International Journal of Robotics and Automation (IJRA)
Vol. 4, No. 4, December 2015, pp. 292~310
ISSN: 2089-4856

Using Deep Learning for Human Computer Interface via Electroencephalography

Vamsi Manchala, Sangram Redkar, Tom Sugar
The Polytechnic School, Arizona State University, USA

Article Info

Article history:

Received Jul 20, 2015
Revised Oct 27, 2015
Accepted Nov 20, 2016

Keyword:

Behavior Classification
Deep Learning
Electroencephalography (EEG)
Motor Imagery
Neutral Network

ABSTRACT

In this paper, several techniques used to perform EEG signal pre-processing, feature extraction and signal classification have been discussed, implemented, validated and verified; efficient supervised and unsupervised machine learning models, for the EEG motor imagery classification are identified.Brain Computer Interfaces are becoming the next generation controllers not only in the medical devices for disabled individuals but also in the gaming and entertainment industries. In order to build an effective Brain Computer Interface, it is important to have robust signal processing and machine learning modules which operate on the EEG signals and estimate the current thought or intent of the user. Motor Imagery (imaginary hand and leg movements) signals are acquired using the Emotiv EEG headset.The signal have been extracted and supplied to the machine learning (ML) stage, wherein, several ML techniques are applied and validated. The performances of various ML techniques are compared and some important observations are reported. Further, Deep Learning techniques like autoencoding have been used to perform unsupervised feature learning. The reliability of the features is presented and analyzed by performing classification by using the ML techniques. It is shown that hand engineered 'ad-hoc' feature extraction techniques are less reliable than the automated ('Deep Learning') feature learning techniques. All the findings in this research, can be used by the BCI research community for building motor imagery based BCI applications such as Gaming, Robot control and autonomous vehicles.

Corresponding Author:

Sangram Redkar,
Departement of Engineering,
The Polytechnic School,
Mesa, AZ-85212, USA.
Email: ssredkar@gmail.com

1. INTRODUCTION

Recent advancements and discoveries in the areas of brain imaging and cognitive neuroscience have enabled researchers to interact directly with the human brain. With the aid of these technologies and sophisticated sensors, researchers are able to observe and monitor the changing thought process in the form of low power electrical signals. These signals are used to make brain-computer interfaces (BCIs) possible and develop communication systems in which users explicitly manipulate their thought process, instead of motor movements, to control computers or communication devices.

Acquiring the brain signals, accurately, is the first step involved in BCIs. It is important to have a complete knowledge of the physiology and anatomy of human brain. This would be helpful in identifying the correct locations of the sensory nodes and measure the required signals.

Electroencephalography refers to the phenomenon of recording the electrical activity along the scalp and Electroencephalogram (EEG) is referred to the recorded signals and is the measure of voltage

fluctuations/variations occurred due to the flow of electrochemical currents in the neurons of the brain. During the signal recording procedure, electrodes consisting of small metal discs are pasted over the scalp. To maintain proper connectivity with the actual electrical signals, these electrodes are wetted by a conducting jell or liquid. However, the BCI world is now seeing some commercial dry EEG headsets, which would serve the purpose of capturing the data and transferring to the Computer through wireless medium. Patterns of the EEG signals, detected by the electrodes, represent that there is continuous activity present in the human brain and the varying intensities of the signal are determined by the changing mental and physical states of the body. These intensities of the EEG Signals recorded over the surface of the brain range from 0 microvolts to 200 microvolts.

The rhythmic activity of the brain signals is often divided to different bands in terms of frequency. Although these frequency bands are a matter of nomenclature, these designations are usually used to imply the fact that the rhythmic activity in a certain frequency range is observed due to certain biological significance and are often noted to have certain distribution over the scalp. Figure 1 shows the different frequency bands the EEG data is divided into, and Table 1 shows the significance of these frequency bands and related cognitive tasks these bands correspond to.

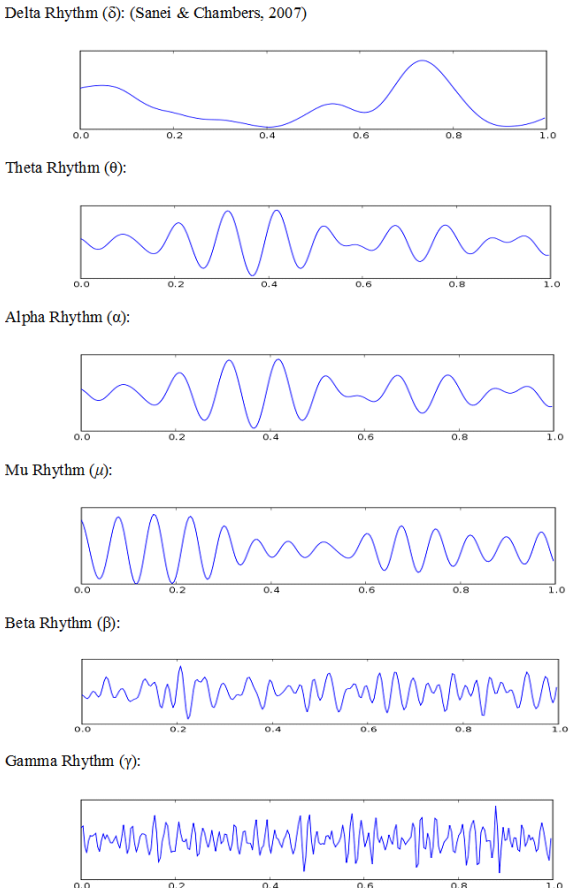

Figure 1. Frequency Plots of EEG in different Frequency Ranges. [1]

Table 1. Significance of EEG in different frequency bands.

Type	Frequency (Hz)	Location	Use
Delta	up to 4	Everywhere	occur during sleep, coma
Theta	4 – 7 Hz	temporal and parietal	correlated with emotional stress (frustration & disappointment)
Alpha	8 – 12 Hz	occipital and parietal	reduce amplitude with sensory stimulation or mental imagery
Beta	12 – 30 Hz	parietal and frontal	can increase amplitude during intense mental activity
Mu	9-11 Hz	frontal (motor cortex)	diminishes with movement or intention of movement

10-20 system is an internationally accepted and practiced scheme of electrode placement on the human scalp. The 10 and 20 in the name refer to the percentage distance of nodes from each other in proportion to the head size. The electrode locations suggested by this method belong to locations on cerebral cortex and the letters F, T, C, P and O denote the frontal, temporal, central, parietal and occipital respectively. Except for the central location the remaining are all lobes of the brain. The numbers indicate the position of the node on the scalp, even number denote right side of the head, odd number denote the left side and Z indicates that the node is located on the central line of the head. Figure 2 illustrates these standard electrode positions and Figure 3 illustrated the kind.

