing the training sessions with and without preprocessing with the Finite Impulse Response Filter (FIR), Artifact Rejection (AAR), and Independent Component Analysis (ICA). There was a total of 1799 datum. Consequentially, each network had 1,349 input and output pairs (45 vectors per person extracted from 4.1 second Event Related Potentials (ERPs) of 3 Electrode Channels as 3 bands of energy – thus, 3 x 3 = 9 input). The test performances were averaged for all 5 simulations.

The Takagi-Sugeno-Kang Fuzzy Neural Network (TSKFNN) produces one of two outputs (+ or – 1) indicating hand (right or left, respectively) clenching. Each training session produces one value. A true positive (top left) corresponds to correct prediction of right hand contraction. A true negative (bottom right) corresponds to correct prediction of left hand contraction. False positive (bottom left) and false negative (bottom right) apply to falsely predicting right hand as left hand contraction and vice versa. The maximum accuracy was the highest accuracy that was achieved for a single neural network out of the 10 that are averaged.

First all 40 of the subjects' data were grouped together for training without Finite Impulse Response (FIR) filtration, AAR, and ICA, Table 12. All of the subject trials grouped together produce n=270 (15 % of 1799) for testing and checking each. Checking validates whether the TSKFNN is overtraining. Checking performance usually decreases

to a maximum neural network performance. After the minimum checking data performance has achieved the network tends to begin learning the data too well and performance decreases. Table 12 demonstrates the performance of the TSKFNN on the testing data. The data is convened using a truth table for the first example with calculations below.

Table 12. Confusion matrix of average for 10 training sessions on the naturally noisy PhysioNet MMI right and left hand clenching data for all test subjects, all forty-five 4.1-second time courses

Trial 1 Testing (n=270)	TSKFNN: Yes (+1)	TSKFNN: No (-1)
Actual: Yes (+1)	70	44
Actual: No (-1)	58	98

The specificity is calculated as:

$$Specificity = \frac{TN}{TN + TP} = \frac{98}{98 + 70} = 58.3\% \tag{89}$$

The sensitivity is calculated as:

$$Sensitivity = \frac{TP}{TP + FN} = \frac{70}{70 + 44} = 61.4\% \tag{90}$$

The selectivity is calculated as:

$$Selectivity = \frac{TP}{TP + FP} = \frac{70}{70 + 58} = 54.7 \tag{91}$$

The accuracy is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{70 + 98}{70 + 98 + 44 + 58} = 62.2\% \tag{92}$$

Thus, the accuracy of the first TKSFNN is 62.2 %. The data for the other 4 trials are given in Table 13.

Table 13. Modified truth table of average for 5 training sessions on the naturally noisy PhysioNet MMI right and left hand clenching data for all test subjects, all forty-five 4.1-second time courses

Subject	Specificity (%)	Sensitivity (%)	Selectivity (%)	Accuracy (%)
1	58.1	61.4	54.7	62.2
2	71.9	68.1	62.9	65.5
3	72.8	69.2	66.4	60.4
4	74.8	60.6	61.2	62.2
5	60.3	66.1	62.6	63.3
Ave	67.6	65.1	61.6	62.7

The resulting neural network produces approximately 62.7 % average classification accuracy; however, the maximum classification accuracy, or maximum performance of any single training session of a new TSKFNN, of all of the subjects is 65.5 %.

Next, all 40 of the subjects' data was grouped together for training with filtration and AAR, but no ICA, Table 12. Again, all of the subject trials are grouped together. This similarly produces 1,349 input for training and 270 for testing and checking each. The training is repeated 5 times using random permutations of the training, checking, and testing data each time. First the all of the data is randomly grouped into training (70%), checking (15%), and testing (15%) by subject. Next the TSKFNN is trained and the performance is recorded. This process is iteratively repeated 5 times. Performance is recorded for all 5 simulation and the results are all averaged, in Table 14.

Table 14. Confusion matrix of average for 10 training sessions on the naturally noisy PhysioNet MMI right and left hand data for all test subjects, all forty-five 4.1-second time courses, with FIR filtration and AAR

Testing (n=270)	TSKFNN: Yes (+1)	TSKFNN: No (-1)
Actual: Yes (+1)	72	41
Actual: No (-1)	47	110

Thus, the first accuracy in Table 15 is 67.4 % per equation (92). The process was repeated iteratively.

Table 15. Modified truth table of average for 5 training sessions on the naturally noisy PhysioNet MMI right and left hand clenching data for all test subjects, all forty-five 4.1-second time courses

Subject	Specificity (%)	Sensitivity (%)	Selectivity (%)	Accuracy (%)
1	60.4	60.5	70.1	67.4
2	74.0	63.07	63.2	69.5
3	65.7	65.4	66.2	65.6
4	64.0	69.7	64.5	67.0
5	65.3	70.3	64.5	68.0
Ave	65.9	65.79	65.7	67.5

The resulting neural network with the additional subject identification feature produces approximately 67.5 % classification accuracy with a maximum of 69.5 %.

Lastly, all 40 of the subjects' data were grouped together for training with filtration, AAR, and ICA. A neural network was trained 10 times using random permutations of the data each time and the performance was averaged, Table 16.

Table 16. Confusion matrix of average for 10 training sessions on the naturally noisy PhysioNet MMI right and left hand clenching data for all test subjects, all forty-five 4.1-second time courses, with advanced signal processing method

Testing (n=270)	TSKFNN: Yes (+1)	TSKFNN: No (-1)
Actual: Yes (+1)	98	37
Actual: No (-1)	42	93

Thus, the first accuracy in Table 15 is 70.4 % per equation (92). The process was repeated iteratively.

Table 17. Modified truth table of average for 5 training sessions on the naturally noisy PhysioNet MMI right and left hand clenching data for all test subjects, all forty-five 4.1-second time courses, with advanced signal processing method

TSKFNN	Specificity (%)	Sensitivity (%)	Selectivity (%)	Accuracy (%)
1	48.6	72.6	70.0	70.4
2	74.0	73.0	71.1	73.0
3	76.5	65.0	75.0	71.5
4	71.8	72.9	72.9	71.8
5	62.8	69.2	66.4	71.4
Ave	66.7	70.5	71.0	71.6

The resulting neural network produces approximately 71.6% classification accuracy. However, the maximum classification accuracy, or maximum performance of any single training session of a new TSKFNN, of all of the subjects is 73.0 %.

Overall, the classification accuracy was only slightly higher than the highest performing algorithm found in literature, Noise Assisted Multivariate Empirical Mode Decomposition (NA-MEMD) with Support Vector Machine (SVM). The number of electrodes used in this method, however, was much less. This model had 3 electrodes whereas the other methods used up to 11 electrodes. Thus this method is more computationally efficient because it requires fewer computations to learn the weights than the former methods.

CHAPTER 5: CONCLUSION AND FUTURE RESEARCH

More recently, the DWT is making its way into the biological research community because of its superior basis functions. Its basis functions are better able to handle non-symmetric sharp EEG signals. Furthermore, it retains the ability to perform to be used in the analysis of band specific information such as energy spectrum. Fuzzy Logic has also been making its way into the world of research. Synergizing with machine learning, the Takagi-Sugeno-Kang Fuzzy Neural Network, or ANFIS, is capable of learning precise outputs functions despite uncertainty in the formulation of the input variable. Together, these tools provide classification performance that is higher than the current methods proposed in the literature.

The proposed algorithm achieves higher classification accuracy for the single subject simulations, relative to the standard reported literature value. However, the average for all of the test subjects remains less than ideal. An average accuracy of 79.4% detection is less then formidable, but still high for this data. Therefore, although this method poses as superior, there remains a need to rectify this model for this type of data before it can be used as an output medical device. Improvements that need to be made include increased accuracy of detection on the single subject trials as well as adaptability for the multiple subject trials.

It is recommended that future classification schemes build on the feature extraction scheme. Although the DWT is a superior decomposition method, the proposed employment of this method negates itself a high-resolution time-frequency analysis for determining ERD. The frequency bands are too broad, and consequently contain excessive noise that is not related to slow moving cortical potentials. Furthermore, the

energy percentage is captured for different frequency bands. However, with great amount of noise and perturbations, the influence of these signals becomes occluded. Therefore, in order to capture the ERD, the source needs to be better isolated, which is not possible with wide bands and high levels of noise.

In addition, further research into noise reduction should be conducted. Until noise can be severely reduced, results will continue to fall short of expectations for an effective medical device. The data set that is used in this study is polluted with noise to the extent that processing with AAR and ICA still did not reduce the noise to an extent that it was easy to observe the activity in the desired region. Use of intracranial electrodes would drastically increase the SNR value; however pose as biocompatibility concerns with its high incidence of infections.

BIBLIOGRAPHY

[1] McGimpsey, G., and T. C. Bradford. *1 Limb Prosthetics Services and Devices Critical Unmet Need: Market Analysis*. N.p.: n.p., n.d. PDF.

[2] Sethuraman, Swaminathan, Uma Maheswari Krishnan, and Anuradha Subramanian. *Biomaterials and nanotechnology for tissue engineering*. Boca Raton: CRC Press/Taylor & Francis, 2017. Print.

[3] Cososchi, Stefan, Rodica Strungaru, Alexandru Ungureanu, and Mihaela Ungureanu. "EEG Features Extraction for Motor Imagery." *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006. doi:10.1109/iembs.2006.4397608.

[4] Wurth, S., M. Capogrosso, S. Raspopovic, J. Gandar, G. Federici, N. Kinany, A. Cutrone, A. Piersigilli, N. Pavlova, R. Guiet, G. Taverni, J. Rigosa, P. Shkorbatova, X. Navarro, Q. Barraud, G. Courtine, and S. Micera. "Long-term usability and bio-integration of polyimide-based intra-neural stimulating electrodes." *Biomaterials* 122 (2017): 114-29. doi:10.1016/j.biomaterials.2017.01.014.

[5] Kha, Ha Hoang. "Real-time brainwave-controlled interface using P300 component in EEG signal processing." *2016 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, 2016. doi:10.1109/rivf.2016.7800300.

[6] Mestais, Corinne S., Guillaume Charvet, Fabien Sauter-Starace, Michael Foerster, David Ratel, and Alim Louis Benabid. "WIMAGINE: Wireless 64-Channel ECoG Recording Implant for Long Term Clinical Applications." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23, no. 1 (2015): 10-21. doi:10.1109/tnsre.2014.2333541.

