

1.ABSTRACT

We know that our brain works with different signals. These signals are generated with every action we take, even in our sleep. We basically capture these signals by placing electrodes on our skulls and then recording it as a wave pattern on the EEG. Normal activity has a normal pattern but an abnormal EEG has some features. After some examination the doctor can detect an abnormal EEG from normal people.

Although the EEG helps us to distinguish between normal and abnormal, it is up to the examiner to make the final decision. To understand the signals extracted by the EEG, we need the help of experts in this field. Our main goal is to reduce the burden on the examiner and reduce test time.

The ultimate goal of this project is to automate the entire process of testing to classify the EEG. We now focus only on classifying the EEG as appropriate. This automation helps to reduce the total time required to detect and test the EEG.

2.PROBLEM STATEMENT

To classify EEG signals into normal and abnormal EEG signals using different methods and to compare the results.

3. INTRODUCTION

The brain is a complex and important part of the human being. It also keeps many secrets of how memories are stored in our brain, such as various abnormalities such as seizures. Now that we have reached the stage of such research, we are very close to discovering the secrets of the brain.

Among the various functions of analysis, electro-encephalograms are widely used in research. The diagnosis of a patient usually takes 20 minutes. This period is not always sufficient for the diagnosis of epilepsy or other compelling disorders.

Here comes the question - how do we capture the EEG? EEGs are basically brain waves, captured by a metal disc placed on the skull. Whenever our brain uses it, the brain releases signals and metal disks, i.e. electrodes, that capture this signal and store it in a record as waves.

EEG reports are confirmed manually by clinically certified physicians. Since this is done manually, it can take a long time to thoroughly analyze the report. Not only this, the analysis is entirely based on the examiner's subjective explanation. This may cause some errors.

As we can see that there can be various problems when manually analyzing, one can try to approach this problem using process automation. The idea of fully automating the process is interesting. But to do this we need to fix the different modules separately and correctly. So in this project, we are looking at how to filter EEG reports in both general and exceptional cases.

We want to proceed with pre-processing data with different methods to find out the specific features in those EEGs. Then comes the training of different models. To do this we hope to use pre-trained models with traditional machine learning algorithms.

Because intensive learning is objective to the features currently used in visual inspection and has the potential to be learned from raw data, it may be an alternative to traditional machine learning methods for visual inspection and EEG analysis.

4.LITERATURE SURVEY

There are many researches that are already being done for this EEG classification. Among these the most prominent ones are:

1. Automated Identification of Abnormal EEGs. By Lopez, S. (2017). Temple University.
2. Deep Learning Enabled Automatic Abnormal EEG Identification. by Subhrajit Roy, Isabell Kiral-Kornek, and Stefan Harrer, IEEE Senior Member
3. Automated detection of abnormal EEG signals using localized wavelet filter banks. by Manish Sharma , Sohamkumar Patel , U. Rajendra Acharya
4. A new feature extraction and classification mechanisms For EEG signal processing. by Hemant Choubey, Alpana Pandey

1.

For the first paper, we can see that it was first introduced by the Temple university Hospital. As the EEG classification by Human Examiner takes a lot of time, they first proposed that automation of this process would be much beneficial in the medical science.

Their main approach was to implement a classifier based on Hidden Markov Model (HMM) , Convolution Neural Network – Multiple Layer Perceptron (CNN-MLP), KNN and Random Forest Ensemble Classifier.

The performance is as follows in their research:

Table 1. Summary of the performance for all the evaluated systems

System Description	Error (%)
kNN (k=20)	41.8%
RF (Nt=50)	31.7%
PCA-HMM #GM = 3 #HMM States = 3)	25.6%
GMM-HMM (#GM = 3 #HMM States = 3)	17.0%

2.

For the Deep Learning based Automatic Abnormal EEG Identification, they have introduced an approach of using deep neural network in order to classify the data into normal and abnormal. They have used some pre-processing techniques in order to extract features. They didn't use hand-engineering techniques to extract the feature but learn the features.

They have used some deep learning techniques to learn the features. We have also introduced this technique in our project.

They have pre-processed the data into three types –

- Direct using the time series wave signal
- Converting the signal into visual spectrogram
- Converting the signal into GAF

Now after processing the data, they have introduced some classifier models to predict the EEG Signals such as

- Logistic Regression
- MLP Classifier
- 1-D CNN
- 2-D CNN
- 1-D CNN-RNN
- TCNN-RNN

The result obtained by their research work is given as:

Pre. proc.	Algo.	Log. Reg.		MLP		1D-CNN		2D-CNN		1D-CNN-RNN		TCNN-RNN	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Time-series		83.72%	49.09%	69.64%	54.15%	82.04%	76.90%	N/A	N/A	99.16%	82.27%	N/A	N/A
Spectrograms		N/A	N/A	N/A	N/A	N/A	N/A	86.31%	70.39%	N/A	N/A	95.22%	71.48%
GAF		N/A	N/A	N/A	N/A	N/A	N/A	79.46%	68.61%	N/A	N/A	92.55%	67.02%

TABLE I: Comparison of performance for combination of pre-processing techniques and learning algorithms.

3.

For the next paper, they have introduced another research approach as novel stop-band energy (SBE) minimized orthogonal wavelet filter bank. After using this method they have used SVM classifier to classify the EEG signals into abnormal. They have used unorthodox method to implement this classifier.

4.