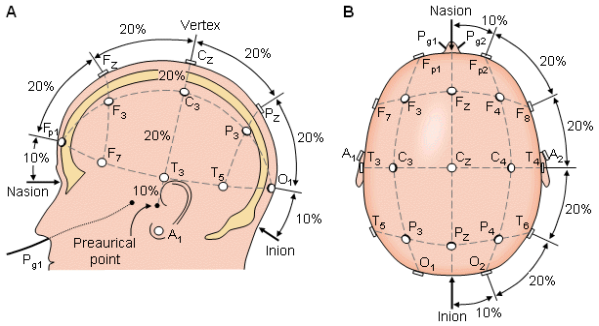

Figure 2. Standard electrode positions and placement on the human scalp. [2]

BCI is a branch of Human Computer Interface, which involves obtaining the brain signals, corresponding to specific form of thoughts, and translating them to machine commands. It is a communication system which performs the transfer of messages or commands by the means of human thoughts and not conventionally by peripheral nerves.

A highly anticipated application amongst the BCI communities is that the future user-communication systems would require a parallel feedback of the user mental state or intentions along with his physical state. For example, it is important for the automobile to react to the user's drowsiness. These future applications are called system-symbiosis or effective computing and require the systems to gather details regarding mental states like emotions, attention, workload, stress, fatigue, etc. and interpret them. [3]

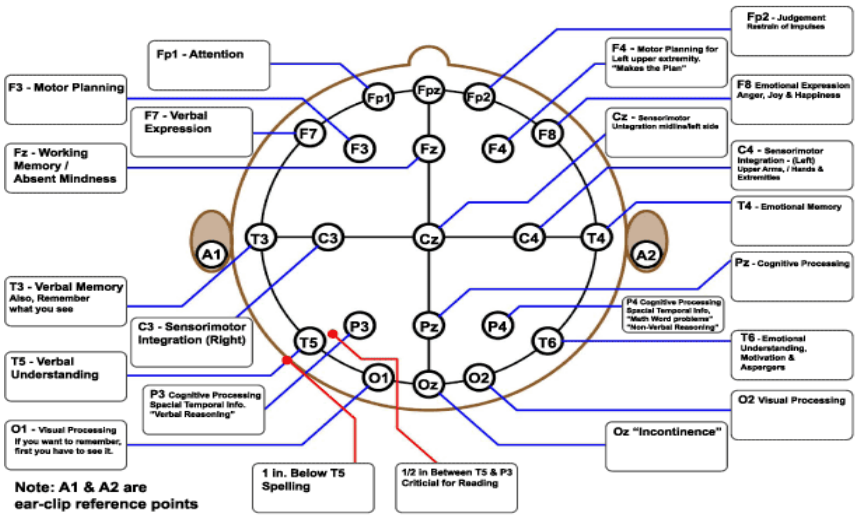

Figure 3. Showing the physiological signals expected from each node of the 10-20 system. [4]

Online and/or offline evaluation of applications using the physiological data might lead to several insights regarding the users state and help in comparing different use cases. For instance, a recent research on analyzing the brain imaging results of cell phone use during driving has proved that even hands free and

voice activated use of mobile phone is as dangerous as drunken driving. Another recent research in this EEG data evaluation has been conducted by Arizona State University, which focuses to find out how to leverage social media to improve educational and training environments. The goal of this research was to analyze the EEG data captured from students while they were using Facebook and try make a record of what they were looking at and also their affected state, and ultimately forward their findings to use in online learning communities and make online learning more interesting for the students. [5]

The gaming industry is earning most of its market share by making use of the wearable technology. Particularly, over the past few years, new game have been developed based on the commercially available EEG headsets by the companies like NeuroSky, Emotiv, Uncle Milton, Mattel and MindGames. The usual gaming experience has been enhanced and enriched by the use of BCIs in the gaming industry. For example, a typical BCI based game would no longer be controlled by the keyboard but would function based on the mental states, immersion, flow, surprise, and frustration etc. of the player. [3]

Brain Computer Interfaces are already being used in controlling many devices like motorized wheel chairs, prosthetic limbs, simulate muscular movement, controlling home appliances, lights, room temperature, television, operating doors, etc. The need for Brain Computer Interfaces in the embedded market is being explored, recent advances in BCI have seen projects using off the shelf EEG headsets and embedded single board computers like Beagle Bone Black and Raspberry Pi. [3]

2. RESEARCH METHOD

As discussed earlier, EEG is a recording of the bio-potentials from the surface of the scalp. More specifically, these recordings are the electrochemical potentials measured from the neurons at the cerebrum of the human brain. Since these signals are recorded from the surface of the scalp, it is most likely that potentials from many cells are being measured at the same time. At first glance, EEG data may look like an unstructured, non-stationary, noisy signal. However, advanced signal processing techniques can be used to separate different components of the brain waves. These separate components can then be associated with different brain areas and functions.

In order to carry on with the signal acquisition stage, it is important to identify whether the BCI signals are going to be dependent or independent; have evoked or spontaneous inputs. In addition to these, it is also important to decide on which method to use in obtaining the signals; a non -invasive or invasive. Ultimately, in the signal acquisition stage, the signals are obtained from the electrodes, amplified, digitized and made available for the further stages.

It is always possible that the acquired EEG data is combined with a lot of artifacts due to the electrical activity of eyes (EOG: Electroocculogram) or muscles (EMG: Electromyogram). The best way to avoid these unwanted components is to maintain ideal conditions during the signal acquisition, like maintaining a relaxed position which would involve minimum or no physical movements. However, on a practical note, maintaining such laboratory conditions in everyday BCIs is not realizable and such systems when used outdoors to operate embedded applications like UGV or wheelchair, is not considered to be robust and reliable. This problem can generally be solved by adopting effective pre-processing techniques which are responsible to clean the signal from unwanted artifacts and/or enhance the information embedded in these signals.

It is observed that the amplitude of these muscle artifacts is much higher than the usual EEG signals and during most offline analysis these can be removed by visual inspection. But to eliminate these artifacts in a more effective manner it is important to apply various spatio-spectro-temporal filtering techniques.

Feature extraction is phenomenon of building a feature vector of features which are considered as subset of data, derived from the main signals and, which best defines the signal of interest and reflects the similarities and differences between signals of same and different classes respectively.

Identifying and extracting relevant features is one of the most important steps in a BCI as it is proved to be crucial for an effective classification stage. If the features extracted from EEG are not relevant to the corresponding neurophysiological action, it would be very difficult for the BCI to classify the training signals into their respective classes and hence the system would not be performing effectively during the test phase. Thus, even if applying classification steps on the raw signals might give results, it would be a slow process and it is recommended to use an effective feature extraction technique in order to maximize the speed and efficiency of the BCI.