[7] Stahl, Bernd Carsten, Kutoma Wakunuma, Stephen Rainey, and Christian Hansen. "Improving brain computer interface research through user involvement - The transformative potential of integrating civil society organisations in research projects." *Plos One* 12, no. 2 (2017). doi:10.1371/journal.pone.0171818.

[8] Schalk, G., D.j. Mcfarland, T. Hinterberger, N. Birbaumer, and J.r. Wolpaw. "BCI2000: A General-Purpose Brain-Computer Interface (BCI) System." *IEEE Transactions on Biomedical Engineering IEEE Trans. Biomed. Eng.* 51.6 (2004): 1034-043. Web.

[9] Fouad, Mohamed Mostafa, Khalid Mohamed Amin, Nashwa El-Bendary, and Aboul Ella Hassanien. "Brain Computer Interface: A Review." *Brain-Computer Interfaces Intelligent Systems Reference Library* (2012): 3-30. Web.

[10] "*Use of International Standard ISO-10993, 'Biological Evaluation of Medical Devices Part 1: Evaluation and Testing' (blue book memo)*." PDF. FDA.

[11] Burle, Borís, Laure Spieser, Clémence Roger, Laurence Casini, Thierry Hasbroucq, and Franck Vidal. "Spatial and temporal resolutions of EEG: Is it really black and white? A scalp current density view." *International Journal of Psychophysiology* 97, no. 3 (2015): 210-20. doi:10.1016/j.ijpsycho.2015.05.004.

[12] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* **101**(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215]; 2000 (June 13).

[13] V. Bajaj and R. B. Pachori, "Classification of Seizure and Nonseizure EEG Signals Using Empirical Mode Decomposition," in *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 6, pp. 1135-1142, Nov. 2012. doi: 10.1109/TITB.2011.2181403

[14] Alessio, Silvia Maria. *Digital signal processing and spectral analysis for scientists: concepts and applications*. Cham: Springer, 2016.

[15] Narayasamy, K. *Precision engineering*. New Delhi: Narosa Pub. House, 2000.

[16] Lin, Faa-Jeng, Ying-Chih Hung, Jonq-Chin Hwang, and Meng-Ting Tsai. "Fault-Tolerant Control of a Six-Phase Motor Drive System Using a Takagi–Sugeno–Kang Type Fuzzy Neural Network With Asymmetric Membership Function." *IEEE Transactions on Power Electronics IEEE Trans. Power Electron.* 28.7 (2013): 3557-572. Web. 29 July 2016.

[17] Jang, J.-S.r. "ANFIS: Adaptive-network-based Fuzzy Inference System." *IEEE Transactions on Systems, Man, and Cybernetics IEEE Trans. Syst., Man, Cybern.* 23.3 (1993): 665-85. Web. 9 Aug. 2016.

[18] Park, Cheolsoo, David Looney, Naveed Ur Rehman, Alireza Ahrabian, and Danilo P. Mandic. "Classification of Motor Imagery BCI Using Multivariate Empirical Mode Decomposition." *IEEE Trans. Neural Syst. Rehabil. Eng. IEEE Transactions on Neural Systems and Rehabilitation Engineering* 21.1 (2013): 10-22. Web. 26 Aug. 2016.

[19] Zhang, Jin, Guoli Li, Chao Xu, Deyang Shao, and Long Jiao. "Study of Harmonic Analysis Based on Improved Discrete Fourier Transform." *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)* (2016): n. pag. Web. 1 Nov. 2016.

[20] H. Yuan, A. Doud, A. Gururajan, and B. He, "Cortical imaging of event-related (de)synchronization during online control of brain-computer interface using

minimum-norm estimates in frequency domain," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 16, no. 5, pp. 425–431, Oct. 2008.

[21] Xiaojing, Guo, Wu Xiaopei, and Zhang Dexiang. "Motor imagery EEG detection by empirical mode decomposition." *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008. doi:10.1109/ijcnn.2008.4634164.

[22] Raouf, Heba, Heba Yousef, and Atef Ghonem. "An Analytical Comparison between Applying FFT and DWT in W IMAX Systems." *IJCSI International Journal of Computer Science Issues* 2nd ser. 11.6 (2014): 18-26. Web. 1 Nov. 2016.

[23] Wasilewski, Filip. "Wavelet Browser by PyWavelets." Daubechies 2 wavelet (db2) properties, filters and functions - Wavelet Properties Browser. Accessed April 11, 2017. http://wavelets.pybytes.com/wavelet/db2/.

[24] Cho, Dongrae, Beomjun Min, Jongin Kim, and Boreom Lee. "EEG-based Prediction of Epileptic Seizures Using Phase Synchronization Elicited from Noise-Assisted Multivariate Empirical Mode Decomposition." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2016, 1. doi:10.1109/tnsre.2016.2618937.

[25] Ni, Shun-Hao, Wei-Chau Xie, and Mahesh Pandey. "Application of Hilbert-Huang Transform in Generating Spectrum-Compatible Earthquake Time Histories." *ISRN Signal Processing* 2011 (2011): 1-17. doi:10.5402/2011/563678.

[26] Looney, D., A. Hemakom, and D. P. Mandic. "Intrinsic multi-scale analysis: a multi-variate empirical mode decomposition framework." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471, no. 2173 (2014): 20140709. doi:10.1098/rspa.2014.0709.

[27] Wang, Yijun, Shangkai Gao, and Xiaornog Gao. "Common Spatial Pattern Method for Channel Selelction in Motor Imagery Based Brain-computer Interface." *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005. doi:10.1109/iembs.2005.1615701.

[28] Patel, Rajesh, Madhukar Pandurangrao Janawadkar, Senthilnathan Sengottuvel, Katholil Gireesan, and Thimmakudy Sambasiva Radhakrishnan. "Suppression of Eye-Blink Associated Artifact Using Single Channel EEG Data by Combining Cross-Correlation With Empirical Mode Decomposition." *IEEE Sensors Journal* 16, no. 18 (2016): 6947-954. doi:10.1109/jsen.2016.2591580.

[29] Alomari, Mohammad H., Emad A. Awada, Aya Samaha, and Khaled Alkamha. "Wavelet-Based Feature Extraction for the Analysis of EEG Signals Associated with Imagined Fists and Feet Movements." *CIS Computer and Information Science* 7.2 (2014): n. pag. Web. 25 July 2016.

[30] H., Mohammad, Ayman Abubaker, Aiman Turani, and Ali M. "EEG Mouse:A Machine Learning-Based Brain Computer Interface." *International Journal of Advanced Computer Science and Applications IJACSA* 5.4 (2014): n. pag. Web. 26 Aug. 2016.

[31] Amin, Hafeez Ullah, Aamir Saeed Malik, Rana Fayyaz Ahmad, Nasreen Badruddin, Nidal Kamel, Muhammad Hussain, and Weng-Tink Chooi. "Feature extraction and classification for EEG signals using wavelet transform and machine learning techniques." *Australasian Physical & Engineering Sciences in Medicine* 38, no. 1 (2015): 139-49. doi:10.1007/s13246-015-0333-x.

[32] Haykin, Simon S. *Neural networks and learning machines*. New Dehli: PHI Learning, 2011.

[33] Hajibabazadeh, Mahdiyeh, and Vahid Azimirad. "Brain-robot Interface: Distinguishing Left and Right Hand EEG Signals through SVM." *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)* (2014): n. pag. Web. 26 Aug. 2016.

[34] Solla, Sara A., Klaus-Robert Müller, and Todd K. Leen. *Advances in neural information processing systems 12: proceedings of the 1999 conference*. Cambridge, MA: MIT Press, 2000.

[35] Support Vector Regression. Accessed April 12, 2017. http://www.saedsayad.com/support_vector_machine_reg.htm.

[36] Xiong, Sheng-Wu, Hong-Bing Liu, and Xiao-Xiao Niu. "Fuzzy support vector machines based on FCM clustering." *2005 International Conference on Machine Learning and Cybernetics*, 2005. doi:10.1109/icmlc.2005.1527384.

[37] Tsang, E.c.c., D.s. Yeung, and P.p.k. Chan. "Fuzzy support vector machines for solving two-class problems." *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*. doi:10.1109/icmlc.2003.1259643.

[38] Xu, Qi, Hui Zhou, Yongji Wang, and Jian Huang. "Fuzzy support vector machine for classification of EEG signals using wavelet-based features." *Medical Engineering & Physics* 31, no. 7 (2009): 858-65. doi:10.1016/j.medengphy.2009.04.005.

[39] Nguyen, Thanh, Saeid Nahavandi, Abbas Khosravi, Douglas Creighton, and Imali Hettiarachchi. "EEG signal analysis for BCI application using fuzzy system." *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015. doi:10.1109/ijcnn.2015.7280593.

[40] Hajibabazadeh, Mahdiyeh, and Vahid Azimirad. "Brain-robot Interface: Distinguishing Left and Right Hand EEG Signals through SVM." *2014 Second*

RSI/ISM International Conference on Robotics and Mechatronics (ICRoM) (2014): n. pag. Web. 26 Aug. 2016.

[41] Jahankani, P., Kodogiannis, V., Lygouras, J.: Adaptive fuzzy inference neural network system for EEG signal classification. In: Jain, L.C., Lim, C. P. (eds.) Handbook on Decision Making. ISRL vol. 4, pp. 453–471. Springer, Heidelberg (2010): 10 March 2016.

[42] Perner, Petra. *Machine Learning and Data Mining in Pattern Recognition 8th International Conference, MLDM 2015, Hamburg, Germany, 2012, Proceedings*. Cham: Springer International Publishing, 2012.

[43] Gomez-Herrero, German, Wim Clercq, Haroon Anwar, Olga Kara, Karen Egiazarian, Sabine Huffel, and Wim Paesschen. "Automatic Removal of Ocular Artifacts in the EEG without an EOG Reference Channel." *Proceedings of the 7th Nordic Signal Processing Symposium - NORSIG 2006* (2006): n. pag. Web. 27 July 2016.

[44] Clercq, Wim De, A. Vergult, B. Vanrumste, W. Van Paesschen, and S. Van Huffel. "Canonical Correlation Analysis Applied to Remove Muscle Artifacts From the Electroencephalogram." *IEEE Transactions on Biomedical Engineering IEEE Trans. Biomed. Eng.* 53.12 (2006): 2583-587. Web. 27 July 2016.

