Report for Data Science Project

Epilepsy Detection

Under the supervision of:

Ncib Lotfi
Mlaouah Ayoub

Written by :

AMORRI Houssein
BEN NEJI Mohamed
DRISS Mohamed Ahmed
GAYAP Hadrien
LONTCHI TABOUA Freddy
SBOUI Mohamed Kadhem

Table of Contents

Tables

Table of Figures

SUMMARY

Data Science has been a growing technology in the last decade, and its use spread to cover different fields including medicine.
Traditionally, doctors had to check the symptoms of every patient, perform all kinds of scans, and analyse the signals in order to provide the best possible diagnostic for the patient.
All these fastidious and long processes can be very exhausting and wearing for the medical staff, and as a consequence, it becomes prone to human errors.
Today, with the technological advancement, Data Science is becoming a very effective tool that can take care of all these tasks while providing outstanding accuracy rates, and saving, at the same time,a lot of precious time and energy from the doctors.
In this project, the challenge for us was to realize these tasks and use models and algorithms that provide the best predictions.
Amongst all of them, the CNN model provided the best accuracy, and it was this model that we used to create an application to support the doctors

INTRODUCTION

In medicine, manual tasks, scans and analysis are essential for patient diagnoses, but could take a long time and eventually be exhausting for the doctors.
In that context, Data Science has been used to reduce the time and the efforts that the doctors provide, to guarantee the best accuracy possible precisely for epileptic diseases.
In order to tackle this problem, our project was based on the IBM Master Plan Methodology, which is a commonly used method to handle Data Science projects.
In this project, we will be handling electroencephalograms (brain signals), we'll analyse them and filter them in order to prepare our data to be used by algorithms and methods.
We'll finally deploy an application which, we hope, will facilitate doctors' job in detecting epilepsy.

I. BUSINESS UNDERSTANDING & DATA SCIENCE OBJECTIVES

1 Introduction

For the first stage of our project we start with finding a deeper understanding of the subject we will work on.
First, we start by establishing the business objectives we have to set for our project.
Next, we look at the data science objectives that we need to fulfill by the end of our project.

2 Project Context

Epilepsy is a common disease that affects people from all different ages, nonetheless the detection of it is still a tedious work that takes a lot of time and effort from neurologists.
As Data Scientists, we aim to make this process a lot faster and reliable since human error is bound to happen.
The end goal of this project is to help neurologists in their work as well as help patients.

3 IBM Master Plan Methodology

To ensure an efficient work flow we chose to use the IBM Master Plan Methodology. It helps the organization of documents by using a single up-to-date repository of product and service information as well as in taking strategic business initiatives.

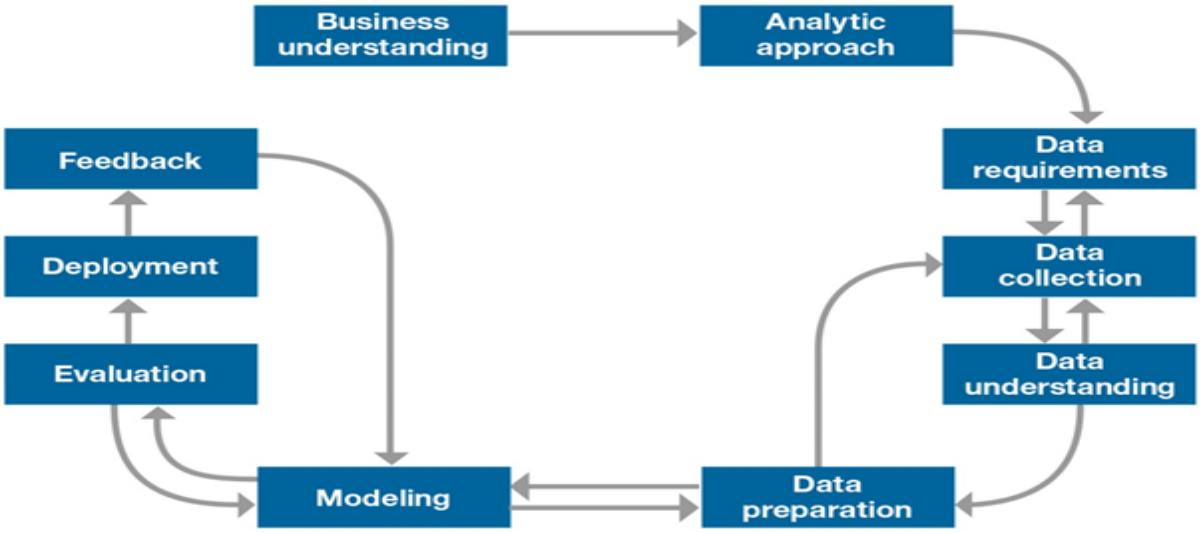


Figure 1 : IBM Master Plan Methodology

4 Business Objectives

For the business understanding we chose to follow the S.M.A.R.T methodology. It gives us the most important points we have to respect in order for our project to achieve its goals from a business perspective.

Figure 2 : S.M.A.R.T Methodology

5 Data Science Objectives

From a Data Science perspective we have a few objectives we hope to achieve through this project mainly :

- Adapt the current deep learning techniques to our subject.
- Practice new deep learning techniques such as transfer learning.
- Fully automate the analytical process for epilepsy detection.
- Develop an application that provides the best accuracy possible.

6 Conclusion

After fully understanding the context of our project, establishing our objectives and the necessities for our work we can move on to the next steps.

II. DATA COLLECTION & PREPARATION

1 Introduction

In the second step of our project we will move on to the data related parts and shed light on the steps we took to accomplish this part.