Often times, if a learning algorithm does not behave as desired it is most likely due to the high bias or high variance problem in the system. High bias is occurred due to under fitting of the algorithm. The bias error of the system is attributed to its inability to appropriately choose the function f, to estimate labels y of an input feature vector, from all the possible set of mapping functions. On the other hand, a high variance

problem is caused due to over fitting of the mapping function. This might reduce the performance of the system when provided with new testing data. [6]

Along with bias and variance problems, it is also important to understand the significance of using cross-validation in the selection procedure of a Machine Learning model, to validate the experimental results. Validation techniques are motivated by two fundamental and most important problems in Machine Learning: Model Selection and Performance Estimation.

a. Model Selection: Almost always, the performance of pattern recognition and the classification techniques depends on single/multiple parameters. For instance, enlisted below are some of the parameters used for model selection in different classification techniques [7].
b. Nonlinear Regression: Polynomials with different degrees.
c. K-Nearest Neighbors: Different choice of K.
d. Decision Trees: Different choices of number of levels.
e. SVM: Different choices of the misclassification penalty hyper parameter *C.*
f. Regularized Models: Different choices of the regularization parameter.
g. Kernel based Methods: Different choices of kernels.
h. Performance Estimation: Once the model is chosen it is important to estimate its performance, which is typically measured by evaluating the true error rate- the classifiers error rate on the entire data set. [7].
i. Machine Learning Classification k- Nearest Neighbor Classifier: k- Nearest Neighbor (k-NN) is simple and effective classifier. The classifier compares the test data with the training data. It evaluates the distances of each vector in the training data form the test vector, finds k nearest neighbors around the test sample and assigns the class label which is found amongst majority of the k nearest neighbors. The bias of the k-NN algorithm is very low since it is deciding based on the nearby points. However, it has a very high variance.

Some of the distance functions used in the k-NN algorithm are Eauclidean, Standardized Euclidean, City block, Chebychev, Cosine distance, Manhattan, Minkowski, Hamming, correlation distance, etc. Figure 4 shows region consisting of the test sample and its nearest neighbors.

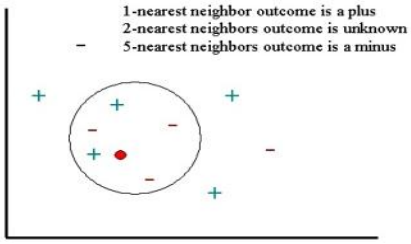

Figure 4. Showing the typical schema of K-NN [8]

j. Linear Discriminant Analysis: The working principle of LDA is to make use of a hyper-plane which separates the signals belonging to different classes. In a two-class problem, the two classes are separated by a hyper-plane and the signals belonging to different classes are on either sides of the hyper-plane. Similar to a two-class problem, different signals belonging to different classes in a multi-class problem are separated by multiple hyper-planes. [9]

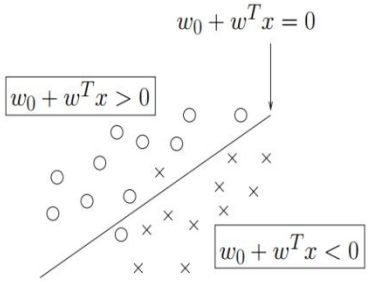

Figure 5. LDA hyper-plane [9]

LDA generally assumes a normal distribution of the data with same covariance matrices for all the signals. Each hyper-plane separating one class from the other classes is obtained by evaluation the projection that maximizes the distance between the mean of one class from the means of all other classes and minimizes the interclass covariance. Figure 5 shows the separating plane between two classes. The main advantages of this method is that it has a very low computational requirements and complexities, which makes it suitable for real time embedded applications. However the main drawback of this method is that it would not work effectively on non-linear complex EEG data.

k. Support Vector Machines (SVM): Like LDA, SVM is also used to classify signals into different classes and identify them when required, with the aid of a hyper-plane. However, SVM tries to solve the problem of non-linear complex signals. In SVM, the selection of the hyper-plane is made to maximize the width of the band which separates the nearest training points to increase the generalization capabilities. [9,10]

The hyper-plane, also called as decision border, segments the feature space into parts equal to the number of classes of the signals. The result of the classification stage would depends on which part of the plane is the test signal located. Figure 6 shows the optimal hyper-plane separating two planes in SVM.

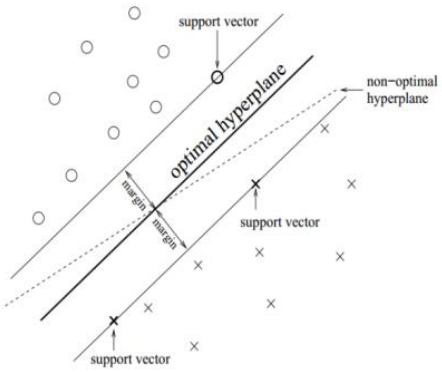

Figure 6. Hyper-plane and support vectors [11]

Depending upon whether or not the time series signals is linearly separable, the SVM method would be able to convert the data into linearly separable and create nonlinear decision boundaries to classify them (Figure 6). This phenomenon of building non-linear decision boundaries is not much complex as is making the use of a kernel trick to implicitly map the data to another space of higher dimensionality, where the data is linearly separable and the regular linear classifiers are still applicable. The kernel generally used in BCI research is the Gaussian kernel:

$$K(x,y) = exp\left(\frac{-x-y^2}{2\sigma^2}\right) \tag{4}$$

l. Naïve Bayes classifier: Naïve Bayes probability function is as follows-

$$p(c_l \mid x_1, x_2, \ldots, x_m) = \frac{p(c_l)\prod_{i=1}^{m} p(x_i \mid c_l)}{\sum_{q=1}^{N}\left[p(c_q)\prod_{i=1}^{m} p(x_i \mid c_q)\right]} \tag{5}$$

Where N is the total number of classes. The individual probabilities on the right-hand side of the equation are evaluated from the training data [10].

m. Logistic regression used for Classification: Unlike in the regression problem, the output values y of the model take a limited number of discrete values in the classification problem. For example in a binary classification the output y might either take a value of 1 or 0 depending on whether or not the input feature vector belongs to the desired class? [12] For logistic regression used for classification, a

sigmoid function is used as a hypothesis to predict the output class as the output of a sigmoid would range between 0 and 1. Vectors which produce output lower than 0.5 would be assigned a 0 class and the ones with an output value more than 0.5 would be assigned a 1, as shown in Figure 7. [12]

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \tag{6}$$

$$g(z) = 1/(1+e^{(-z)}) \tag{7}$$

Here, $g(z)$ is called the logistic function or the sigmoid function and $\theta_i's$ are the parameters (also called as weights) parameterizing the space of logistic function mapping *X* and *Y*.

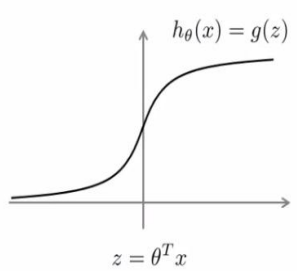

Figure 7. Showing the logistic function. [12]

The main focus of the logistic regression classifier is to evaluate the values of the weights θ_i, in an iterative fashion, so as to reduce the difference between the hypothesis of an input feature vector and the corresponding output. This is achieved by computing the cost function $J(\theta)$ for every set of weights and comparing it with that obtained from earlier sets of theta.[12]

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta\left(x^{(i)}\right) + (1-y^{(i)})\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right)^2 \tag{8}$$

The efficiency of the Logistic Regression function in classifying the correct class depends on the selection of the data fitting function. The function might either under-fit or over-fit the data (Figure 8).