For the next paper, they have proposed a method of extracting the features using Masking and Check-in based Feature Extraction Technique (MCFET) and an integrated K-Means with K-Nearest Neighbour classification algorithms to classify the EEG Signals into Normal and Abnormal Signals.

For this method of extracting, first mask is generated , then the check in function is calculated, upper and lower envelops are calculated. Finally the features are extracted from this.

After obtaining this feature, they have used knn classifier to classify the signals into normal and abnormal EEG Signal.

5.1.PROPOSED METHODOLOGY

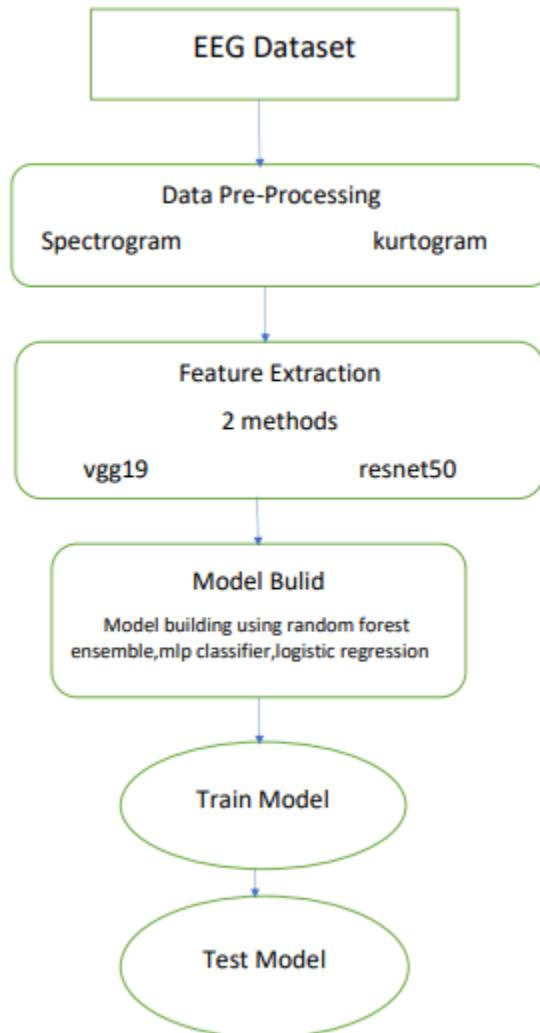
To classify EEG waveform signals into two classes, they are simple and abnormal EEG signals, we introduced a three-phase structure.

In the first stage, we saw that the brain captures the signal and has several nodes plotting on the EEG. In a single session conducted by the examiner, we can see that the patient has 30 to 40 such nodes associated with his or her EEG report. A specific data format is used to store EEG reports. Its name is EDF. Now we have the raw data in EDF format and to use this data, we process it according to our requirements. So, the first step is pre-processing the data.

In the second step, we need to extract the attributes from the processed data. To do this, we want to use certain techniques and take the best technique after achieving the desired results.

In the final step, we need to build some taxonomic models and train using the features collected from the previous step. We then have to test the classification using the test data we have and get the accuracy we want so that the EEG signal can be classified as normal and abnormal EEG signal.

5.2.BLOCK DIAGRAM



In this block diagram, we can see the basic Block Diagram of our work-flow.

5.2.1.

First Block of our diagram is the dataset and for this block of our work-flow, we are mainly focusing on how the data is situated as EDF format. How this data is then converted as matlab data array.

5.2.2.

Second Block of our diagram is the data pre-processing. In this flow of our work-flow, we basically pre-processed the data into suitable data to obtain

temporal frequency spectrum. Our main intuition was that the data is basically signal with temporal axis. So, temporal frequency spectrum would be appropriate for data pre-processing.

5.2.3.

Third block of our block diagram is feature extraction. In this stage, we used pre-trained model to extract features. in that case we used 2 models, i.e VGG19 and ResNet50 model. In convolution neural network, we see that in each convolution layer the model tries to obtain new features from the image and tries to recognise more complex feature. Using that intuition, we used these 2 pre-trained models to extract features from our dataset.

5.2.4.

In the fourth block, we built our main classifier model that would classify our data into Normal and Abnormal EEG signals. There are many binary classifier that are available. We used random forest ensemble, logistic regression and MLP classifier.

5.2.5.

In the next step, we used our extracted feature vector to train our 3 models.

5.2.6.

After that, we tested our test data to obtain how much accuracy our model can gain. This part concludes our project.

6.1.DATASET

To implement our project, we need a classification of EEG signals and their general and exceptional ones. For this we have received the "Temple University Hospital EEG Data Corpus", which was written by our esteemed patron, Dr.Rishav Singh.

In this dataset, the EEG signal is obtained from several patients by obtaining an approximately 15-minute EEG session and by examining the brain signals with the help of a metal disc placed on the patient's skull. These codes are stored in EDF data format. These EDF data formats include where the signals are coming from, the sample rate of the signal, the waveform signal and so on.

The dataset distribution is given below:

Table 1. File statistics for the full evaluation set.

Evaluation					
Description	Files		Patients		hours
Abnormal	126	46.4%	105	41.5%	48.9
Normal	150	53.6%	148	58.5%	55.4
Total	276	100.0%	253	100.0%	104.4

Table 2. File statistics for the full training set.