[45] A. Andrei, *et al.*, "Chronic behavior evaluation of a micro-machined neural implant with optimized design based on an experimentally derived model," *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, MA, 2011, pp. 2292-2295. doi: 10.1109/IEMBS.2011.6090577

[46] "Brain Waves." Welcome to EcoDeCrypt. Accessed April 13, 2017. http://www.ecodecrypt.com/brain-waves.html. [48] B. He, S. Gao, H. Yuan, and J. R. Wolpaw,‘‘Brain-computer interface,’’ in Neural Eng.,B. He, Ed., 2nd ed.Boston, MA: Springer,2013, pp. 87–151.

[47] Grinsell, D., and C. P. Keating. "Peripheral Nerve Reconstruction after Injury: A Review of Clinical and Experimental Therapies." *BioMed Research International* 2014 (2014): 1-13. doi:10.1155/2014/698256.

[48] Turnip, Arjon, and Edy Junaidi. "Removal Artifacts from EEG Signal Using Independent Component Analysis and Principal Component Analysis." *2014 2nd International Conference on Technology, Informatics, Management, Engineering & Environment* (2015): n. pag. Web. 31 Oct. 2016.

[49] Whitmarsh, Stephen, Ingrid L. C. Nieuwenhuis, Henk P. Barendregt, and Ole Jensen. "Sensorimotor Alpha Activity is Modulated in Response to the Observation of Pain in Others." *Frontiers in Human Neuroscience* 5 (2011). doi:10.3389/fnhum.2011.00091.

[50] Gevensleben, Holger, Albrecht, Henry, Tibor Auer, Wan Ilma Dewiputri, Renate Schweizer, Gunther Moll, Hartmut Heinrich, and Aribert Rothenberger. "Neurofeedback of slow cortical potentials: neural mechanisms and feasibility of a placebo-controlled design in healthy adults." *Frontiers in Human Neuroscience* 8 (2014). doi:10.3389/fnhum.2014.00990.

[51] B. Rockstroh, M. Muller , M. Wagner, R. Cohen, T. Elbert "Probing the nature of the CNV*." Electroencephalogr. Clin. Neurophysiol*. (1993) 87, pp. 235–241. doi: 10.1016/0013-4694(93)90023-o

[52] A. A. Ioannides, P. B. Fenwick, J. Lumsden J., *et al.* "Activation sequence of discrete brain areas during cognitive processes: results from magnetic field tomography*." Electroencephalogr. Clin. Neurophysiol.* (1994) vol.91, pp. 399–402. doi: 10.1016/0013-4694(94)90125-2

[53] D. Durstewitz. "Neural representation of interval time." *Neuroreport* (2004) pp.15, 745–749. doi: 10.1097/00001756-200404090-00001

[54] Y. Nagai, H. D. Critchley, E. Featherstone,*et al*. "Brain activity relating to the contingent negative variation: an fMRI investigation." *Neuroimage* (2004) pp. 21, 1232–1241. doi: 10.1016/j.neuroimage.2003.10.036

[55] W. G. Walter, R. Cooper , V. J. Aldridge, et al "Contingent negative variation: an electric sign of sensorimotor association and expectancy in the human brain." Nature 203, 380–384. doi:10.1038/203380a0

[56] T. Banaschewski, D. Brandeis, H. Heinrich, *et al*. "Association of ADHD and conduct disorder–brain electrical evidence for the existence of a distinct subtype." *J. Child Psychol. Psychiatry* (2003) pp. 44, 356–376. doi: 10.1111/1469-7610.00127

[57] "Software." Still breathing. Accessed April 13, 2017. http://still-breathing.net/software/.

[58] Northrop, Robert B. *Analysis and Application of Analog Electronic Circuits to Biomedical Instrumentation*. Boca Raton: CRC, 2012. Print.

[59] Kappenman, Emily S., and Steven J. Luck. "The effects of electrode impedance on data quality and statistical significance in ERP recordings." *Psychophysiology*, 2010. doi:10.1111/j.1469-8986.2010.01009.x.

[60] Gomez-Herrero, German, Wim Clercq, Haroon Anwar, Olga Kara, Karen Egiazarian, Sabine Huffel, and Wim Paesschen. "Automatic Removal of Ocular Artifacts in the EEG without an EOG Reference Channel." *Proceedings of the 7th Nordic Signal Processing Symposium - NORSIG 2006* (2006): n. pag. Web. 27 July 2016.

[61] Hu, Jing, Chun-Sheng Wang, Min Wu, Yu-Xiao Du, Yong He, and Jinhua She. "Removal of EOG and EMG artifacts from EEG using combination of functional link neural network and adaptive neural fuzzy inference system." *Neurocomputing* 151 (2015): 278-87. Web.

[62] "5. Aliasing Artifacts and Noise in CT Images." *Principles of Computerized Tomographic Imaging*, 2001, 177-201. doi:10.1137/1.9780898719277.ch5.

[63] Mihov, Georgy. "Complex filters for the subtraction procedure for power-line interference removal from ECG." *International Journal of Reasoning-based Intelligent Systems* 5, no. 3 (2013): 146. doi:10.1504/ijris.2013.058184.

[64] Niemarkt HJ, Jennekens W, Pasman JW, Katgert T, Van Pul C, Gavilanes AW, et al. Maturational changes in automated EEG spectral power analysis in preterm infants. Pediatr Res. 2011;70(5):529–534. doi: 10.1203/PDR.0b013e31822d748b.

[65] Duan, Suolin, Tingting Xu, Wei Zhuang, and Dan Mao. "The Feature Extraction of ERD/ERS Signals Based on the Wavelet Package and ICA." *Proceeding of the 11th World Congress on Intelligent Control and Automation* (2014): n. pag. Web. 25 July 2016.

[66] Scolaro, G. R., F. M. Azevedo, and C. F. Boos. "Evaluation of Different Wavelet Functions Applied in the Development of Digital Filters to Attenuate the Background Activity in EEG Signals." *IFMBE Proceedings World Congress on Medical Physics and Biomedical Engineering May 26-31, 2012, Beijing, China* (2013): 340-43. Web. 21 Aug. 2016.

[67] Long, Mian. *World Congress on Medical Physics and Biomedical Engineering: May 26-31, 2012, Beijing, China*. Heidelberg: Springer, 2013.

[68] Addison, Paul S. "Wavelet transforms and the ECG: a review." *Physiological Measurement* 26, no. 5 (2005). doi:10.1088/0967-3334/26/5/r01.

[69] Al-Fahoum, Amjed S., and Ausilah A. Al-Fraihat. "Methods of EEG Signal Features Extraction Using Linear Analysis in Frequency and Time-Frequency Domains." *ISRN Neuroscience* 2014 (2014): 1-7. doi:10.1155/2014/730218.

[70] Onton, Julie, and Scott Makeig. "Information-based modeling of event-related brain dynamics." *Progress in Brain Research Event-Related Dynamics of Brain Oscillations*, 2006, 99-120. doi:10.1016/s0079-6123(06)59007-7.

[71] Shayegh, Farzaneh, and Abbas Erfanian. "Real-Time Ocular Artifacts Suppression from EEG Signals Using an Unsupervised Adaptive Blind Source Separation." *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006. doi:10.1109/iembs.2006.259611.

[72] Hamaneh, Mehdi Bagheri, Numthip Chitravas, Kitti Kaiboriboon, Samden D. Lhatoo, and Kenneth A. Loparo. "Automated Removal of EKG Artifact From

EEG Data Using Independent Component Analysis and Continuous Wavelet Transformation." *IEEE Transactions on Biomedical Engineering* 61, no. 6 (2014): 1634-641. doi:10.1109/tbme.2013.2295173.

[73] Klados, Manousos A., C. Papadelis, C. D. Lithari, and P. D. Bamidis. "The Removal Of Ocular Artifacts From EEG Signals: A Comparison of Performances For Different Methods." *IFMBE Proceedings 4th European Conference of the International Federation for Medical and Biological Engineering*, 2009, 1259-263. doi:10.1007/978-3-540-89208-3_300.

[74] Liu, Wei, Danilo P. Mandic, and Andrzej Cichocki. "Blind Source Separation Based on Generalised Canonical Correlation Analysis and Its Adaptive Realization." *2008 Congress on Image and Signal Processing*, 2008. doi:10.1109/cisp.2008.515.

[75] Allison, Brendan. *Towards Practical Brain-computer Interfaces: Bridging the Gap from Research to Real-world Applications*. Heidelberg: Springer, 2012. N. pag. Print.

[76] B. He, S. Gao, H. Yuan, and J. R. Wolpaw,‘‘Brain-computer interface,’’ in Neural Eng.,B. He, Ed., 2nd ed.Boston, MA: Springer,2013, pp. 87–151.

[77] Pfurtscheller, G. "Graphical Display and Statistical Evaluation of Event Related Desynchronization (ERD)." *Clin. Neur. Physiol.* 43.5 (1977): 757-60. Web. 3 Nov. 2016

[78] Jeon, Yongwoong, Chang S. Nam, Young-Joo Kim, and Min Cheol Whang. "Event-related (De)synchronization (ERD/ERS) during Motor Imagery Tasks: Implications for Brain–computer Interfaces." *International Journal of Industrial Ergonomics* 41.5 (2011): 428-36. Web. 25 July 2016.

[79] Chen, Xiaogang, Guangyu Bin, Ian Daly, and Xiaorong Gao. "Event-related desynchronization (ERD) in the alpha band during a hand mental rotation task." *Neuroscience Letters* 541 (2013): 238-42. doi:10.1016/j.neulet.2013.02.036.

[80] Wright, David J., Paul S. Holmes, and Dave Smith. "Using the Movement-Related Cortical Potential to Study Motor Skill Learning." *Journal of Motor Behavior* 43.3 (2011): 193-201. Web. 1 Sept. 2016.

[81] "What Is Fuzzy Logic?" *MathWorks*. N.p., n.d. Web. 29 July 2016.

[82] S. J. Yoo, Y. H. Choi, and J. B. Park, “Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: Adaptive learning rates approach,” *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 53, no. 6, pp. 1381–1395, Jun. 2006.