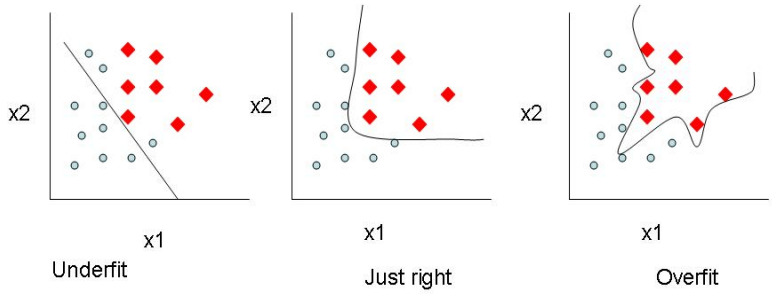

Figure 8. Showing different kinds of functions used to fit the data [13]

n. Artificial Neural Networks: ANN is an assembly of several artificial neurons which have capability to produce non-linear decision boundaries and when combined with classifiers are capable of solving the multi class problem. A typical ANN is composed of several layers of neurons: an input layer, one or several hidden layers and an output layer, the number of neurons in which are based on the number of classes in the problem. [10]

Neural Networks behave as universal approximates when built of enough neurons and layers as they can approximate any continuous function. Another advantage that make the Neural Networks flexible for a great number of problems is that they can classify any number of classes.

The intuition for Neural Networks can be built over the understanding developed on Logistic Regression, in the previous section. Consider a supervised learning problem, provided with labeled training data $(x(i), y(i))$. Neural Networks give a way of defining a complex, non-linear form of hypotheses $h_{W,b}(x)$, with parameters W, b that one can fit to our data.

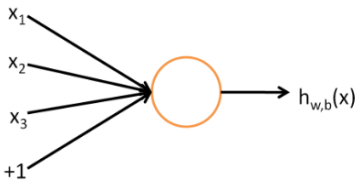

Figure 9. Single Neuron used in a NN [14]

A single neuron (Figure 9) is a basic computational unit in a complex NN, takes inputs $x1, x2, x3$..... and outputs $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$, where $f : R \rightarrow R$ is called the activation function. Most cases it is either a sigmoid function or a tanh function.

$$f(z) = 1/(1+e^{(-z)}) \tag{9}$$

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{10}$$

A Neural Network is built by connecting multiple simple neurons together to form a complex network. [14]. For example, Figure 10 shows a NN which is built with one input layer, one hidden layer and an output layer, capable of classifying two different actions.

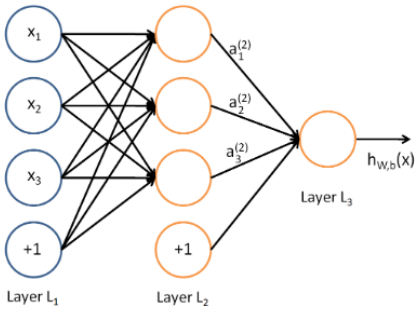

Figure 10. Typical NN. [14]

The computation for neural network in Figure 11 is given by (using the notations as per Table 2):

$$\begin{aligned} a_1^{(2)} &= f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}) \\ a_2^{(2)} &= f(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}) \\ a_1^{(2)} &= f(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}) \\ h_{w,b}(x) = a_1^{(3)} &= f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)}) \end{aligned} \tag{11}$$

For fixed training set $\{(x^{(1)}, y^{(1)}),.....(x^{(m)}, y^{(m)})\}$ of m training examples. For a single training example (x, y), the cost function is defined as-

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 \tag{12}$$

And for a training set of m samples, the overall cost function would be-

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2 \end{aligned} \tag{13}$$

The first term in the above equation is an average sum-of-squares error term. The second term is a regularization term (also called a weight decay term) that tends to decrease the magnitude of the weights, and helps prevent over fitting. [14] The ultimate goal in the Neural Networks is to come up with the best set of parameters $(W,\ b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, which would minimize the $J(W,b)$. To train the network, we will initialize each parameter $W_{ij}^{(l)}$ and each $b_i^{(l)}$ to random non-zero values, and update the $W_{ij}^{(l)}$ and $b_i^{(l)}$ for every iteration by applying techniques like gradient descent. One iteration of gradient decent updated the parameters as follows:

$$\begin{aligned} W_{ij}^{(l)} &:= W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b) \\ b_i^{(l)} &:= b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b) \end{aligned} \tag{14}$$

Table 2. Notations used in Neural Networks

$(x^{(i)}, y^{(i)})$	$i-th$ Training example
$h_{w,b}(x)$	Output of hypothesis on input x, using parameters W, b.
$W_{ij}^{(l)}$	The parameter associated with the connection between unit j in layer l, and unit i in layer $l+1$.
$b_i^{(l)}$	The bias term associated with unit i in layer $l+1$.
$a_i^{(l)}$	Activation of unit i in layer l of the network.

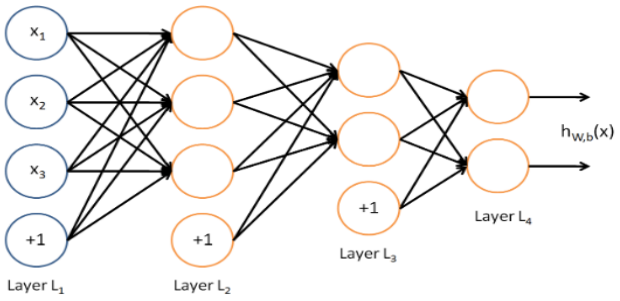

Figure 11. Neural Network showing input, hidden and output layers for multi-class classification [14]

o. Deep Learning: Many of the features are discovered by observation of raw data by many researchers, over several years. The area of feature extraction is considered to be almost saturated and the researchers are seeing themselves getting interested to explore more sophisticated and automated feature extraction techniques. Deep Learning is one area currently being explored by the Machine Learning research communities to emulate the feature learning and classification mechanism taking place in the human brain to understand the information it gets from different natural sensors, by breaking down the complex information into new and simple representations. One potential use of Deep Learning is unsupervised feature learning, which tries to understand the complex data and represent it in much less complexity.