Training					
Description	Files		Patients		Hours
Abnormal	1346	50.2%	899	42.1%	546.4
Normal	1371	49.8%	1239	58.0%	518.3
Total	2717	100.0%	2138	100.0%	1064.7

EEG:

The electro-encephalogram (EEG) is a signal that is recorded by capturing the electrical activity of the brain by putting metal discs (known as nodes) on the scalp.

EEG activity is measured in micro-volt. It has 4 main frequency observed on human. They are:

1. Delta:

It has a frequency of 3 Hz or below. It has the highest amplitude and the slowest waves. It is mostly found in frontal area in adult human.

2. Theta:

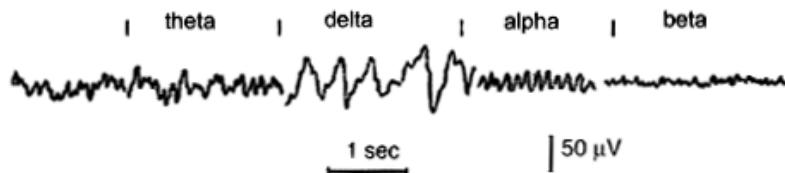
It has a frequency of 3.5 to 7.5 Hz. It can be found on children upto the age of 13 and when an average human sleeps. But when it is found on adults when they are awake, it is abnormal.

3. Alpha:

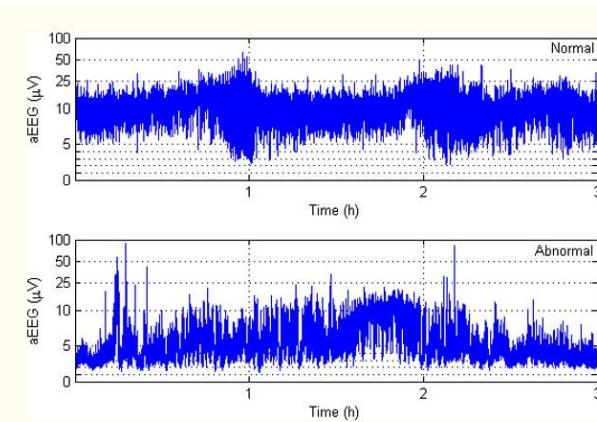
It has a frequency of 7.5 to 13 Hz. It occurs when we close our eyes and relax.

4. Beta:

It has a frequency of 14 and greater. It is a fast activity.



Different EEG Frequencies obtained on human



Example of a Normal and Abnormal EEG

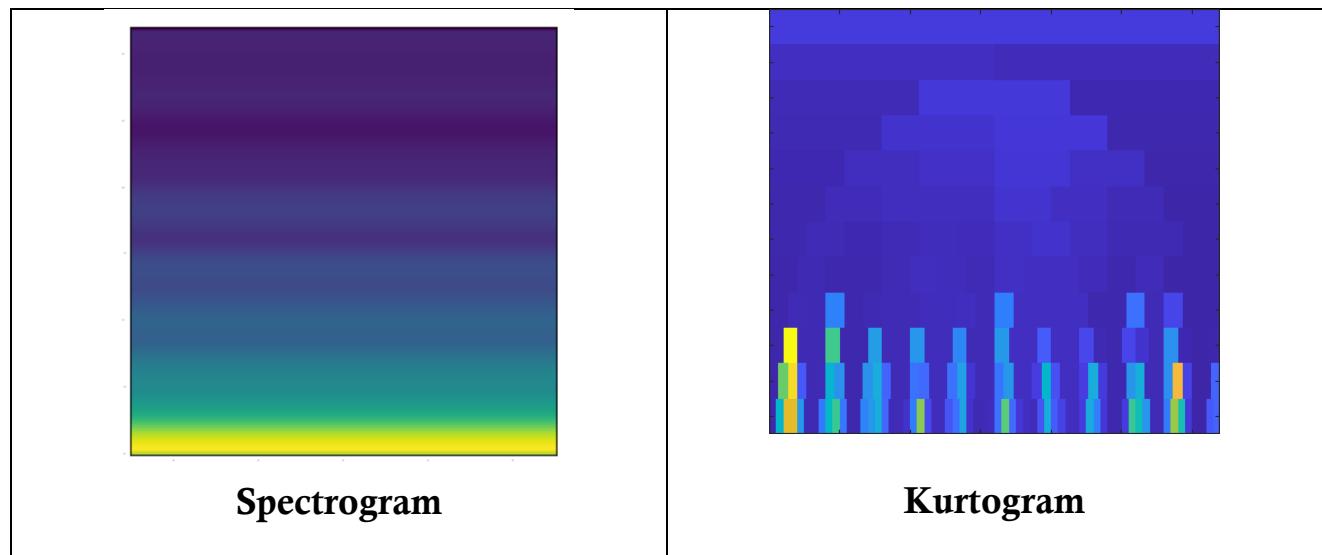
6.2. DATA PRE-PROCESSING

First, the dataset is in EDF format. Therefore, in order to process this data into our useful data, we have converted this EDF data into Matlab array data. Now that we have the matrix data we can apply a lot of pre-processing techniques on it.

The dataset consists of several nodes from which 21 signals are extracted from each data point and for this our accuracy is increased by 10%.

Several pre-processing methods are available. We wanted to use CNN-based feature extraction in the next step, using pre-processing methods that convert the signals into a temporary frequency spectrum. Some of these methods are spectrogram and kurtogram.

We used these two methods to pre-process the data into spectrograms and kurtograms and stored them for the next step. It helps to gather transient frequency spectrum properties.