[83] Takagi, Tomohiro, and Michio Sugeno. "Fuzzy Identification of Systems and Its Applications to Modeling and Control." *Readings in Fuzzy Sets for Intelligent Systems*, 1993, 387-403. doi:10.1016/b978-1-4832-1450-4.50045-6.

[84] Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern classification*.

[85] "Curse of Dimensionality." *SpringerReference*, 2010. doi:10.1007/springerreference_178904.

[86] Grover, Nidhi. "A study of various Fuzzy Clustering Algorithms." *International Journal of Engineering Research* 3.3 (2014): 177-81. Web.

[87] Almeida, R., and J. C. Sousa. "Comparison of fuzzy clustering algorithms for classification." *2006 International Symposium on Evolving Fuzzy Systems* (2006): n. pag. Web.

[88] Priyono, Agus, Muhammad Ridwan, Ahmad Jais Alias, Riza Atiq O. K. Rahmat, Azmi Hassan, and Mohd. Alauddin Mohd. Ali. "Generation of Fuzzy Rules with Subtractive Clustering." *Jurnal Teknologi* 43, no. 1 (2005). doi:10.11113/jt.v43.782.

[89] "8. Fuzzy Clustering Algorithms." *Data Clustering: Theory, Algorithms, and Applications* (2007): 151-59. Web.

[90] Vernieuwe, H., B. De Baets, and N.e.c. Verhoest. "Comparison of clustering algorithms in the identification of Takagi–Sugeno models: A hydrological case study." *Fuzzy Sets and Systems* 157.21 (2006): 2876-896. Web.

[91] Dave, R.n. "Robust fuzzy clustering algorithms." *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems* (n.d.): n. pag. Web.

[92] Omerhodzic, Ibrahim, Samir Avdakovic, Amir Nuhanovic, Kemal Dizdarevic, and Kresimir Rotim. "Energy Distribution of EEG Signal Components by Wavelet Transform." *Wavelet Transforms and Their Recent Applications in Biology and Geoscience* (2012): n. pag. Web.

[93] Delorme, Arnaud, and Scott Makeig. "EEGLAB: An Open Source Toolbox for Analysis of Single-trial EEG Dynamics including Independent Component Analysis." *Journal of Neuroscience Methods* 134.1 (2004): 9-21. Web. 29 July 2016.

[94] Metsomaa, Johanna, Jukka Sarvas, and Risto J. Ilmoniemi. "Blind Source Separation of Event-Related EEG/MEG." *IEEE Transactions on Biomedical Engineering* (2016): 1. Web. 31 Oct. 2016.

[95] Wang, Jiang, Guizhi Xu, Lei Wang, Huiyuan Zhang, and Jiang Wang. "Feature Extraction of Brain-Computer Interface Based on Second Order Blind

Identification and ICA." *2011 5th International Conference on Bioinformatics and Biomedical Engineering* (2011): n. pag. Web.

[96] Ryu, Shingo, Hiroshi Higashi, Toshihisa Tanaka, Shigeki Nakauchi, and Tetsuto Minami. "Spatial Smoothing of Canonical Correlation Analysis for Steady State Visual Evoked Potential Based Brain Computer Interfaces." *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2016): n. pag. Web.

[97] Dahal, Nabaraj, Nanda Nandagopal, Bernadine Cocks, Naga Dasari, Ramasamy Vijayalakshmi, and Paul Gaertner. "Event Related Synchronization and Desynchronization of EEG Data During Distracted Virtual Driving." *2013 7th Asia Modelling Symposium* (2016): n. pag. Web. 31 Oct. 2016.

[98] Yilmaz, O., W. Cho, C. Braun, N. Birbaumer, and A. Ramos-Murguialday. "Movement Related Cortical Potentials in Severe Chronic Stroke." *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2013): n. pag. Web. 31 Oct. 2016.

[99] Pfurtscheller, G., and W. Klimesch. "Functional Topography During a Visuoverbal Judgment Task Studied with Event-Related Desynchronization Mapping." *Journal of Clinical Neurophysiology* 9.1 (1992): 120-31. Web. 3 Nov. 2016.

[100] Ramoser, H., J. Muller-Gerking, and G. Pfurtscheller. "Optimal Spatial Filtering of Single Trial EEG during Imagined Hand Movement." *IEEE Transactions on Rehabilitation Engineering* 8.4 (2000): 441-46. Web. 3 Nov. 2016.

[101] Shih, Jerry J., Dean J. Krusienski, and Jonathan R. Wolpaw. "Brain-Computer Interfaces in Medicine." *Mayo Clinic Proceedings* 87.3 (2012): 268–279. *PMC*. Web. 25 July 2016.

[102] Qadeer, Shaik, Mohammed Zafar Ali Khan, Syed Abdul Sattar, and Ahmed. "A Radix-2 DIT FFT with Reduced Arithmetic Complexity." *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2014): n. pag. Web.

[103] Graimann, Bernhard, Brendan Allison, Gert Pfurtscheller, and Jane Huggins. *Brain-computer Interfaces: Revolutionizing Human-computer Interaction*. Heidelberg: Springer, 2010. Print.

[104] Niedermeyer, Ernst, Donald L. Schomer, and Lopes Da Silva F. H. *Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Philadelphia: Wolters Kluwer/Lippincott Williams & Wilkins Health, 2011. Print.

[105] Niemarkt, Hendrik J., Ward Jennekens, Jaco W. Pasman, Titia Katgert, Carola Van Pul, Antonio W D Gavilanes, Boris W. Kramer, Luc J. Zimmermann, Sidarto

Bambang Oetomo, and Peter Andriessen. "Maturational Changes in Automated EEG Spectral Power Analysis in Preterm Infants." *Pediatr Res Pediatric Research* 70.5 (2011): 529-34. Web. 25 July 2016.

[106] Zedeh, L.a. "Knowledge Representation in Fuzzy Logic." *IEEE Trans. Knowl. Data Eng. IEEE Transactions on Knowledge and Data Engineering* 1.1 (1989): 89-100. Web. 26 July 2016.

[107] H., Mohammad, Aya Samaha, and Khaled Alkamha. "Automated Classification of L/R Hand Movement EEG Signals Using Advanced Feature Extraction and Machine Learning." *International Journal of Advanced Computer Science and Applications IJACSA* 4.6 (2013): n. pag. Web. 29 July 2016.

[108] R. J. Wai and C. M. Li, "Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot," *IEEE Trans. Ind. Electron.*, vol. 56, no. 7, pp. 2667–2683, Jul. 2009.

[109] Güler, Inan, and Elif Derya Übeyli. "Adaptive Neuro-fuzzy Inference System for Classification of EEG Signals Using Wavelet Coefficients." *Journal of Neuroscience Methods* 148.2 (2005): 113-21. Web. 29 July 2016.

[110] Kim, Hyun Seok, Min Hye Chang, Hong Ji Lee, and Kwang Suk Park. "A Comparison of Classification Performance among the Various Combinations of Motor Imagery Tasks for Brain-computer Interface." *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)* (2013): n. pag. Web. 1 Sept. 2016.

[111] Goupillaud, A., Grossman, J. Morlet. "Cycle-Octave and Related Transforms in Seismic Signal Analysis." *Geoexploration* 23, 1984, 85-102

[112] G. Valenza, N. Vanello, M. Milanesi, E. P. Scilingo and L. Landini, "Decoding underlying brain activities in time and frequency domains through complex independent component analysis of EEG signals," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Chicago, IL, 2014, pp. 3192-3195.doi: 10.1109/EMBC.2014.6944301

[113] R. Kottaimalai, M. P. Rajasekaran, V. Selvam and B. Kannapiran, "EEG signal classification using Principal Component Analysis with Neural Network in Brain Computer Interface applications," *Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), 2013 International Conference on*, Tirunelveli, 2013, pp. 227-231.

APPENDICES

APPENDIX A – List of Acronyms

AAR – Automated Artifact Rejection
ANFIS – Adaptive Neuro-Fuzzy Inference System
BCI – Brain Computer Interface
BP – Back Propagation
BSS – Blind Source Separation
DFT – Discrete Fourier Transform
DWT – Discrete Wavelet Transform
EDF – European Data Format
EEG – Electroencephalography
EMD – Empirical Mode Decomposition
EMG – Electromyograms
EOG – Electrooculargrams
ERD – Event Related Desynchronization
ERS – Event Related Synchronization
ERP – Event Related Perturbation
FIS – Fuzzy Inference System
FFT – Fast Fourier Transform
GUI – Graphical User Interface
HHT – Hilbert-Huang Transform
ICA – Independent Component Analysis
SCP – Slow Cortical Potentials
SMR – Sensorimotor Rhythms
SVM – Support Vector Machine
LSE – Least Squares Error
MEMD – Multivariate Empirical Mode Decomposition
MMI – Motor Movement Imagery
TSKFNN – Takagi-Sugeno-Kang Fuzzy Neural Network