Deep Learning refers to a rather wide class of machine learning techniques and architecture. Based on how the architecture has been designed and its intended use, Deep Learning techniques can be classified into three major areas.

p. Deep networks for unsupervised or generative learning: Used to capture high-order correlation of the data to analyze patterns and synthesize them when no information about the target class is available.
q. Deep networks for supervised learning: Target label data are made available for such kinds of techniques to directly provide discriminative information for pattern classification purposes.
r. Hybrid deep networks: It is a blend of both Supervised and Unsupervised techniques to produce higher classification rates. Herein, the network works with unsupervised and largely generative pre-training to boost the effectiveness of supervised training. This is procedure is found critical when the training data are limited.
s. Training a Deep Model: Deep Models are trained in a greedy layer-wise unsupervised manner. This greedy layer-wise unsupervised learning algorithm first starts with the training of the first layer of the model in an unsupervised fashion to yield an initial set of parameters for the first layer of the network [15]. The output form the first layer is a reduced representation of the input and is supplied as an input to the second layer which is similarly trained using the same unsupervised algorithm, to yield the initial parameters of that layer. Again, the output form the second layer is used as an input to train the third and this process continues until all the parameters of each layer have an initial values which are reduced representations of the previous layer [15].

Following this unsupervised pre-training phase, of obtaining the initial parameters of the stacked neural network, the complete network can then be fine-tuned by applying supervised backpropagation in the reverse direction. Backpropagation is responsible to readjust the weights in an iterative fashion by trying to reduce the error (cost function) between the true labels and the labels obtained from the network, during each iteration. As the weights are adjusted to obtain the closest output labels, the internal hidden units become the best representations of the input features [15].

t. Autoencoder: Autoencoders offer a method of automatically learning features from unlabeled data, allowing for unsupervised learning. It performs backpropagation without any knowledge of the labels [16]. An autoencoder is an artificial neural network that is able to be trained in a completely unsupervised manner. In the usual neural networks, labeled data were required to train the network using the back propagation phase by fine-tuning the initially assigned weights. Whereas, the autoencoders provide the ability to learn the information without the need for labeled data. An autoencoder neural network performs backpropagation by setting the target values to the input values. In other words, an autoencoder neural network (shown in Figure 12), an unsupervised feature learning algorithm that trains the $h_{w,b}(x)$ setting the target values to be equal to the inputs i.e. it uses $y^{(i)} = x^{(i)}$.

This structure has been proved to be used effectively in different kinds of applications, one being the solution to the dimensionality problem of the EEG data, wherein the intermediary activation values of the hidden layer can be passed as features (with reduced dimensionality) to a supervised learning algorithm. For example, consider an EEG motor imagery data set consisting of single trials with data spanned over 5 seconds with 128Hz frequency. The total number of features in a single trial are $128*5 = 640$ which is huge and computationally intense for a normal classification technique like LDA, SVM. However, one can use a hidden layer of 200 nodes to construct an autoencoder and the activations $a_i^{(l)}$ for each training sample are unique and is totally based on the weights of the network obtained by training it using all the training samples. By limiting the number of hidden units, and performing the training, autoencoder will result in a compressed representation of the data.

The above discussion of being able to come up with a new representation of the input features, with reduced dimensionality is realizable only if the hidden layer has a lower number of nodes. But even when the number of hidden units is large, may be greater than the number of input, one can still come up with interesting features by imposing other constraints on the network [14]. One way to achieve this is to impose sparsity constraint on the hidden units. *"Sparsity is a very useful property of some Machine Learning*

algorithms. Such an algorithm yields a sparse result when, among all the coefficients that describe the model, only a small number are non-zero." [17]

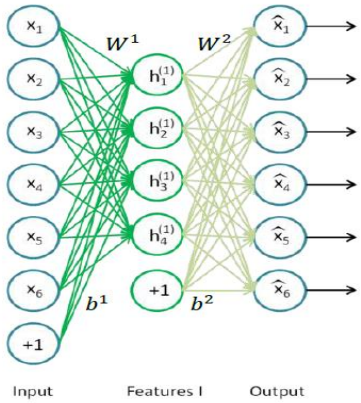

Figure 12. Sparse Autoencoder. [18]

These concepts can be further built upon to develop deep architectures to solve the multi-class classification problems. Several autoencoder layers can be stacked together to form a deep learning network called as a stacked autoencoder network. A stacked autoencoder is a neural network consisting of multiple layers of sparse autoencoders in which the unsupervised pre-training is performed on one layer at a time and the outputs of each layer is fed into the inputs of the successive layer. It follows the 'greedy layer-wise' learning algorithm to effectively pre-train the neural network. This approach is particularly useful when the network is composed of several layers wherein it would be difficult to attain the global minima of the cost function, as large initial weights might cause autoencoders to find poor local minima and small initial weights would make it infeasible to train many-layered autoencoders [15]. In the case of a stacked autoencoder, the weights are initialized to good solutions before starting the supervised learning and adjusting the weights by back propagating over the network. For example, consider the stacked autoencoder architecture. The ultimate goal here would be to train this neural network by adopting the deep feature learning techniques. To achieve this, first consider the single (first) layer of autoencoder, in Figure 12, it consists of an input, output and hidden layer, the W^1 matrix is composed of the weighted connections between the input data and the hidden units, while W^2 contains the weighted connections between the hidden units and the output. Similarly, b^1 represents the biases from the bias unit in the input layer to each hidden unit, while b^2 represents the bias from the bias unit in the hidden layer to the output layer. That is, each single layer module has a set of parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ representing the weights and biases connecting the network.

Now, this sparse autoencoder module will be trained using all the input vectors to obtain the suitable parameter set $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, by using the backpropagation/gradient descent techniques to lower the overall cost function of this particular layer, over multiple iterations. Each time a new input feature vector $x1, x2, x3$..... is supplied to the autoencoder and performed backpropagation, the cost function of the model is expected to attain global minima.

Equation (15) shows the overall cost function which takes into consideration, all the sparsity constraints. KL-divergence is a standard function for measuring how different two different distributions are. $KL(\rho \| \hat{\rho}_j) = 0$ if $\hat{\rho}_j = \rho$ and increases monotonically as $\hat{\rho}_j$ diverges from ρ.

Note that the output units of the single autoencoder will not be present in the final stacked autoencoder. They are simply used to train the single layer to obtain the initial parameters. Rather, the activation values, which represent unique features of and obtained for each input vector are used as inputs to the second layer would be present in the final stacked autoencoder. That is, the hidden units of the first autoencoders can be considered to be the next visible inputs of the next autoencoder. These inputs to the next autoencoder are obtained by performing feed forward propagation, using every single input vector, over the initial autoencoder once it is completely trained.As expected, the output units of the second autoencoder in the stack are a representation of the hidden units of the first autoencoder. This process would repeat until the

final output layer of the stacked autoencoder is reached, wherein the softmax classifier should be trained. The softmax classifier is trained in a similar fashion by providing the activations of the final hidden layer as inputs to an autoencoder and trying to fit a model with available inputs and outputs.

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{n} KL(\rho \,||\, \hat{\rho}_j) \qquad (15)$$

where the notations used in equation (15) are given in Table 3

Table 3. Notations used in equation (15)

$J_{sparse}(W,b)$	Overall cost function with sparsity constraint.
$J(W,b)$	Cost function of a NN, shown in equation (15)
$\hat{\rho}_j$	Average activation of hidden unit j with n nodes.
ρ	Sparsity Parameter
$KL(\rho \,\|\| \hat{\rho}_j)$	Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$
β	Controls the weight of the sparsity penalty term.