In order to read the dataset in python, we have used “pyedflib” library in python. After reading the data in python, we obtained the signal in array and used that array to save in matlab array using the “scipy.io” and savemat() function.

After obtaining the matlab data array, we used matlab software to convert the array signal into spectrogram and kurtogram.

In order to obtain the spectrogram and kurtogram took a lot of time, one whole day to be precise. I think the time took such a long time as the processing of a single data was done at a time. For this reason, for converting 3k data points individually took such a long time.

6.2.1. SPECTROGRAM

The spectrogram contains a compromise between time resolution and frequency resolution: the large window provides less local processing at a time and more discrimination at times. The window receives a piece of signal, where the spectral features are almost invariant

the found parts move the timer window at a certain interval. The spectrogram is defined as the size of $S(m, k)$, represented as $A(m, k)$, as shown in equation

$$A(m,k) = \frac{1}{N} |S(m,k)|^2.$$

Spectrogram adjustment can be improved to change the window length; The long window offers the best solution for frequency, but the worst time fix. A shorter window, however, offers better timing but better resolution. Good visibility in the spectrogram depends on choosing the right window length and spacing. Figure 2 shows the signal spectrogram, which is a multi-signal signal display. Spectrum programming is usually as follows: x-axis represents time, y-axis represents frequency, and the third dimension is the amplitude (visual content) of the frequency, which is encoded. This three-dimensional data can also create a 3D structure, where power is represented as height on z-axis but a 2D chart provides better understanding.

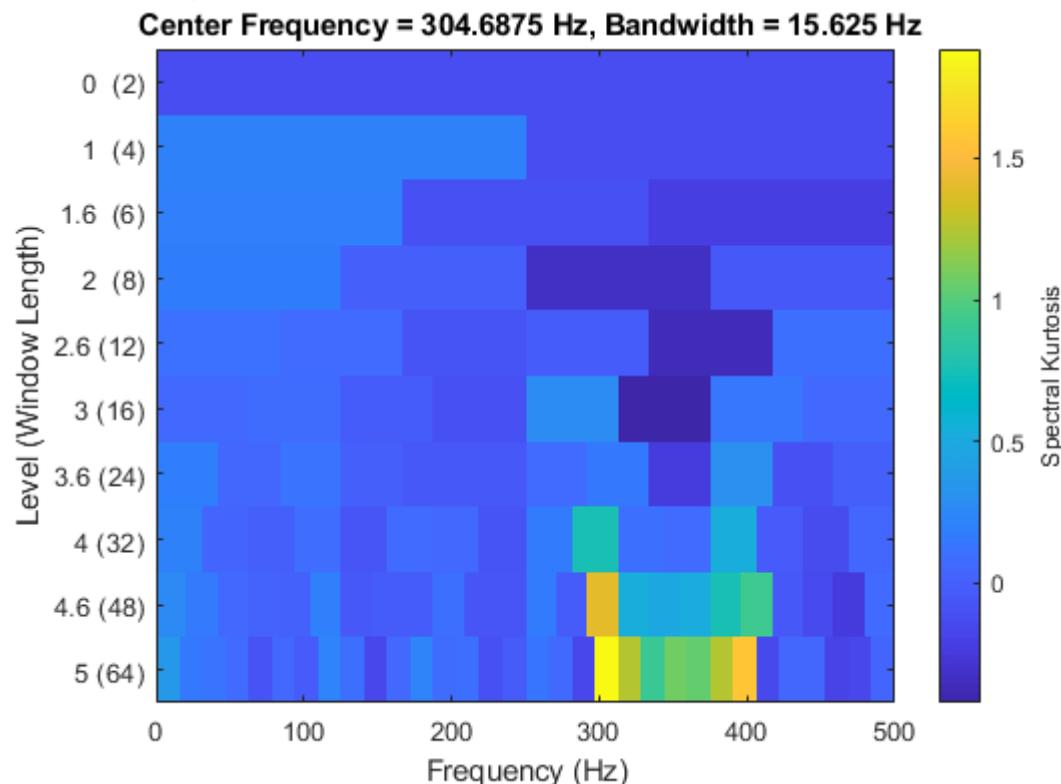


Spectrogram of
An EEG signal

6.2.2. KURTOGRAM

A kurtogram is a spectral analysis tool used to detect irregularities in the signal area. It can be used effectively to find a suitable filter to remove the feature element from the vibration damp, because error signal-induced signal times can be considered static. However, the efficiency of the kurtogram decreases when the signal is collected in a structure that operates under various speed conditions.

A kurtogram is a newly developed spectral analysis tool for the fourth order to detect and mark non-signal objects. The paradigm is based on the premise that each temporal type is associated with the correct dyad {frequency (frequency / frequency) dyad {} which increases its kurtosis, hence its detection. However, a complete overhaul of all aircraft () is a daunting task that is incomparable to online industrial applications.



7.FEATURE EXTRACTION

For the second phase of our project, we intend to use CNN based pre-trained models to extract features from spectrograms and kurtograms. We used VGG19 and RESNET50 pre-trained models for our project.