APPENDIX B – MATLAB Code

annot_auto.m

```
[ complete ] = annot_automate1( 'HandImagery9',[2.5,20] );clc

function [ complete ] = annot_automate1( data_type,bw )
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % %

% annot_automate This file is the file where annotations are added to be
% so that the new annotation reconstructions of (2.5-25 Hz) can be saved
% for processing through EEGLAB
% eeglab: cd '/Users/rory_donovan/Google
Drive/MATLAB/Thesis/databases/eeglab13_6_5b'

% to be used with {'eeglab_auto.m' 'create_event.m'}

% input
% data_type = file name for the data
% bw = band width of the FIR filter

% data_type = type of data being used (example 'HandImagery' or 'MotorImagery')
FILT = 0; % this set needs to go through filt because it is the continuous data
AAR = 0;  % this set needs to go through AAR because it is the continuous data
ICA = 0;  % this set needs to go through ICA because it is the continuous data

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % %
% Make Directories

% new EEG data path
data_type_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type];
mkdir(data_type_path) % makes new director with the name data_type

% new filtered data path
fir_filt_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[1]fir_filt/'];
mkdir(fir_filt_path) % makes new director with the name 'fir_filt'

% new AAR processed signals path
aar_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[2]aar/'];
mkdir(aar_path) % makes new director with the name 'aar_filt'
```

```
% new indiv. left/right hand data path
conc1_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[3]conc1/'];
mkdir(conc1_path) % makes new director with the name conc1

% new indiv. left/right ica hand data path
ica_all_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[4]ica_all/'];
mkdir(ica_all_path) % makes new director with the name ica_all

% new fir pic path
pic_fir_path = ['/Users/rory_donovan/Google Drive/Thesis/'...
   'My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[1]fir_filt/'];
mkdir(pic_fir_path) % makes new director with the name 'fir_filt'

% new aar pic path
pic_aar_path = ['/Users/rory_donovan/Google Drive/Thesis/'...
   'My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[2]aar/'];
mkdir(pic_aar_path) % makes new director with the name 'aar'

% new spectrogram pic path
pic_spectro_path = ['/Users/rory_donovan/Google Drive/Thesis/'...
   'My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[3]spectrogram/'];
mkdir(pic_spectro_path) % makes new director with the name 'spectro'

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % %
% Start Automation

% new path
cd(fir_filt_path) % open so it saves locally

% open path to read raw EEG files
% reconstruct raw signal into specified bandwidth, add events
sample = 'S001R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S001R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S001R12';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
```

```
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S002R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S002R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S002R12'
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S003R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S003R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S003R12';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S004R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S004R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S004R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
```

```
eeglab_auto( sample,data_type,AAR )

sample = 'S005R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S005R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S005R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S006R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S006R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S006R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S007R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S007R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S007R12';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S008R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S008R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S008R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S009R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S009R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S009R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S010R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S010R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S010R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S011R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S011R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S011R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S012R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S012R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S012R12'
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S013R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S013R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S013R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S014R04';
```

```
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S014R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S014R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S015R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S015R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S015R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S016R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S016R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S016R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S017R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
```

```
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S017R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S017R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S018R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S018R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S018R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S019R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S019R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S019R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S020R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
```

```
eeglab_auto( sample,data_type,AAR )

sample = 'S020R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S020R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S021R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S021R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S021R12';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S022R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S022R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S022R12'
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S023R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S023R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S023R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S024R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S024R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S024R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S025R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S025R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S025R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S026R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S026R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S026R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S027R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S027R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S027R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S028R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )


sample = 'S028R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )


sample = 'S028R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S029R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S029R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S029R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S030R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S030R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S030R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S031R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S031R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S031R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S032R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )
```

```
sample = 'S032R08';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S032R12'
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S033R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S033R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S033R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S034R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S034R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S034R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S035R04';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S035R08';
```

```
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S035R12';
val = create_event(data_type,FILT,sample,4.2,4.1,bw,...
   [0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S036R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S036R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S036R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S037R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S037R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S037R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S038R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S038R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
```

```
   [0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S038R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S039R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,2,0,1,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S039R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,1]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S039R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S040R04';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,1,0,2,0,1,0,2,0,2,0,1,0,1,0,2,0,1,0,2,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S040R08';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,1,0,2,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

sample = 'S040R12';
val = create_event(data_type,FILT,sample,4.1,4.1,bw,...
   [0,2,0,1,0,1,0,2,0,2,0,1,0,2,0,1,0,2,0,1,0,1,0,2,0,2,0,1,0,2]);cd(fir_filt_path)
eeglab_auto( sample,data_type,AAR )

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
% Notes - this part of file concatenates the dataset of the the trials of a single
% person into 1 long l/r hand data set for each person.

eeglab     % clear EEGLAB workspace
close(gcf) % close EEGLAB figure
```

```
% load eeg sets
ALLEEG = pop_loadset('filename',{'S001R04_T1.set' 'S001R04_T2.set'
'S001R08_T1.set' 'S001R08_T2.set' 'S001R12_T1.set' 'S001R12_T2.set'...
                    'S002R04_T1.set' 'S002R04_T2.set' 'S002R08_T1.set'
'S002R08_T2.set' 'S002R12_T1.set' 'S002R12_T2.set'...
                    'S003R04_T1.set' 'S003R04_T2.set' 'S003R08_T1.set'
'S003R08_T2.set' 'S003R12_T1.set' 'S003R12_T2.set'...
                    'S004R04_T1.set' 'S004R04_T2.set' 'S004R08_T1.set'
'S004R08_T2.set' 'S004R12_T1.set' 'S004R12_T2.set'...
                    'S005R04_T1.set' 'S005R04_T2.set' 'S005R08_T1.set'
'S005R08_T2.set' 'S005R12_T1.set' 'S005R12_T2.set'...
                    'S006R04_T1.set' 'S006R04_T2.set' 'S006R08_T1.set'
'S006R08_T2.set' 'S006R12_T1.set' 'S006R12_T2.set'...
                    'S007R04_T1.set' 'S007R04_T2.set' 'S007R08_T1.set'
'S007R08_T2.set' 'S007R12_T1.set' 'S007R12_T2.set'...
                    'S008R04_T1.set' 'S008R04_T2.set' 'S008R08_T1.set'
'S008R08_T2.set' 'S008R12_T1.set' 'S008R12_T2.set'...
                    'S009R04_T1.set' 'S009R04_T2.set' 'S009R08_T1.set'
'S009R08_T2.set' 'S009R12_T1.set' 'S009R12_T2.set'...
                    'S010R04_T1.set' 'S010R04_T2.set' 'S010R08_T1.set'
'S010R08_T2.set' 'S010R12_T1.set' 'S010R12_T2.set'...
                    'S011R04_T1.set' 'S011R04_T2.set' 'S011R08_T1.set'
'S011R08_T2.set' 'S011R12_T1.set' 'S011R12_T2.set'...
                    'S012R04_T1.set' 'S012R04_T2.set' 'S012R08_T1.set'
'S012R08_T2.set' 'S012R12_T1.set' 'S012R12_T2.set'...
                    'S013R04_T1.set' 'S013R04_T2.set' 'S013R08_T1.set'
'S013R08_T2.set' 'S013R12_T1.set' 'S013R12_T2.set'...
                    'S014R04_T1.set' 'S014R04_T2.set' 'S014R08_T1.set'
'S014R08_T2.set' 'S014R12_T1.set' 'S014R12_T2.set'...
                    'S015R04_T1.set' 'S015R04_T2.set' 'S015R08_T1.set'
'S015R08_T2.set' 'S015R12_T1.set' 'S015R12_T2.set'...
                    'S016R04_T1.set' 'S016R04_T2.set' 'S016R08_T1.set'
'S016R08_T2.set' 'S016R12_T1.set' 'S016R12_T2.set'...
                    'S017R04_T1.set' 'S017R04_T2.set' 'S017R08_T1.set'
'S017R08_T2.set' 'S017R12_T1.set' 'S017R12_T2.set'...
                    'S018R04_T1.set' 'S018R04_T2.set' 'S018R08_T1.set'
'S018R08_T2.set' 'S018R12_T1.set' 'S018R12_T2.set'...
                    'S019R04_T1.set' 'S019R04_T2.set' 'S019R08_T1.set'
'S019R08_T2.set' 'S019R12_T1.set' 'S019R12_T2.set'...
                    'S020R04_T1.set' 'S020R04_T2.set' 'S020R08_T1.set'
'S020R08_T2.set' 'S020R12_T1.set' 'S020R12_T2.set'...
                    'S021R04_T1.set' 'S021R04_T2.set' 'S021R08_T1.set'
'S021R08_T2.set' 'S021R12_T1.set' 'S021R12_T2.set'...
                    'S022R04_T1.set' 'S022R04_T2.set' 'S022R08_T1.set'
'S022R08_T2.set' 'S022R12_T1.set' 'S022R12_T2.set'...
                    'S023R04_T1.set' 'S023R04_T2.set' 'S023R08_T1.set'
```

```
'S023R08_T2.set' 'S023R12_T1.set' 'S023R12_T2.set'...
                    'S024R04_T1.set' 'S024R04_T2.set' 'S024R08_T1.set'
'S024R08_T2.set' 'S024R12_T1.set' 'S024R12_T2.set'...
                    'S025R04_T1.set' 'S025R04_T2.set' 'S025R08_T1.set'
'S025R08_T2.set' 'S025R12_T1.set' 'S025R12_T2.set'...
                    'S026R04_T1.set' 'S026R04_T2.set' 'S026R08_T1.set'