After having trained each layer of the network on the unlabeled data, the parameters are now starting with a comparatively better values as compared to initializing them randomly, thus accounting for a fundamental flaw in previous neural networks. Now, the stacked autoencoder is finally ready to be combined and fine-tuned to improve the performance. While layer-wise pre-training is used for finding the features of the network, fine-tuning is used to slightly modify the features of the network in order to adjust the boundaries between the classification classes. Fine-tuning is performed by treating the entire network as a single model and applying forward propagation and backward propagation iteratively for every input vector available. A single iteration of fine-tuning improves all the weights of the stacked autoencoder, at every level as shown in Figure 13.

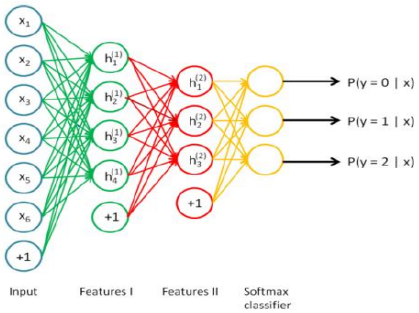

Figure 13. Final network of the stacked autoencoder [18]

3. RESULTS AND ANALYSIS

This section will discuss the techniques adopted to acquire the data, experiments/analysis performed to understand the data in hand and the results obtained. Besides using the Classification/Machine Learning techniques implemented as part of this paper, some of the open source EEG/BCI toolboxes have been used to gather the results. Irrespective of the Machine Learning technique/toolbox used, it is important to verify and validate the reliability/efficiency with which the techniques would perform the classification. To achieve this task, the initial study in this paper has been started off by using the standard EEG data available online.

For data collection, BCI2000 (http://www.schalklab.org/research/bci2000) was used. BCI2000 comes with a multi-purpose stimulus presentation program, shown in Figure 14. Its main aim is to ease the process of data acquisition from any headset (that BCI2000 supports through community contributions) in real-time and integrate the with feed-back applications if required. This Stimulus Presentation program

(shown in Figure 15) has been used in this paper to generate cues and simultaneously record the EEG Data along with the event markers corresponding to the cues, in Real Time. To have an effective database we need to collect data by performing a decent amount of single trials. In this paper data has been collected from 2 healthy subjects, using Emotiv EEG Headset. Each subject was prompted, by the BCI2000 Stimulation Presentation application in the form of cues, to perform either left motor imagery, right motor imagery task or remain at rest. 60 trails for each of the three tasks (left, right, rest) have been collected.

Before data collection, Emotiv Headset was connected to the PC via the USB dongle. The electrodes on the headset have been soaked in multipurpose saline solution to enhance the contact between the sensor and the scalp. Each test subject was helped to put on the headset in such a way that the F3 and F4 electrodes are right above the motor cortex area rather than the frontal area. This arrangement is must, as the Emotiv headset does not contain electrodes which would go directly on top of the motor cortex when placed normally. The difference between the positioning of the Emotiv headset normally and that suitable to capture motor imagery data has been shown in Figures 16 and 17.

Figure 14. Stimulus presentation module in BCI2000

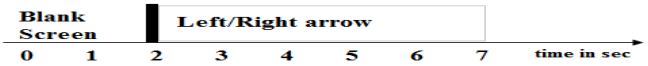

Figure 15. Paradigm created using Stimulus Presentation to capture data from Emotiv

Figure 16. Showing usual electrode placement using Emotiv Headset

Figure 17. Showing the electrode placement used in this paper to acquire MI data

Suitable settings are made in the parameter files of the BCI2000 Stimulus Presentation application to generate cues for left (left arrow), right (right arrow) and rest (blank screen) for 5s each, in random order, with blank screen in between/after few trails encouraging the test subject to relax his/her muscle, blink eyes, etc.

During the data collection, when the left or right arrow is displayed, the subject was asked to imagine that he or she is continuously opening or closing the respective hand (e.g., squeezing a tennis ball) at a rate of about one opening and closing per second, and remain in a resting state when a blank screen was seen [19]

3.1. Cross Validation and Classification

The main aim of this section is to reiterate the importance of selecting a suitable machine Learning model for the data being analyzed. As discussed previously, it is important to validate the techniques based on the principles like Model Selection and Performance estimation, to come up with the best Classification technique with best set of parameters. For the available data, it is ideal to use k-fold cross validation method as the number of samples available are very less and it would basically involve all the data trails for both training and testing. The validation error obtained might not be accurate enough, if other methods are used.

Model selection can be a crucial step when the machine learning technique involves several parameters as discussed before, for example in K Nearest Neighbors technique, it is important to select the value of 'K' and also the method used to evaluate the distance between different data points. It is advised to select optimized model by varying one feature at a time. However, obtained accuracy in classification using different models might vary from data set to data set and person to person. The following results are obtained by using the MATLAB. K-fold cross validation methods are used in order to make every feature vector participate in the training and testing phase.

a. K Nearest Neighbors**:** It is evident from the above results obtained by applying K-Nearest Neighbors algorithm on the data set under consideration, extracting different features with varying 'K' value that a 'K' value of 5 would on an average fetch a well-trained classifier for the kind of signals used. Now, it would be ideal to set the value of 'K' equal to 5 and observe the performance by varying different features, here, the formula used to evaluate the distance between the data points. Again, it is important to note that the value of k or for that matter the value of parameters being considered in model selection vary person to person. Careful analysis is required for the signals from each subject and it is ideal to select separate models for different subjects (**S1**- Subject 1, **S2**-Subject 2).

Table 4. Accuracies with KNN Method using different K values on data from Emotiv (S1- Subject 1, S2-Subject 2)

		K=1	K=3	K=5	K=7	K=9
S1	Band power	46.6667	62.8571	66.1905	59.0476	62.3810
	Average	67.6190	63.0952	65.2381	52.8571	59.5238
	RMS	35.2381	40.0000	49.0476	55.7143	50.0000
S2	Band power	65.7143	64.7619	69.5238	60.0000	59.0476
	Average	59.5238	38.0952	34.2857	40.4762	42.8571
	RMS	54.2857	56.1905	62.3810	62.8571	56.6667

MATLAB inbuilt implementation of KNN is available with different types of distance measurements like, Euclidean, cityblock, cosine, correlation, hamming, etc. It is shown in Table 5.

Table 5. Accuracies for a fixed K value and changing distance formula on Emotiv data (S1- Subject 1, S2-Subject 2)

	K=5	euclidean	seuclidean	Chebychev	Mahalanobis	Hamming
S1	Band power	66.1905	67.6190	67.6190	59.5238	43.8095
	Average	65.2381	59.5238	60.0000	62.3810	27.6190
	RMS	43.8095	39.538	49.5238	47.1429	52.3810
S2	Band power	69.5238	63.3333	65.2381	63.3333	22.3810
	Average	34.7619	37.6190	31.4286	50.0000	50.9524
	RMS	59.5238	59.0476	62.8571	61.9048	59.0476

b. Support Vector Machines (SVM):The results presented in Table 6 were obtained by applying support vector machines with different kernel functions. The different kernels tried in this section were linear, quadratic, polynomial, Gaussian Radial Basis Function(Rbf), Multilayer perception (Mlp) kernel.