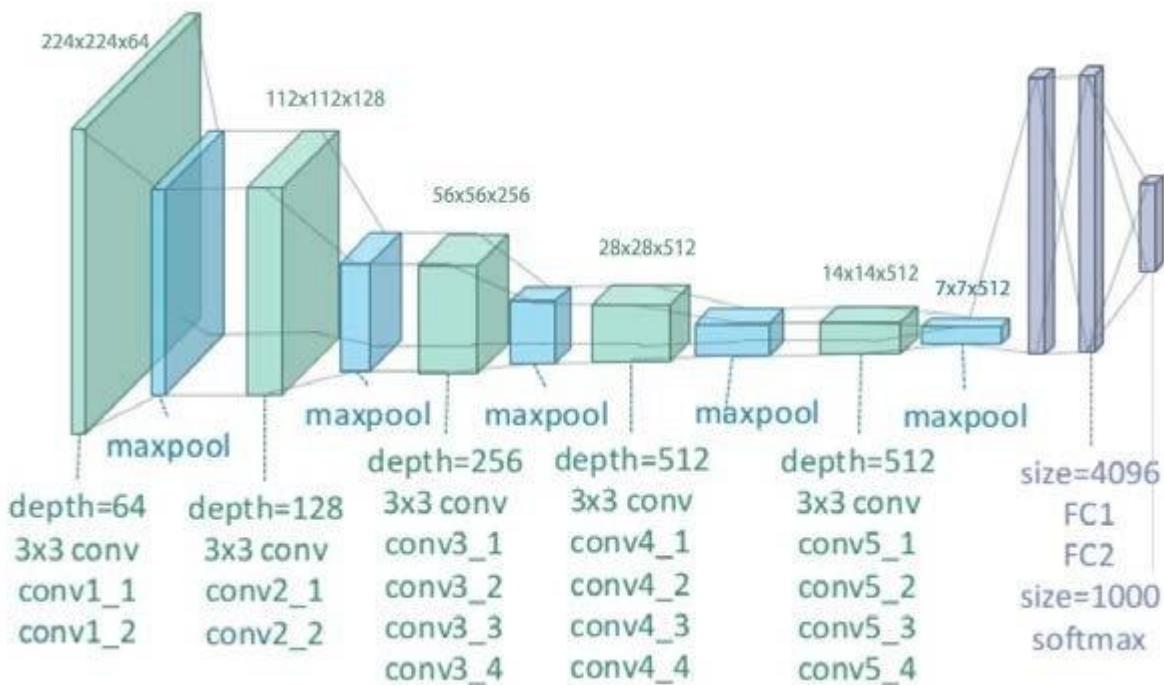
The VGG19 already trained on the "Imagenet" dataset has 8 convolution layers and fully integrated layers with a total of 11 layers. In the case of VGG19, we used "block4_conv1" layer output as the extracted attribute. After obtaining the attribute, we flatten the attribute, construct it as a feature vector and use it as training data in the next step.

ResNet50 consists of 5 steps, each consisting of a convention block and an identity block. Each convention block has 3 convolution layers and each detection block has 3 convolution layers. For our project, we used some layers like "conv3_block2_add".

After collecting the properties, we flattened the output layer and obtained a vector with the desired properties. Now we use it to train our taxonomy model to achieve accuracy.

7.1. VGG19:

For the VGG19 model, the architecture is given as:



VGG19 contains total 19 layers of which 16 layers are convolution layer, 3 layers are fully connected layer and 5 max-pool layer and 1 softmax layer.

Convolution Layer:

Convolution layer is the layer which uses convolution operation to extract features from the image. Convolution operation is done on the image by several filters or kernels. These filters learn the features and make the whole convolution process successful.

Fully Connected Layer:

Fully Connected Layer is that layer of which every neuron is connected to every neuron of next layer. It is like a layer of deep neural network where we can see that every neuron of that layer is connected to every neuron of the next and previous layer.

Max Pool Layer:

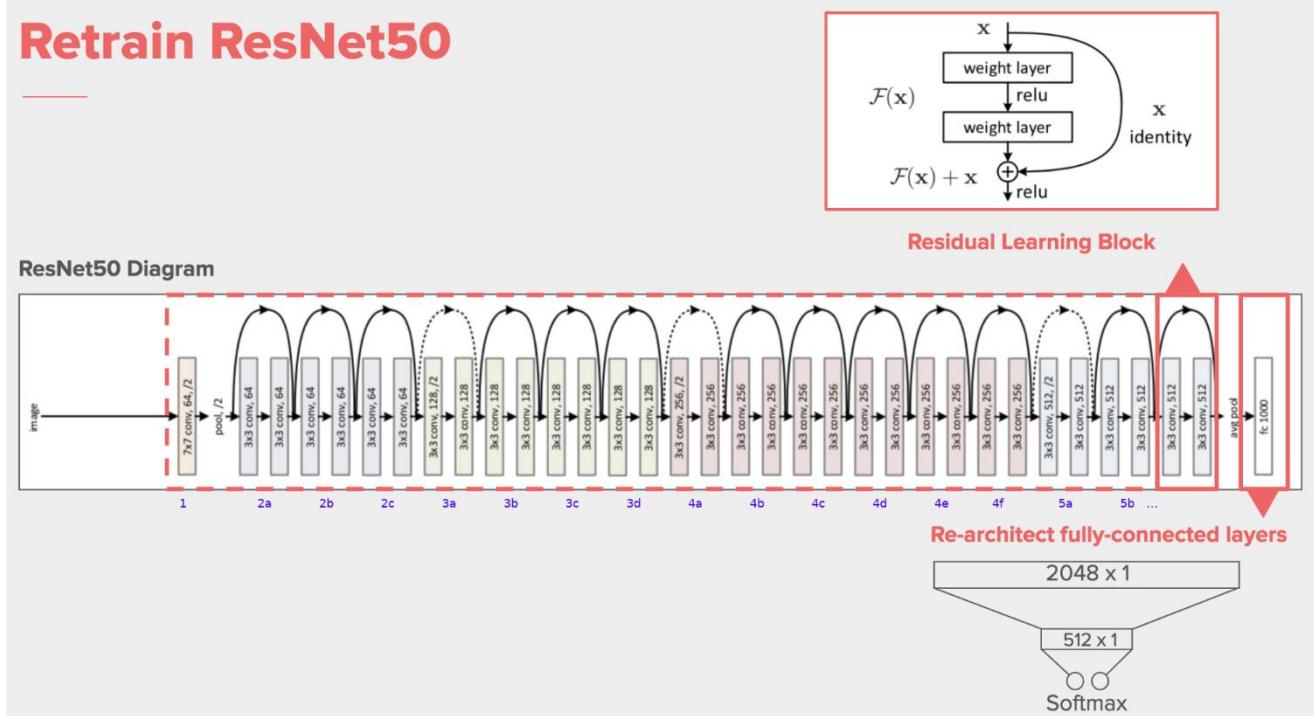
It decreases the shape of the features. Therefore the number of parameters to learn also decreases. Therefore it increases the time efficiency of the model. It uses max as a pool function meaning that the max value will be pooled onto the next layer.