'S026R08_T2.set' 'S026R12_T1.set' 'S026R12_T2.set'...
                    'S027R04_T1.set' 'S027R04_T2.set' 'S027R08_T1.set'
'S027R08_T2.set' 'S027R12_T1.set' 'S027R12_T2.set'...
                    'S028R04_T1.set' 'S028R04_T2.set' 'S028R08_T1.set'
'S028R08_T2.set' 'S028R12_T1.set' 'S028R12_T2.set'...
                    'S029R04_T1.set' 'S029R04_T2.set' 'S029R08_T1.set'
'S029R08_T2.set' 'S029R12_T1.set' 'S029R12_T2.set'...
                    'S030R04_T1.set' 'S030R04_T2.set' 'S030R08_T1.set'
'S030R08_T2.set' 'S030R12_T1.set' 'S030R12_T2.set'...
                    'S031R04_T1.set' 'S031R04_T2.set' 'S031R08_T1.set'
'S031R08_T2.set' 'S031R12_T1.set' 'S031R12_T2.set'...
                    'S032R04_T1.set' 'S032R04_T2.set' 'S032R08_T1.set'
'S032R08_T2.set' 'S032R12_T1.set' 'S032R12_T2.set'...
                    'S033R04_T1.set' 'S033R04_T2.set' 'S033R08_T1.set'
'S033R08_T2.set' 'S033R12_T1.set' 'S033R12_T2.set'...
                    'S034R04_T1.set' 'S034R04_T2.set' 'S034R08_T1.set'
'S034R08_T2.set' 'S034R12_T1.set' 'S034R12_T2.set'...
                    'S035R04_T1.set' 'S035R04_T2.set' 'S035R08_T1.set'
'S035R08_T2.set' 'S035R12_T1.set' 'S035R12_T2.set'...
                    'S036R04_T1.set' 'S036R04_T2.set' 'S036R08_T1.set'
'S036R08_T2.set' 'S036R12_T1.set' 'S036R12_T2.set'...
                    'S037R04_T1.set' 'S037R04_T2.set' 'S037R08_T1.set'
'S037R08_T2.set' 'S037R12_T1.set' 'S037R12_T2.set'...
                    'S038R04_T1.set' 'S038R04_T2.set' 'S038R08_T1.set'
'S038R08_T2.set' 'S038R12_T1.set' 'S038R12_T2.set'...
                    'S039R04_T1.set' 'S039R04_T2.set' 'S039R08_T1.set'
'S039R08_T2.set' 'S039R12_T1.set' 'S039R12_T2.set'...
                    'S040R04_T1.set' 'S040R04_T2.set' 'S040R08_T1.set'
'S040R08_T2.set' 'S040R12_T1.set' 'S040R12_T2.set'...
                    },'filepath',aar_path);

ALLEEG = eeg_checkset( ALLEEG ); % verify

% concatenate sets

% T1 = left hand
task = '1';
for idx1 = 1:40 % each subject
   fn = ['S',num2str(idx1),'T'];
   EEG = pop_mergeset( ALLEEG, (idx1-1)*3+(1:2:5), 0);
```

```
   EEG = eeg_checkset( EEG );
   EEG = pop_saveset( EEG, 'filename',[fn,task,'.set'],'filepath',conc1_path);
end

% T2 = right hand
task = '2';
for idx2 = 1:40
   fn = ['S',num2str(idx2),'T'];
   EEG = pop_mergeset( ALLEEG, (idx2-1)*3+(2:2:6), 0);
   EEG = eeg_checkset( EEG );
   EEG = pop_saveset( EEG, 'filename',[fn,task,'.set'],'filepath',conc1_path);
end

% Notes - this part of the file concatenates the dataset of the the trials of all of
% the people into 1 long l/r hand data set for each person.
if ICA == 1

eeglab     % clear EEGLAB workspace
close(gcf) % close figure

% load EEG sets
ALLEEG = pop_loadset('filename',{'S1T1.set' 'S1T2.set'...
                        'S2T1.set' 'S2T2.set'...
                        'S3T1.set' 'S3T2.set'...
                        'S4T1.set' 'S4T2.set'...
                        'S5T1.set' 'S5T2.set'...
                        'S6T1.set' 'S6T2.set'...
                        'S7T1.set' 'S7T2.set'...
                        'S8T1.set' 'S8T2.set'...
                        'S9T1.set' 'S9T2.set'...
                        'S10T1.set' 'S10T2.set'...
                        'S11T1.set' 'S11T2.set'...
                        'S12T1.set' 'S12T2.set'...
                        'S13T1.set' 'S13T2.set'...
                        'S14T1.set' 'S14T2.set'...
                        'S15T1.set' 'S15T2.set'...
                        'S16T1.set' 'S16T2.set'...
                        'S17T1.set' 'S17T2.set'...
                        'S18T1.set' 'S18T2.set'...
                        'S19T1.set' 'S19T2.set'...
                        'S20T1.set' 'S20T2.set'...
                        'S21T1.set' 'S21T2.set'...
                        'S22T1.set' 'S22T2.set'...
                        'S23T1.set' 'S23T2.set'...
                        'S24T1.set' 'S24T2.set'...
                        'S25T1.set' 'S25T2.set'...
```

```
                    'S26T1.set' 'S26T2.set'...
                    'S27T1.set' 'S27T2.set'...
                    'S28T1.set' 'S28T2.set'...
                    'S29T1.set' 'S29T2.set'...
                    'S30T1.set' 'S30T2.set'...
                    'S31T1.set' 'S31T2.set'...
                    'S32T1.set' 'S32T2.set'...
                    'S33T1.set' 'S33T2.set'...
                    'S34T1.set' 'S34T2.set'...
                    'S35T1.set' 'S35T2.set'...
                    'S36T1.set' 'S36T2.set'...
                    'S37T1.set' 'S37T2.set'...
                    'S38T1.set' 'S38T2.set'...
                    'S39T1.set' 'S39T2.set'...
                    'S40T1.set' 'S40T2.set'...
                    },'filepath',conc1_path);

ALLEEG = eeg_checkset( ALLEEG ); % verify

% concatenate & save
EEG = pop_mergeset( ALLEEG, 1:size(ALLEEG,2), 0);
EEG = eeg_checkset( EEG ); % verify
EEG = pop_saveset( EEG, 'filename','conc_eeg.set','filepath',ica_all_path);
EEG = eeg_checkset( EEG );

eeglab
close(gcf)

EEG = pop_loadset('filename','conc_eeg.set','filepath',['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[4]ica_all/']);
EEG = eeg_checkset( EEG );
EEG = pop_runica(EEG, 'extended',1,'interupt','off');
EEG = eeg_checkset( EEG );
EEG = pop_saveset( EEG, 'filename','ica.set','filepath',ica_all_path);
EEG = eeg_checkset( EEG );
end
complete = 'complete'

end

%--------------------------------------------------------------------------
function [ val ] = create_event( data_type,filt,sample,n1,n2,bw,annotations )
%create_event This function decomposes and recomposes the raw 160 Hz EEG
%   signal using the DWT and adds an annotation file. The output signals
%   are of the specified bandwidth range
```

```
%   {Function used with the file 'save_annot_imagery.m'}

%   Parameters -
%   sample = sample name (e.g 'S001R12')
%   n1 = # of seconds of first occuring event (e.g 4.1 or 4.2)
%   n2 = # of seconds of second occuring event (e.g 4.1 or 4.2)
%   bw = FIR filter bandwidtch, [lower,higher]
%   annotations = annotated event channel - e.g [1,1,1,1...,2,2,...]

%   Example of use -
%   [ val ] = create_event( file,4.1,4.2,'db2',6,[0,1,0,2,...])
%   This decomposes the signals into  6 levels using debauchees with 2
%   vanishing moments.

% extract bandwidth values, note upper>lower
upper = bw(2); % low pass frequency
lower = bw(1);  % high pass frequency

% import data
EEG =
pop_importdata('dataformat','matlab','nbchan',0,'data',['/Users/rory_donovan/Google
Drive/MATLAB/Thesis/databases/eeglab13_6_5b/eegmmidb/Raw/',sample,'_edfm.mat'],'
srate',160,'pnts',0,'xmin',0);
EEG.setname=sample;         % give set a name
EEG = eeg_checkset( EEG );  % verify set

% add events to recomposed wave
if n1 == 4.2 && n2 == 4.1
   for i = 1:15
      annot_channel((i-1)*(4.1+4.2)*(160)+1:(i-1)*(4.1+4.2)*(160)+(4.2*160)) =
annotations(i*2-1);
      annot_channel((i-1)*(4.1+4.2)*(160)+(4.2*160)+1:(i-
1)*(4.1+4.2)*(160)+(4.2*160)+(4.1*160)) =  annotations(i*2);
   end
elseif n1 == 4.1 && n2 == 4.1
   for i = 1:30
      annot_channel((i-1)*4.1*160+1:i*4.1*160) = repmat(annotations(i),1,4.1*160);
   end
end

EEG.data=EEG.data(:,1:size(annot_channel,2));
EEG.data(65,:)=annot_channel;

% following needed to graph
EEG =
pop_importdata('dataformat','matlab','nbchan',0,'data',EEG.data,'srate',160,'pnts',0,'xmin',
```

```
0);
EEG.setname=sample;
EEG = eeg_checkset( EEG );

% event channel - leading, including duration
EEG = pop_chanevent(EEG, 65,'edge','leading','edgelen',0,'duration','on');
EEG = eeg_checkset( EEG );

% add channel locations for 2D viewing
EEG=pop_chanedit(EEG, 'lookup','/Users/rory_donovan/Google
Drive/MATLAB/Thesis/databases/eeglab13_6_5b/plugins/dipfit2.3/standard_BESA/stan
dard-10-5-cap385.elp','load',{'/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/chan_locs.ced' 'filetype'
'autodetect'});
EEG = eeg_checkset( EEG );
EEG=pop_chanedit(EEG, 'plotrad',0.5); % change radius to that of head
EEG = eeg_checkset( EEG );          % verify set

if filt == 1
% check it out
% create log power spectrum
figure;fig1 = pop_spectopo(EEG, 1, [0       4093.75], 'EEG' , 'percent', 15, 'freq', [10 25],
'freqrange',[2 50],'electrodes','off');
close(gcf)                      % close current figure

% Hamming Windowed Sinc FIR filter
EEG = pop_eegfiltnew(EEG, lower, upper, 264, 0, [], 1);
val = [EEG.data;annot_channel];     % store filtered data, w/ event channel
save([sample,'.mat'],'val');
close(gcf)                      % close figure
EEG = eeg_checkset( EEG );          % verify set

% create log power spectrum
figure;fig2 = pop_spectopo(EEG, 1, [0       4093.75], 'EEG' , 'percent', 15, 'freq', [],
'freqrange',[0 50],'electrodes','off');
close(gcf)                      % close figure
freq = linspace(0,50,(30));       % scale to proper size

% plot log-power spectral density
fig = figure(1);
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 10 2];
title('Hamming Windowed Sinc Filter')
hold on
subplot(1,2,1)
plot(freq,fig1(:,1:30)')
```

```
title('Before FIR Filtration')
ylabel('Log Power Specral Density 10*log(µV^2/Hz)','FontSize',6)
xlabel('frequency (Hz)')
subplot(1,2,2)
plot(freq,fig2(:,1:30)')
title('After FIR Filtration')
ylabel('Log Power Specral Density 10*log(µV^2/Hz)','FontSize',6)
xlabel('frequency (Hz)')
hold off

cd(['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[1]fir_filt/']);
saveas(gcf,[sample,'_fir.jpg'])
close(gcf)
end
end

%---------------------------------------------------------------------
function [ complete ] = eeglab_auto( sample,data_type,AAR )
%eeglab_auto EEG automation function
%   This file imports the data, creates events from the event channel, adds
%   channel labels for 2S viewing, AARs the EOG and EMG & removes baseline

%   {to be used with 'annot_auto.m'}

%   sample = subject (example 'S001R04')
%   data_type = type of data being used (example 'HandImagery')
%   AAR = (BOOLEAN) whether to run AAR or not

% parameters
freq = 160;  % sampling frequency Hz
n_chan = 64; % number of channels (annotation file is channel 65)

fir_filt_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[1]fir_filt/',samp
le,'.mat'];
aar_path = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[2]aar/'];

% import Hamming Windowed Sinc FIR filtered data
EEG =
pop_importdata('dataformat','matlab','nbchan',0,'data',fir_filt_path,'srate',freq,'pnts',0,'xmi
n',0);
EEG.setname=sample;
EEG = eeg_checkset( EEG ); % verify
```

```
% event channel - from leading edge, include duration
EEG = pop_chanevent(EEG, n_chan+1,'edge','leading','edgelen',0,'duration','on');
EEG = eeg_checkset( EEG );

% add channel locations for 2D viewing
EEG=pop_chanedit(EEG, 'lookup','/Users/rory_donovan/Google
Drive/MATLAB/Thesis/databases/eeglab13_6_5b/plugins/dipfit2.3/standard_BESA/stan
dard-10-5-cap385.elp','load',{'/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/chan_locs.ced' 'filetype'
'autodetect'});
EEG = eeg_checkset( EEG );
EEG=pop_chanedit(EEG, 'plotrad',0.5); % change radius to that of head
EEG = eeg_checkset( EEG );

% re-reference the data (remove 40dB of noise)
EEG = pop_reref( EEG, []);   % re-reference large window to zero mean
EEG = eeg_checkset( EEG );   % verify
fig1 = EEG.data(:,:,1);      % plot 1 epoch to check before AAR

if AAR == 1 % only run if  user specifies

   % EOG artifact removal using bss (sobi) algorith
   EEG = pop_autobsseog( EEG, [EEG.xmax], [EEG.xmax], 'sobi', {'eigratio',
[1000000]},...
      'eog_fd', {'range',[2  21]});
   EEG = eeg_checkset( EEG );

   % EMG artifact removal using bss (bsscca) algorithm
   EEG = pop_autobssemg( EEG, [81.9188], [81.9188], 'bsscca', ...
      {'eigratio', [1000000]}, 'emg_psd', {'ratio', [10],'fs', [160],'femg', [15],...
      'estimator',spectrum.welch({'Hamming'}, 80),'range', [0  32]});

   % save data for figure
   EEG = eeg_checkset( EEG );                % verify
   fig2=EEG.data(:,:,1);                     % plot 1 epoch to check after AAR
   x = linspace(0,45*2*4.1,size(EEG.data(:,:,1),2)); % scale plot to EEGLAB true value

end

% plot
fig = figure(1);                  % setup figure
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 10 2];
title('Automated Artifact Rejection')
hold on
subplot(1,2,1)                    % plot before AAR
```

```
plot(x,fig1')
title('Before AAR')
xlabel('time(s)')
ylabel('Potential (µV)')
subplot(1,2,2)                    % plot after AAR
plot(x,fig2')
title('After AAR')
xlabel('time(s)')
ylabel('Potential (µV)')
hold off

% save to desired location
cd(['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[2]aar/']);
saveas(gcf,[sample,'_aar.jpg'])       % save AAR before/after
close(gcf)                        % close current figure

cd(['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Pictures/',data_type,'/[3]spectrogram/'
]);
EEG_temp = EEG; % store for use by both epoching
epoch_range = [0,4.1];
for idx1 = 1:2 % each event
   % 4.1 sec epoch the data for each event, remove baseline, save
   hand = num2str(idx1);
   EEG = pop_epoch( EEG_temp, { idx1 }, epoch_range, 'newname', [sample,'_T',hand],
'epochinfo', 'yes');
   EEG = eeg_checkset( EEG );
   EEG = pop_rmbase( EEG, epoch_range*1000); % remove baseline
   EEG = eeg_checkset( EEG );
   EEG = pop_saveset( EEG, 'filename',[sample,'_T',hand,'.set'],'filepath',aar_path);
   EEG = eeg_checkset( EEG );

   SNR = 20; %db
   for idx2 = 1:size(EEG.data,3)
      elect = [9,11,13];
      for idx3 = 1:3
         sfig =
[sample,'_M',num2str(idx2),'_H',num2str(idx1),'_E',num2str(elect(idx3)),'.jpg'];
         fig = figure(1);
         title(sfig)

spectrogram(EEG.data(elect(idx3),:,idx2),100*hamming(64),60,[1:.2:20],freq,'MinThres
hold',-SNR,'yaxis')
         myscale = [linspace(1,0,24)',ones(24,2)                    ;...
                  linspace(0,1,12)',ones(12,1)      , linspace(1,0,12)';...
```

```
                ones(12,1)      ,linspace(1,0,12)', zeros(12,1)     ];
            colormap(myscale)
            fig.PaperPosition = [0 0 6 2];
            saveas(gcf,sfig)
            close(gcf)
        end
    end

end

cd(['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',data_type,'/[1]fir_filt/'])
clc
complete = sample;

end
```