Table 6. SVM Classification results for different kernel functions on Emotiv data (S1- Subject 1,S2-Subject 2)

		linear	quadratic	polynomial	Rbf	Mlp
	Band power	69.5238	69.0476	59.0476	72.3810	53.8095
S1	Average	71.4286	66.1905	63.8095	60.0000	58.5714
	RMS	50.0000	59.5238	57.1429	63.3333	59.0476
S2	Band power	71.4286	60.0000	65.2381	68.3810	65.7143
	Average	65.2381	53.3333	55.7143	50.0000	55.7143
	RMS	50.0000	66.6667	67.1429	70.9524	43.8095

c. Linear Discriminant Analysis: The results presented in Table 7 were obtained by applying linear discriminant analysis with different types of discriminant functions. The different discriminant functions tried in this section are 'linear', 'pseudolinear', 'diaglinear', 'quadratic', 'pseudoquadratic'.

Table 7. LDA Classification results for different kernel functions on data from Emotiv (S1- Subject 1, S2-Subject 2).

		Linear	pseudoLinear	diagLinear	Quadra	PseudoQuadra
	Band power	69.5238	69.5238	60.0000	69.0476	69.0476
S1	Average	68.5714	68.5714	47.1429	65.2381	65.2381
	RMS	37.1429	37.1429	37.1429	37.1429	39.0476
	Band power	74.7619	74.7619	71.4286	65.7143	65.7143
S2	Average	47.1429	47.1429	47.1429	50.4762	50.4762
	RMS	33.8095	33.8095	33.8095	47.6190	47.6190

It is observed from Tables 4-7, that a particular classification model cannot be generalized to be working effectively for different datasets, acquired from different test subjects under varying experimental conditions at different times. Also, it is seen that the performance of some models is extremely low in the case of certain subject, this may be due to either the machine learning problems like over-fitting and under-fitting (algorithm might not be able to fit the data available into model effectively) or may be due to the availability of limited number of samples for training.

3.2. Unsupervised Feature Learning and Deep Learning

a. Neural Networks: Multilayer neural networks can be used to perform feature learning as they learn a representation of the input at the hidden layers, which is used for subsequent classification or regression at the output layer. As discussed earlier, a typical neural network trains the model in a supervised fashion by updating the model parameters, iteratively, each time it is provided with a training sample. On every update, the error (cost function) in classification is expected to be reduced.

The neural network shown in Figure 18 is a simple one input layer, one hidden layer and one output layer neural network that was used for classification. However, to validate the performance the number of neurons in the hidden layer have been varied.

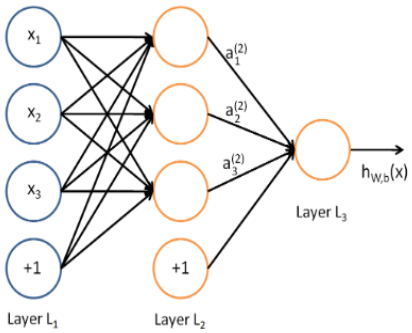

Figure 18. Structure of the neural network used [14]

Table 8. Classification results of NN with different hidden neurons, on Emotiv data

Hidden units =	550	650	750
Subject 1	65.63	58.00	55.75
Subject 2	62.50	68.75	59.38

It is generally misunderstood in the case of neural networks that the performance might increase with increase in the number of hidden units, as the number of features being made available to the next layer in the network is increased, meaning amount of information given to learn is increased. However, this is not always true, one specific case is when the features are over-fitting the model. That is, the network has learned a model which fits most of the data points from all the input vectors but it might not be able to perform the same on a test input vector. It can be observed from the above results that classification accuracies are decreasing with increasing number of hidden neurons. There are very high chances that the models with increased hidden units are suffering from over-fitting problem as the number of input vectors for training are limited.

b. Autoencoder + Neural Network: It has been pointed out in the previous section that the initial weights of the neural network are randomly selected. *However, random selection of the weights might cause more problems than it solves, particularly in the case of a neural network which contains multiple hidden layers.* It is difficult to optimize the weights of a neural network with multiple hidden layers. As discussed earlier, with large initial weights, networks typically find poor local minima and with small initial weights result in tiny gradients in the early layers, that makes training the remaining layers almost impossible. If the initial weights are close to the good solution, gradient descent works well [15].The Autoencoder + Neural Network approach used the technique of pre-training the network to come up with a good set of initial weights and set them as the initial weights of the regular neural network.

Table 9 Results of SA+NN with different hidden neurons, on Emotiv data

Hidden units =	550	650	750
Subject 1	84.38	46.88	68.75
Subject 2	71.88	65.63	65.63

It is evident from the results tabulated in Tables 8 and 9 that the performance of a neural network with SA pre-training is better than the performance of that with just neural network random initialization of the weights.

c. Using the Learned features (Unsupervised Learning):This section tabulates the classification accuracies of different machine learning techniques like k nearest neighbors, support vector machines and discriminant analysis, similar to the previous section where in the performance of ML techniques is compared for different features and different parameters. Except, the features being considered are the ones learned from the autoencoder.

As discussed earlier, feature learning is currently being used as an alternative to the regular hand engineering the features and extracting them from raw data. To achieve this, in this research, the activation values of the hidden layer of the autoencoder used in previous section for pre-training are supplied as input features to the regular machine learning techniques.

In order to construct an autoencoder, consider a neural network of three layers; one input layer, one hidden layer and one output layer. An autoencoder is used to represent the complex input vector as less complex vector which is a weighted summation of the provided input vector. After the autoencoder has been trained with all the available training samples, the less complex representations of the input vectors are obtained from the corresponding activation values of the hidden unit. It trains an identity function $y^{(i)} = x^{(i)}$ for each input vector supplied to the network. The number of units in the hidden layer are user dependent and can be chosen based on how compressed you want to represent the original data. In this paper, the number of hidden units are kept equal to the number of input features divided by 2. The reason for this being the good results obtained with the same setup in the previous section wherein autoencoders are used to obtain initial weights of a neural networks.

The results obtained using the learned features (shown in Tables 10-13) are better in most cases than the ones obtained previously (as shown in Tables 4-7).