Softmax Layer:

It is the layer that produces output. It basically assigns probability to each of the classes that are present in the model. After assigning the probability, we can simply see that the class having highest probability would be the predicted class.

7.2. ResNet50:

For the ResNet50 model, the architecture is as follows:



It has 48 Convolution layer and 1 max pool and 1 average pooling layer.

Residual Learning:

In deep CNN, several convolution layer, fully connected layer, pooling layer are stacked upon each other. They learn several features at the end of layers. But in residual learning, instead of learning features, it learns residuals. Residuals can be simply described as subtraction feature learned from that input layer. ResNet does this learning by connecting n th input layer to the input layer of $(n+p)$ th layer. By this concept ResNet50 is implemented.

$$Y(L) = H(X_L) + F(X_L, W_L)$$

$$X(L+1) = G(Y_L)$$

F is a stacked non-linear layer and G is a Relu activation function. It has been found that when $G(Y_L)$ and $H(X_L)$ are identity mappings, the signal can be directly sent to any other unit whether backward or forward.

Convolution Layer:

Convolution layer is the layer which uses convolution operation to extract features from the image. Convolution operation is done on the image by several filters or kernels. These filters learn the features and make the whole convolution process successful.

Max Pool Layer:

It decreases the shape of the features. Therefore the number of parameters to learn also decreases. Therefore it increases the time efficiency of the model. It uses max as a pool function meaning that the max value will be pooled onto the next layer.

Average Pool Layer:

It decreases the shape of the features. Therefore the number of parameters to learn also decreases. Therefore it increases the time efficiency of the model. It uses average as a pool function meaning that the max value will be pooled onto the next layer.

8.MODEL BUILD

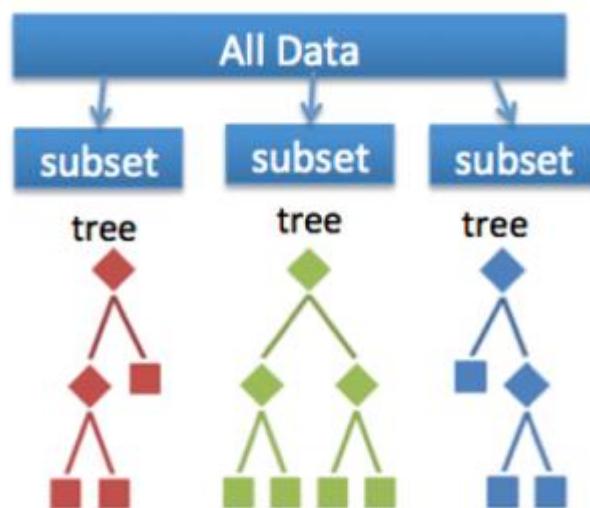
For our classifier model, we used several models. These models are mainly Random Forest Classifier, Logistic Regression and MLP Classifier.

8.1. Random Forest Classifier:

A random forest algorithm with supervised sections and a regression algorithm. As the name suggests, this random algorithm creates a forest with several trees.

Usually, as the trees grow in the forest, the forest becomes stronger. Similarly, in random forest planning, as the number of trees in a forest grows, the results are more accurate.

In simple terms, a random forest forms multiple decision-making trees (called forests) and combines them to produce more accurate and stable predictions. The constructive forest is a collection of Decision Trees, trained in the form of bagging.



The random forest ensures that the performance of each tree is not significantly related to the behavior of any other tree in the model by using the following two methods:

- Bagging or Bootstrap Aggregation
- Random feature selection

- Bagging or Bootstrap Aggregation:

Decision trees are more sensitive to the data they have been trained on, a small change in the training data set can lead to a very different tree construction.