APPENDIX B – MATLAB Code

tskfnn.m

```
clc
[ av_perf,tmp,mode,conf_mat,perf,n_test,tp2tn ] =
tskfnn1([2],'db2',10,[2,0.1,0.9,1.1],'HandImagery1','/[3]conc1/' );
performance = [tmp;100*ones(50,4)];

function [ av_perf,performance,mode,conf_mat,perf,n_test,tp2tn ] = tskfnn1(
S,w,n_sym,trn_opt,data_type,path )
%tskfnn Function for l/r hand (motor) classification of EEG data
%   This function takes user information and neural network options to
%   train a takagi-sugeno-kang n`eural network to classify l/r hand data
%   from EEG data sets that have undergone AAR and ICA decomposition and
%   are concatenated into segments of 1 user.
%   Example - [ av_perf,performance,mode,conf_mat,perf,n_test,tp2tn ]
%             = tskfnn([1:40],'db2',[10,NaN,0.009,0.9975,1.0025],1,'HandImagery' );
%   electrodes used in ICA
2,4,6,9,11,13,22,24,30,32,34,36,38,41,42,43,44,47,49,51,53,55,61,63

% open EEGLAB session
cd '/Users/rory_donovan/Google Drive/MATLAB/Thesis/databases/eeglab13_6_5b'
eeglab
close(gcf)

%   S = # subjects being used (example: [1:10])
%   n_sym = # of simulations run (to test accuracy)
%   w = wavelet mother function (example: 'db4', etc...)
%   trn_opt = training options (example: [epoch#,error,eta,])
%   data_type = type of data (example: HandImagery)
%   conf_mat = array with columns representing tp and tn by subject
%   perf = average percent tp and tn
%   n_test = array where columns are # of subjects tested
%   tp2tn = array where columns are [TP,FP,FN,TN]
%   performance = [spec,sen,sel,acc]
%   add_id = (bool) whether or not to add identifier on all subjects
%   ann = (bool) indicate whether the training is bp (0) or anfis (1)

% init
ann = 1;
add_id = 0;
acc(1:n_sym) = 0; spec(1:n_sym) = 0; sen(1:n_sym) = 0; sel(1:n_sym) = 0; % init

% calc. perf
```

```
for idx1 = 1:n_sym
   [ performance(idx1,:),id_inc,conf_mat(idx1,:),n_test(idx1,:),tp2tn(idx1,:) ]...
      = eeg_class_dwt2anf( S, w, trn_opt, data_type,add_id,ann,path );

   % give feedback each simulation
   fprintf('performance %.0f = %.2f %.2f %.2f %.2f \n',idx1, performance(idx1,:));
   inc_matr(idx1,1:size(id_inc,2)) = id_inc; % saver erroneous trials
end

% find mode (to determine if the is any problems with the part. trial)
occ(1:56) = 0; % init
for idx2 = 1:56
   occ(idx2) = sum(inc_matr(:)==idx2); % occurence of each #
end
[~,mode] = sort(occ(:),'descend'); % sort by worst performing trials in desc. order

% display results
perf = mean(conf_mat,1);
av_perf = mean(performance,1,'omitnan');
fprintf('[spec,sen,sel,acc] = [%.2f,%.2f,%.2f,%.2f] \n', av_perf);
end
%--------------------------------------------------------------------------
function [ performance,id_inc,conf_mat,n_test,tp2tn ] = ...
   eeg_class_dwt2anf( S, w, trn_opt, data_type, add_id,ann,path )
%dwt2anf Feature extract using dwt and run through anfis
%   This file takes the concatenated data file and extracts energy levels
%   (5-10, 10-20, and 20-40 Hz), power (), and enters it into a
%   Takaki-Sugeno style fuzzy inference network.

%   { precursor file: 'tskfnn.m'}
%   { subsequent file: 'eeg_data_extr.m' and 'eeg_anfis' or 'bpnn.m'}

%   S = subject number
%   omit = trial numbers to omit
%   w = wavelet type (example, 'db4', etc...)
%   trn_opt = training options (example: [epoch#,error,eta,])
%   fp = file path

fp = ['/Users/rory_donovan/Google
Drive/Thesis/My_Algorithm/eeglab13_6_5b/EEG_Datasets/',...
   data_type,path]; % location
p = []; t = []; % init
for idx1 = 1:40 % for each subject
   for idx2 = 1:2 % for each task (T1 & T2)
      if ismember(idx1,S) % if the subject has been entered
         % load data
```

```
        fn = ['S',num2str(idx1),'T',num2str(idx2),'.set'];
        EEG = pop_loadset('filename',fn,'filepath',fp);
        EEG = eeg_checkset( EEG );

        % add identifier to the data
        id(1:size(EEG.data,3)) = idx1;
        p(:,end+(1:size(EEG.data,3))) = [eeg_data_extr( EEG.data,[2,4,6],w);id];
        t(end+(1:size(EEG.data,3))) = idx2; % add matrix sub partition
        clear EEG id fn %
      end
   end
end
if ann == 1 % anfis
   [ performance,id_inc,conf_mat,n_test,tp2tn ] = eeg_anfis( p, t, trn_opt, add_id );
elseif ann == 0 % nn w/ bp
   [ performance,id_inc,conf_mat,n_test,tp2tn ] = eeg_bpnn( p, t, add_id );
end; end
%--------------------------------------------------------------------------
function [ p ] = eeg_data_extr( data,e,w )
%dwt_ext DWT decomposition into 5-10,10-20,and 20-40 Hz energy levels
%   This file extracts features corresponding to 5-10,10-20,and 20-40 Hz
%   energy levels. The are to be used as the input to the neural network.
%   The inputs go in ascending order as far as electrodes and are generated
%   1 electrode at a time
%   Example: [feat1,feat2,etc...,feat6] = [e1 5-10 Hz, e1 10-20 Hz, ... ,
%   e2 20-40 Hz]

%   { previous file = 'eeg_class_dwt.m'}
%   { no subsequent file }

%   data = EEG data file size ()x()x().
%   l = levels
%   e = electrodes being used
%   w = wavelet type

% Parameters
l = 6; % levels used
j = 0; % variable for coutning
d = 4; % detail to start with
perc_en = []; % init

% DWT
for idx1 = 1:size(data,3) % for each epoch
   for idx2 = 1:64 % for each electrode
      if ismember(idx2,e)
         j=j+1;
```

```
        % Decomposition
        [C1,L1] = wavedec(data(idx2,:,idx1),l,w); % (l) level decomp of wt (db2 or 4 etc...)

        % coefficients & energy of details
        for idx3 = 1:l
           coeff{idx3} = wrcoef('d',C1,L1,w,idx3); % extract detail
           energy(idx3) = sum(coeff{idx3}.^2); % energy for detail
           if idx3 == l
              coeff{idx3+1} = wrcoef('a',C1,L1,w,idx3); % extract approx
              energy(idx3+1) = sum(coeff{idx3+1}.^2); % energy for approx
           end
        end
        % percent energy ( energy of level / total)
        perc_en((l-d+1)*(j-1)+(1:(l-d+1)),1) = energy(d:l)/sum(energy,2);
        end
        clear coeff energy C1 L1;
    end
    p(:,idx1) = perc_en;
    j=0;
    perc_en = [];
end
end
%--------------------------------------------------------------------------
function [ performance,id_inc, conf_mat,n_test,tp2tn ] = eeg_anfis( p, t, TRNOPT,
add_id )
%eeg_anfis Takagi-Sugeno fuzzy inference neural network with Gaussian mf
%   This function classifies movement based input features using symmetric
%   Gaussian membership functions. The learning is a LSE-BP hybrid

%   { precursor file = 'eeg_class_dwt.m'}
%   { last file in the sequence }

%   p = input
%   t = target
%   perf = performance
%   id_inc = identifier for the incorrectly classified data point. Used to
%            check where the error comes from.

if add_id == 0 || size(p,2)<=50
    p_temp = p(end,:); % store id
    p(end,:) = []; % delete identifiers
end

t(find(t == 1))=-.9;
t(find(t == 2))=.9;
```