Table 10. Classifying the features learned from autoencoder using KNN(Emotiv data)

	K=1	K=3	K=5	K=7	K=9
S1	67.1429	72.8571	66.6667	60.9524	53.8095
S2	60.9524	72.3810	66.1905	62.8571	53.8095

Table 11. Classifying the features learned from autoencoder using KNN(Emotiv data)

K=3	Euclidean	seuclidean	chebychev	Mahalanobis	Hamming
S1	72.8571	72.8571	72.8571	72.8571	59.5238
S2	72.3810	72.3810	72.3810	72.3810	50.4762

Table 12. Classifying the features learned from autoencoder using SVM(Emotiv data)

	Linear	quadratic	polynomial	Rbf	Mlp
S1	64.2857	73.3333	73.8095	67.1429	60.4762
S2	65.2381	59.5238	59.5238	65.2381	65.2381

Table 13. Classifying the features learned from autoencoder using LDA(Emotiv data)

	Linear	pseudoLinear	diagLinear	Quadra	PseudoQuadra
S1	60.4762	60.4762	60.4762	70.4762	70.4762
S2	55.2381	55.2381	55.2381	62.3810	62.3810

Tables 10 through 13 depict the classification accuracies of the KNN, SVM and LDA with the features learned by using the unsupervised feature learning technique. As it can be observed from Tables 10-13 and Tables 4 to 7, the classification accuracies obtained from learned features are better than those obtained by using the extracted features.

4. CONCLUSION

In the course of this research, authors have focused on developing a systematic and step by step approach to select a good machine learning model to solve the classification problem of EEG signals used in Brain Computer Interface applications. The different stages of a typical BCI, feature extraction and machine learning, have been discussed and the need for validation & verification, in every stage has been pointed out. To further strengthen the assertion, several feature extraction techniques and machine leaning techniques have been investigated and used on standard datasets available online and the data acquired at Arizona State University as part of this. Several observations are made from the results obtained thereby.

It is known that the feature extraction stage is one of the important stages of a BCI application. However, it has been pointed out in this paper, that it is not effective/efficient to pick a single kind of feature and use it for every BCI problem. Because, it is not always possible to tell which features are the best amongst the known set of features for a given biological signal, as they might all not be equally informative, might lose some significant information which was otherwise present in the raw data, some of them might be noisy,correlated or irrelevant.

Machine Learning stage is another important entity of a BCI system, wherein the computer would be given the information of how the brain signals of a particular cognitive task might look, so it would be able to recognize the same in the future. But it is not as simple as it sounds, the most critical aspect here is how effectively is the computer able to understand and learn the information provided. Machine learning models are mathematical representations of the signal data. For a particular machine learning technique, like K Nearest Neighbors, Neural Networks and etc., the model might vary for different parameters of the algorithm. In this paper, it has been pointed out, that a particular classification model cannot be generalized to be working effectively for different datasets, acquired from different test subjects under varying experimental conditions at different times. Several problems, like over-fitting and under-fitting, which might arise for a particular machine learning model to fit the training data has been investigated and the same has been proved from the results obtained by classifying the motor imagery data using several machine learning models.

Alongside, it has been asserted and proved that hand engineered feature extraction techniques are less reliable than the automated feature learning techniques. Feature representations for the complex time

series data has been obtained by using Deep Learning techniques like autoencoders, in an unsupervised fashion. These features are further fed to the machine learning models investigated in this paper.

A particular case of neural networks, which involved random initialization of model parameters, has been further investigated and assertion was made that the performance would increase if the model parameters were initialized intelligently to learned values. Initial model parameters of neural networks have been obtained by performing pre-training using stacked autoencoders. The results prove the assertion. Several applications and extensions of this work are in progress.

REFERENCES

[1] M. Ortiz, July 2012. [Online]. Available: http://cec.sonus.ca/econtact/14_2/ortiz_biofeedback.html.

[2] "EEG_Measurement_Setup," [Online]. Available:http://www.bci2000.org/wiki/index.php/User_Tutorial:EEG_Measurement_Setup.

[3] J. B. V. Erp, F. Lotte and M. Tangermann, "Brain-Computer Interfaces: Beyond Medical Applications," *Computer - IEEE Computer Society-, IEEE*, pp. 26-34, 2012.

[4] John, "mapping-10-20-system-to-brain-functioning," 6 October 2014. [Online]. Available: http://www.diytdcs.com/2014/10/mapping-10-20-system-to-brain-functioning/.

[5] "ANGLE's Facebook project," 16 May 2013. [Online]. Available: http://angle.lab.asu.edu/site/?p=1515.

[6] S. Kakade and D. McAllester, "Statistical Decision Theory, Least Squares, and Bias Variance Tradeoff," 17 October 2006. [Online]. Available: http://ttic.uchicago.edu/~dmcallester/ttic101-06/lectures/biasvar/bias_var.pdf.

[7] P. Rai, "Model Selection and Feature Selection," 22 September 2011. [Online]. Available: http://www.cs.utah.edu/~piyush/teaching/22-9-print.pdf.

[8] O. Tulsa, "Electronic Statistics Textbook," 2013. [Online]. Available: http://www.statsoft.com/textbook/.

[9] F. Lotte, "Study of Electroencephalographic Signal Processing and Classification Techniques towards the use of Brain-Computer Interfaces in Virtual Reality Applications. Human Computer Interactions," INSA de Rennes, 2008.

[10] I. V. Gerla, "Automated Analysis of Long-Term EEG Signals," 2012.

[11] F. LOTTE, "Study of Electroencephalographic Signal Processing and Classification Techniques towards the use of Brain-Computer Interfaces in Virtual Reality Applications," 2008.

[12] A. Ng, "Machine Learning," 22 April 2013. [Online]. Available: http://online.stanford.edu/course/machine-learning.

[13] T. V. Ganesh, "simplifying-machine-learning-bias-variance-regularization-and-odd-facts-part-4," 3 January 2014. [Online]. Available: https://gigadom.wordpress.com/2014/01/03/simplifying-machine-learning-bias-variance-regularization-and-odd-facts-part-4/.

[14] A. Ng, "CS294A Lecture notes," 2011. [Online]. Available: https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf.

[15] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with Neural Networks," www.sciencemag.org, 2006.

[16] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition.," *Biological cybernetics*, pp. 59(4-5), 291-294, 1988.

[17] O. Bousquet, 08 November 2005. [Online]. Available: http://ml.typepad.com/machine_learning_thoughts/2005/11/when_does_spars.html.

[18] C. Gravelines, "Deep Learning via Stacked Sparse Autoencoders for Automated Voxel-Wise Brain Parcellation Based on Functional Connectivity," 2014.

[19] G. Schalk and J. Mellinger, A Practical Guide to Brain–Computer Interfacing with BCI2000, London: ©Springer-Verlag London Limited, 2010.

BIOGRAPHIES OF AUTHORS

Vamsi Krishna Manchala graduated from Arizona State University with Master of Science (EGR). His research interests are in human-machine interfaces, computational modeling and artificial intelligence.

Sangram Redkar is an associate professor at the Polytechnic School at Arizona State University. His research interests are in nonlinear dynamics and controls, human-machine interface and robotics.

Tom Sugar is a professor at the Polytechnic School at Arizona State University. His research interests are in wearable robotics and human-machine interface