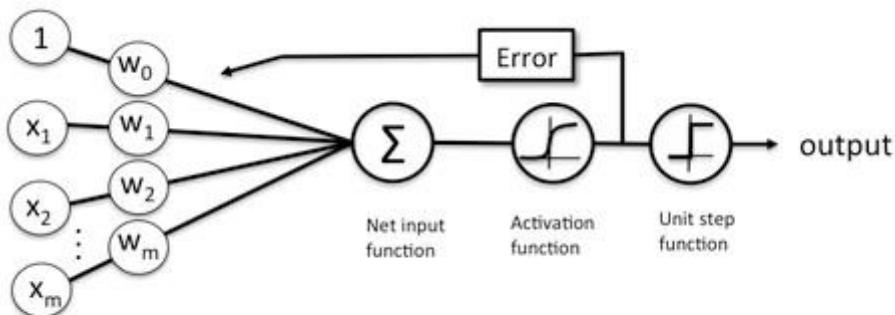
The random forest takes advantage of this by allowing each tree to randomly sample in the database by inserting another, which results in a variety of trees. This process is called Bagging.

- Random feature selection:

each tree in a random forest can only select from a random subset of features. This forces a very large difference between the trees in the model and ultimately leads to lower connections in the trees and more division.

So in a random forest, we end up with trees trained in different sets of data and use a variety of factors to make decisions.

8.2. Logistic Regression:



Schematic of a logistic regression classifier.

In this figure logistic regression is represented as a single layer neural network.

The nodes are represented as neurons of the network and the node having 1 inside is called as bias and the weights w_i , where $i = 0, 1, 2, \dots, m$ are the parameters of this logistic regression.

- The leftmost layer is the **Input Layer**. It contains the value of input vector x_i , where $i = 0, 1, 2, \dots, m$. x_0 is known as the bias. Here the node having 1 inside is known as x_0 .
- The next layer is the **Output Layer**. It computes the weighted mean of the input vector and passes through the sigmoid activation function to get the value in the range of (0,1).

$$Y = F(W^T X + b) = F\left(\sum_{i=1}^m w_i x_i + b\right)$$

Where W is weight vector and X is the input vector, b is the bias. F(x) is the sigmoid function.

$$F(x) = \frac{1}{1 + e^{-x}}$$

Now the loss function of the model is binary cross-entropy. That is:

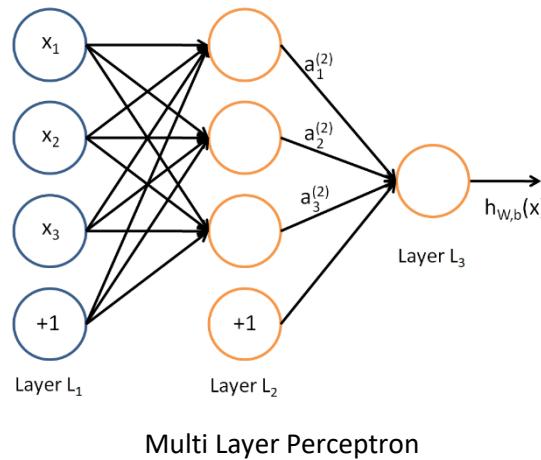
$$L(w, b) = -\frac{1}{N} \sum_{i=1}^N y_i \log(f(o_i^{k+1})) + (1 - y_i) \log(1 - f(o_i^{k+1}))$$

Now after calculating the loss function, we need to minimize it using back-propagation algorithm. The algorithm uses gradient descent to find the value of the parameters and update them by using the following equations:

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

In this way, Logistic regression works. After that we take our test data to test our model.

8.3. MLP Classifier:



In this figure, the nodes are represented as the neurons of the network and the node having $+1$ inside denotes the bias of the network.

- The leftmost layer denotes the **Input Layer**. The nodes having x_i , where $i = 1, 2, 3$ denotes the input fed into the network.
- The next layer is known as the **Hidden Layer**. The parameters of these layers are learnt and not observed during the iteration. Hence its name hidden layer.
- The rightmost layer is known as the **Output Layer**. It gives the output of the model.

Algorithm:

In MLP the input layer mainly takes input vector and assigns it to each of the node of input layer x_i , where $i = 1, 2, 3, \dots, n$

$$Model_{out} = f(W^T X + b) = f\left(\sum_{i=1}^n (w_i x_i + b)\right)$$

Where W is the weight vector and X is the input vector, $f(x)$ is the sigmoid activation function.

$$f(x) = \frac{1}{1 + e^{-x}}$$

In MLP there can be more than one hidden layer. For that, the main equation changes a bit. Let us assume that in kth layer there are n neurons and the sum of the weighted input of the ith layer is o_i^k . Now by using this we can find out the next output layer that is o_i^{k+1} .

$$\text{For } k=1, \quad o_i^{(2)} = \sum_1^n (w_{ij}^1 x_j + b_i^1)$$

For $k \geq 2$,

$$o_i^{(k+1)} = \sum_1^n (w_{ij}^k f(o_j^k) + b_i^k)$$

Now the loss function of the model is binary cross-entropy. That is:

$$L(w, b) = -\frac{1}{N} \sum_{i=1}^N y_i \log(f(o_i^{k+1})) + (1 - y_i) \log(1 - f(o_i^{k+1}))$$

Now after calculating the loss function, we need to minimize it using back-propagation algorithm. The algorithm uses gradient descent to find the value of the parameters and update them by using the following equations:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} L(W, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} L(W, b)$$

Now in this way, MLP Classifier learns and give us output.

9. TRAINING THE MODEL

To train the models, we used the collected feature vector and fed it to many of the models we created. We built and trained our model on Google Collab using 25GB RAM and 6GB graphics memory. For training, we used two feature vectors. One is derived from VGG19 and the other from ResNet50.