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Neural Network

% randomely divide the test data
[trainInd,checkInd,testInd] = dividerand(size(p,2),0.7,0.15,0.15);

% create takagi-sugeno type inference neural network
trnData = [p(1:size(p,1),trainInd);t(trainInd)]';
chkData = [p(1:size(p,1),checkInd);t(checkInd)]';
testData = [p(1:size(p,1),testInd);t(testInd)]';
%testInd = [(ind-1)*45+(1:45)];

genOpt = genfisOptions('SubtractiveClustering','ClusterInfluenceRange',[0.5]);
tks_fis  = genfis(p(1:size(p,1),trainInd)',t(trainInd)',genOpt);

opt = anfisOptions('InitialFIS',tks_fis,'EpochNumber',TRNOPT(1));
opt.ValidationData = chkData;
opt.DisplayANFISInformation = 0;
opt.DisplayFinalResults = 0;
opt.DisplayErrorValues = 0;
opt.DisplayStepSize = 0;
opt.InitialStepSize = TRNOPT(2);
opt.StepSizeDecreaseRate = TRNOPT(3);
opt.StepSizeIncreaseRate = TRNOPT(4);

[~,error,~,chk_fis,chkerror] = anfis(trnData,opt);
Y = evalfis(p(1:size(p,1),testInd)',chk_fis);

figure()
hold on
plot(chkerror,'k')
plot(error,'r')
hold off

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% performance individual trials

% thresholding
Y(Y(:,1)>=0,1)=.9;
Y(Y(:,1)<0,1)=-.9;
target(:,1) = t(testInd);
output = [Y,target]; % compare what you got to the actual
err = Y-target; % output - target
err(find(abs(err(:,1))>0),1)=1; % find where the target isn't the output

% True/false positive/negative (TP, FP, TN, FN)
```

```
TP = 0;FP = 0;TN = 0;FN = 0; % init
for idx = 1:size(target,1)
   if target(idx,1) == .9 && Y(idx,1) == .9 % True positive
      TP = TP + 1;
   elseif target(idx,1) == -.9 && Y(idx,1) == .9 % False positive
      FP = FP + 1;
   elseif target(idx,1) == -.9 && Y(idx,1) == -.9 % True negative
      TN = TN + 1;
   elseif target(idx,1) == .9 && Y(idx,1) == -.9 % False negative
      FN = FN + 1;
   end
end
tp2tn = [TP,FP,FN,TN];

% specificity, sensitivity, selectivity, accuracy
performance(1) = 100*(TN/(TN+FP)); % specifity
performance(2) = 100*(TP/(TP+FN)); % sensitivity
performance(3) = 100*(TP/(TP+FP)); % selectivity
performance(4) = 100*((TN+TP)/(TN+TP+FP+FN)); % accuracy

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%  performance group trials

% confusion matrix for running all of the subjects together.

% init.
conf_mat1(1:40) = 0; % correct
conf_mat2(1:40) = 0; % incorrect

% add id back.
if add_id == 0 || size(p,2)<= 50
   p(end+1,:) = p_temp;
end

% create a confusion matrix for output.
for idx = 1:size(target,1)
   if target(idx,1) == .9 && Y(idx,1) == .9 % True positive
         conf_mat1(p(end,testInd(idx))) = conf_mat1(p(end,testInd(idx)))+1;
   elseif target(idx,1) == -.9 && Y(idx,1) == .9 % False positive
         conf_mat2(p(end,testInd(idx))) = conf_mat2(p(end,testInd(idx)))+1;
   elseif target(idx,1) == -.9 && Y(idx,1) == -.9 % True negative
         conf_mat1(p(end,testInd(idx))) = conf_mat1(p(end,testInd(idx)))+1;
   elseif target(idx,1) == .9 && Y(idx,1) == -.9 % False negative
         conf_mat2(p(end,testInd(idx))) = conf_mat2(p(end,testInd(idx)))+1;
   end
end
```

```
% calculate correct versus incorrect
conf_mat(1:40) = 0;
for idx = 1:40
   n_test(idx) = (conf_mat1(idx)+conf_mat2(idx));
   conf_mat(idx) = conf_mat1(idx)/n_test(idx)*100;
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% mode error individual trials

% incorrectly identified
temp = find(err==1);
for idx = 1:size(testInd)
   id_inc = testInd(temp);
end
close(gcf)
end
%-----------------------------------------------------------------------
function [ performance,id_inc, conf_mat,n_test,tp2tn ] = eeg_bpnn( p, t, add_id )
%bp_nn Supplemental training method for tskfnn to test performance
%   This file calcs perf=[specificity, sensitivity, selectivity, accuracy]
%   and the mode of error of the subject trials

%   { precursor file = 'eeg_class_dwt2anf.m' }
%   { no subsequent file }

%   p = input
%   t = target
%   perf = performance
%   id_inc = identifier for the incorrectly classified data point. Used to
%           check where the error comes from.

if add_id == 0 || size(p,2)<=50
   p_temp = p(end,:); % store id
   p(end,:) = []; % delete identifiers
end

t(find(t == 1))=-.9;
t(find(t == 2))=.9;

% neural network
net = feedforwardnet(100,'traingd'); % 100 hidden neurons
[trainInd,chkInd,testInd] = dividerand(size(p,2),0.85,0,0.15); % divide data
net.trainParam.showCommandLine = false;
net.trainParam.showWindow = false;
```

```
net.trainParam.show = NaN;
[net,tr] = train(net,p(:,trainInd),t(trainInd)); % train
Y = net(p(:,testInd)); % test
net

figure(1)
plotperform(tr)


% thresholding
Y(find(Y>=0))=.9;
Y(find(Y<0))=-.9;
Y = Y';

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% performance individual trials
target(:,1) = t(testInd);
err = Y-target; % output - target
err(find(abs(err(:,1))>0),1)=1; % find where the target isn't the output

% True/false positive/negative (TP, FP, TN, FN)
TP = 0;FP = 0;TN = 0;FN = 0; % init
for idx = 1:size(target,1)
   if target(idx,1) == .9 && Y(idx,1) == .9 % True positive
      TP = TP + 1;
   elseif target(idx,1) == -.9 && Y(idx,1) == .9 % False positive
      FP = FP + 1;
   elseif target(idx,1) == -.9 && Y(idx,1) == -.9 % True negative
      TN = TN + 1;
   elseif target(idx,1) == .9 && Y(idx,1) == -.9 % False negative
      FN = FN + 1;
   end
end
tp2tn = [TP,FP,FN,TN];

% specificity, sensitivity, selectivity, accuracy
performance(1) = 100*(TN/(TN+FP)); % specicity
performance(2) = 100*(TP/(TP+FN)); % sensitivity
performance(3) = 100*(TP/(TP+FP)); % selectivity
performance(4) = 100*(1-mean(err)); % accuracy

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% performance group trials

% confusion matrix for running all of the subjects together.
```

```
% init
conf_mat1(1:40) = 0; % correct
conf_mat2(1:40) = 0; % incorrect

% add id back.
if add_id == 0 || size(p,2)<= 50
   p(end+1,:) = p_temp;
end

for idx = 1:size(target,1)
   if target(idx,1) == .9 && Y(idx,1) == .9 % True positive
         conf_mat1(p(end,testInd(idx))) = conf_mat1(p(end,testInd(idx)))+1;
   elseif target(idx,1) == -.9 && Y(idx,1) == .9 % False positive
         conf_mat2(p(end,testInd(idx))) = conf_mat2(p(end,testInd(idx)))+1;
   elseif target(idx,1) == -.9 && Y(idx,1) == -.9 % True negative
         conf_mat1(p(end,testInd(idx))) = conf_mat1(p(end,testInd(idx)))+1;
   elseif target(idx,1) == .9 && Y(idx,1) == -.9 % False negative
         conf_mat2(p(end,testInd(idx))) = conf_mat2(p(end,testInd(idx)))+1;
   end
end

conf_mat(1:40) = 0;
for idx = 1:40
   n_test(idx) = (conf_mat1(idx)+conf_mat2(idx));
   conf_mat(idx) = conf_mat1(idx)/n_test(idx)*100;
end

% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% mode error individual trials

% incorrectly identified
temp = find(err==1);
for idx = 1:size(testInd)
   id_inc = testInd(temp);
end
end
```