Classification of EEG Signals into Normal and Abnormal

Project Report

BACHELOR OF TECHNOLOGY
Computer Science and Engineering

Submitted By
Pritam Sarkar (171210044)
Shubham Musale (171210050)

NATIONAL INSTITUTE OF TECHNOLOGY, DELHI

ABSTRACT

We know that our brain functions with various signals. These signals are generated with every activity that we do, even in our sleep. We basically capture these signals by putting electrodes on our scalp and then in the EEG it is recorded as wave patterns. Normal activity would have the usual pattern but abnormal EEG has some distinguishable features. Doctors can identify abnormal EEG from the normal ones after some observation.

Although EEGs helps us differentiate between Normal and Abnormal, it is still heavily dependent upon the examiner to give the last judgement. In order to interpret the signals captured by EEG, we need the help of an expert in this field. Our main objective is to lessen the burden on examiner and reduce the time of examination.

The final goal of this project is to automate the whole process of examination of classifying EEGs. We have now focused only on classifying an EEG into its appropriate type. This automation will help reducing the overall time required to identify the EEG and help the examiner.

INTRODUCTION

Brain is a complex as well as significant part of human. It also holds many secrets such as how the memories are stored in our brain, how different abnormalities like seizures etc. Now we have come to such point of research that we are much closer to discovering the mysteries of brain.

Among the various measures of analysis, electro-encephalogram is one the most used in the research. Normally outpatient EEG diagnosis takes about 20 min. But this duration is not always sufficient for the analysis of epilepsy or other seizure disorders.

Here comes the question-How do we capture EEG? Basically EEG is the brain waves which captured by metal disc placed on the scalp. Whenever we use our brain, brain emits signal and the metal disc, i.e electrodes, captures this signal and store them in the record as waves.

EEG reports are manually diagnosed by medically certified physicians. As it is performed manually, it may take a lot of time to analyze each of the report thoroughly. Not only this, the analysis is purely based on the subjective interpretation of the examiner. This may lead to some errors.

As we can see that there can be various issue when manually analyzing, we can try to approach this issue using automation of the process. The thought of fully automating the process is interesting. But in order to do that we have to solve the individual module separately and accurately. So in this project, we are focusing on how to filter the EEG reports into normal and abnormal cases.

We want to proceed with first pre-processing the data with various methods to learn the distinguishable features in those EEG. Then comes the training of various models. In order to do that we are hoping to use pre-trained models with traditional machine learning algorithms.

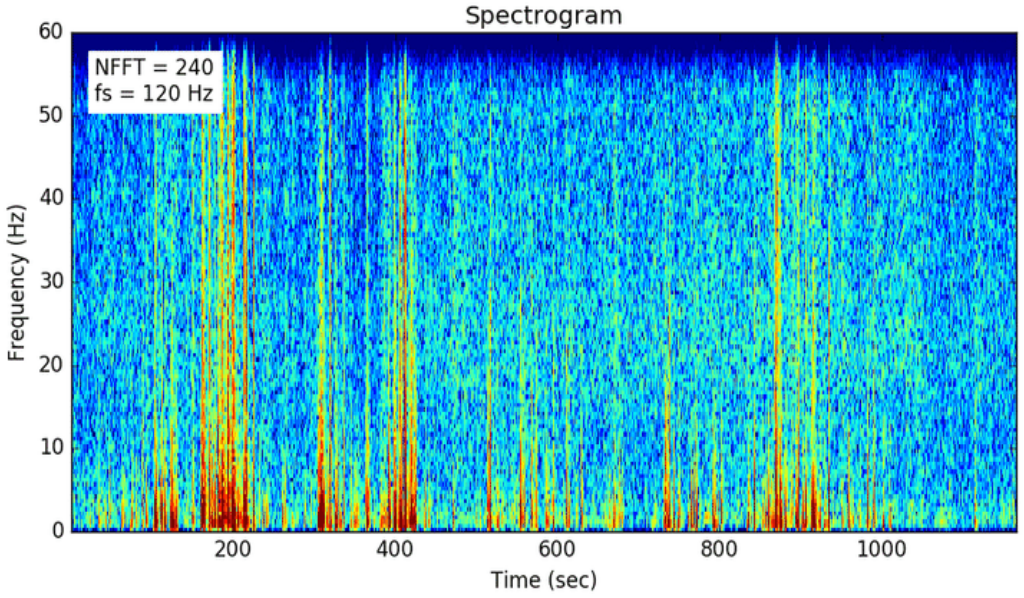
Given that deep learning is unbiased towards the features currently used in visual inspection and is able to learn from raw data, it can be an alternative to visual inspection and traditional machine learning methods for EEG analysis.

Data Preprocessing

The dataset that we are using is the TUH Abnormal EEG Corpus dataset. It has total 2993 EEG records. Among them, 2717 are Train Data and 276 are Evaluation data. Normally a session of EEG recording took about more than 15 min. Each of these sessions are having various files of EEG record. Now one file from each of these sessions are taken for the dataset.

Signal pre-processing is necessary to maximize the signal-to-noise ratio (SNR) because there are many noise sources encountered with the EEG signal. Noise sources can be non-neural or neural.

All EEG data considered in our study were high-pass filtered using a 4th order Butterworth filter with a cutoff frequency of 0.01 Hz. Taking inspiration from that, we converted the recorded EEG time series signal into spectrograms.



Now we have to extract the features from this spectrogram. In order to do that, we are going to use pretrained model of VGG19. After extracting the

features, we are using that feature vector to train our various models and getting the output.

We present results for classical machine learning algorithms such as logistic regression and multi-layer perceptron as well as more sophisticated deep learning algorithms such as convolutional neural networks

Training:

```
[8]  layers_list = ["block4_conv1"]
     for layer_feature in layers_list:
          #downloading model for transfer learning

         optimizer = Adam(lr=0.0001)

         arg_model= Model(inputs=model_vgg19.input, outputs=model_vgg19.get_layer(layer_feature).output)
         arg_model.compile(loss='categorical_crossentropy',
                           optimizer=optimizer,
                           metrics=['accuracy'])

         from keras.applications.vgg19 import preprocess_input #preprocessing the input so that it could wo
         bottleneck_train1=arg_model.predict(preprocess_input(x_train[1000:2000]),batch_size=25,verbose=1)
         #bottleneck_val1=arg_model.predict(preprocess_input(X_val),batch_size=50,verbose=1)
         bottleneck_test1=arg_model.predict(preprocess_input(x_test[100:200]),batch_size=25,verbose=1)
         print(bottleneck_test1.shape)
         print(bottleneck_train1.shape)
```

```
 2/40 [>.............................] - ETA: 1sWARNING:tensorflow:Callbacks method `on_predict_batch_
40/40 [==============================] - 5s 129ms/step
2/4 [==============>...............] - ETA: 0sWARNING:tensorflow:Callbacks method `on_predict_batch_en
4/4 [==============================] - 1s 136ms/step
(100, 32, 32, 512)
(1000, 32, 32, 512)
```

Result:

```
[12] from sklearn.ensemble import RandomForestClassifier
     clf = RandomForestClassifier(max_depth=2000, random_state=0)
     rftrain = bottleneck_train
     clf.fit(rftrain,y_train1)
     rftest = bottleneck_test
     classifer_rf_pred = clf.predict(rftest)
```

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

     62.0% Accuracy
```