9.1. Random Forest Classifier:

For the random forest ensemble, we used a maximum depth of 5000 and trained until it was completed. Takes 20 to 30 minutes with the help of our GPU.

We can see that we have used sklearn library to implement our Random Forest Classifier. The model is named as clf and we have fitted the model with our extracted features.

9.2. Logistic Regression:

For logistic regression, we used 100 to 200 repetitions to train our model. It took 30 minutes.

We are able to see that we have used sklearn to implement our logistic regression. The model that we are using is having max iteration of 100. The solver that the model is using is sag. It converges faster for large dataset and also handles L2 or no penalty. For that reason, we have used sag as our solver. After that we have fitted that model with our feature vector.

9.3. MLP Classifier:

For the MLP classification, we used 100 iterations to train our model. It took about 30 minutes to train.

We can see that the MLPClassifier model is implemented using sklearn library. After that it is having some parameters such as max iteration and so on. After that we have fitted the model using our extracted feature vector.

10. TESTING THE MODEL

To test for test data, we undergo the same feature extraction as we did for data training. After we collected the features from the test data, we used many of our trained models to estimate the output and compare different test accuracies..

10.1. Random Forest Classifier:

We can see that for random forest classifier after fitting, we have used our test data extracted vector to predict if it is normal and abnormal. After that we have stored it into classifier_rf_pred array for generating the accuracy for later.

10.2. Logistic Regression:

We can see that for Logistic Regression after fitting, we have used our test data extracted vector to predict if it is normal and abnormal. After that we have stored it into logistic_regression_pred array for generating the accuracy for later.

10.3. MLP Classifier:

We can see that for MLP Classifier after fitting, we have used our test data extracted vector to predict if it is normal and abnormal. After that we have stored it into MLP_pred array for generating the accuracy for later.

11.RESULT

The several accuracies obtained by our models are given below:

11.1. VGG19 model based feature vector:

11.1.1. Random Forest Classifier:

```
print(str(sum(classifier_rf_pred==y_test1)/rftest.shape[0]*100)+"% Accuracy")
```

63.67%

In this case, we are able to see that the accuracy obtained is 63.67%.

11.1.2. Logistic Regression:

```
print(str(sum(logistic_regression_pred==y_test1)/bottleneck_test.shape[0]*100)+"% Accuracy")
```

60.34%

In this case, we are able to see that the accuracy obtained is 60.34%.

11.1.3. MLP Classifier:

```
print(str(sum(MLP_pred==y_test1)/bottleneck_test.shape[0]*100)+"% Accuracy")
```

61.27%

In this case, we are able to see that the accuracy obtained is 61.27%.

To summarise the accuracies for all the models, the beneath table is given:

Model	Accuracy
Random Forest Ensemble	63.67%
Logistic Regression	60.34%
MLP Classifier	61.27%

11.2. ResNet50 model based feature vector:

11.2.1. Random Forest Classifier:

```
print(str(sum(classifier_rf_pred==y_test1)/rftest.shape[0]*100)+"% Accuracy")
```

61.34%

In this case, we are able to see that the accuracy obtained is 61.34%.

11.2.2. Logistic Regression:

```
print(str(sum(logistic_regression_pred==y_test1)/bottleneck_test.shape[0]*100)+"% Accuracy")
```

59.08%

In this case, we are able to see that the accuracy obtained is 59.08%.

11.2.3. MLP Classifier:

```
print(str(sum(MLP_pred==y_test1)/bottleneck_test.shape[0]*100)+"% Accuracy")
```

60.25%

In this case, we are able to see that the accuracy obtained is 60.25%.

To summarise the accuracies for all the model the beneath table is given:

Model	Accuracy
Random Forest Ensemble	61.34%
Logistic Regression	59.08%
MLP Classifier	60.25%

From this above tables, we can see that VGG19 is better than ResNet50 to extract features from our Dataset. And the best model that gave the best accuracy is Random Forest Ensemble.

12.CONCLUSION

The division of deep learning has been used successfully in many EEG activities, including the burden of mental performance, the goal on the sleep stage, the perception of energy and emotions associated with the event. The design of these in-depth network studies is very different from the input structure and network structure.

Many studies analyze multiple public databases, allowing them to directly compare the performance of categories based on their design. In general, CNN, RNN and DBN belong to other types of intensive networks such as SAE and MLPN. In addition, CNN used signal values or (spectrogram) images as input, and DBN did very well when using signal values or attributes.

We also discussed in-depth network recommendations for each type of work. This recommendation diagram is provided with the expectation that it will guide the delivery of intensive learning in EEG datasets in future research. Boltzmann's equipment has shown promise with accuracy in the learning stages and transmission compared to traditional designs.

We appreciate the in-depth study of these compounds, especially the number and layout of different layers, including RBMs, repetitive layers, fixture layers and fully joined layers. Without network design, we encourage further research into how deep networks translate into a raw EEG and compare to a degraded one, as this has not yet happened.

13.FUTURE SCOPE

From our project, we can see that various models have tried to reduce the error rate and the best error rate is 36%. It is still not enough, though it does reach cutting-edge accuracy. For further improvement, we may use other pre-processing methods and use other nodes that are not included in our project.

Also, the temporary part of many performances is very large, which means we work with sessions longer than 15 minutes. During these 15 minutes, there is a brief interruption of the deviation from the normal state of the EEG signal. For that we can test where the abnormality starts and try to use our model to check and classify only that period.