2 Data Collection

Internal Data

The main data we collected was provided by the Temple University Hospital. It contains 23 gb of information about patients as well as their electroencephalogram in edf files divided by time of visit. The data is divided into two main categories : Epileptic and No-Epiletic.
To use these types of data we had to use the mne library which allows us to read these types of files.

3 Data Understanding

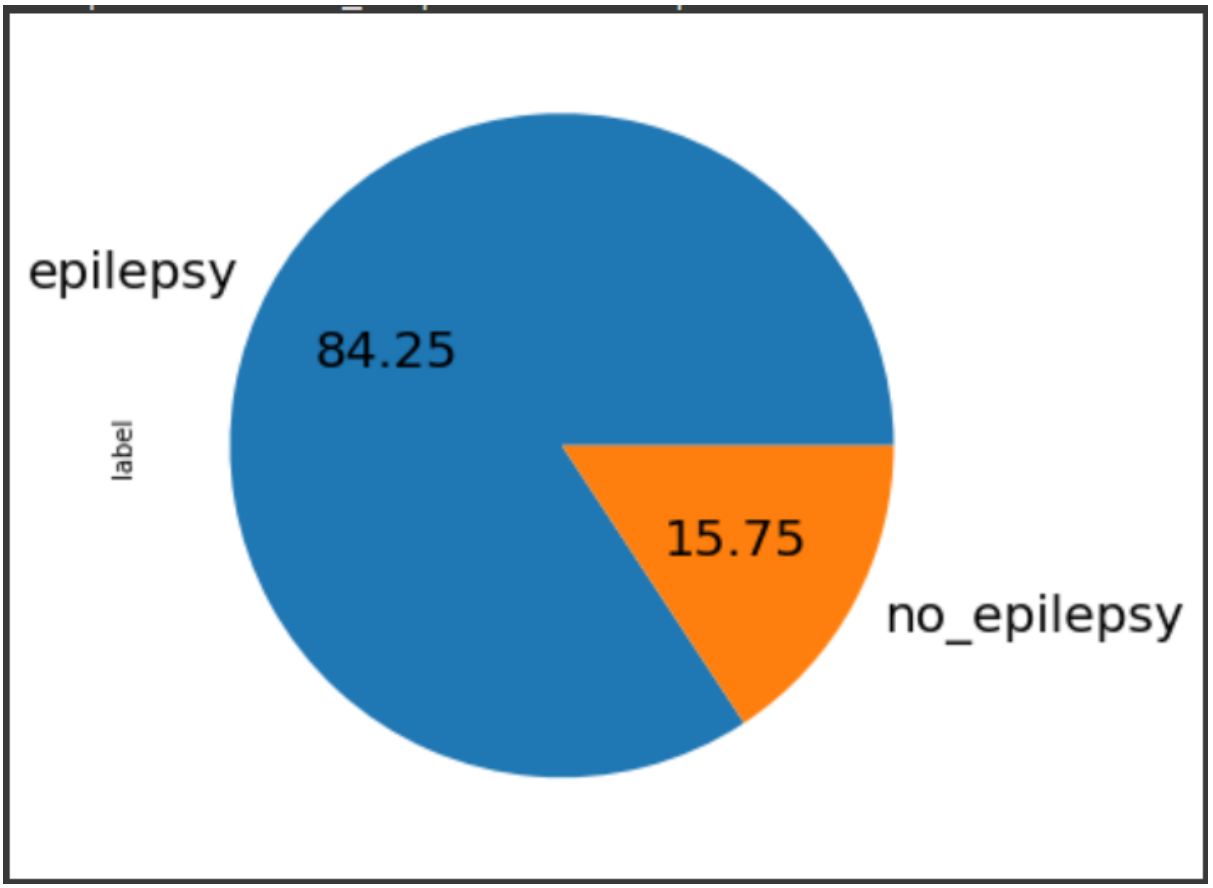


Figure 3: Database samples

The data we are going to use consists of electroencephalograms (EEG). They represent the signal of the brain activity for the patients, detected by electrodes.

```
Entrée [38]: # What does our data look like?

print('Data type: {}\n\n{}\n'.format(type(raw), raw))
# print('{} \n'.format(raw))

# Get the sample rate
print('Sample rate:', raw.info['sfreq'], 'Hz')

# Get the size of the matrix
print('Size of the matrix: {}\n'.format(raw.get_data().shape))

# We can use the mne.info class to learn moure about the data.
print(raw.info)
```

```
Data type: <class 'mne.io.edf.edf.RawEDF'>

<RawEDF | 00000355_s003_t000.edf, 36 x 331250 (1325.0 s), ~91.0 MB, data loaded>

Sample rate: 250.0 Hz
Size of the matrix: (36, 331250)

<Info | 7 non-empty values
 bads: []
 ch_names: EEG FP1-REF, EEG FP2-REF, EEG F3-REF, EEG F4-REF, EEG C3-REF, ...
 chs: 36 EEG
 custom_ref_applied: False
 highpass: 0.0 Hz
 lowpass: 125.0 Hz
 meas_date: 2013-01-04 09:16:52 UTC
 nchan: 36
 projs: []
 sfreq: 250.0 Hz
>
```

Figure 4 : Data Characteristics

Here we are going to inspect our signals and its characteristics. The signal is measured using 36 electrodes, and filtered between a high-pass frequency equal to 125Hz and a low-pass frequency equal to 0Hz.

```
9]: print('The actual data is just a matrix array!\n\n {}\n'.format(raw.get_data()))
```

```
The actual data is just a matrix array!

 [[ 1.35838116e-05  1.89243836e-05  1.99924980e-05 ...  3.50000000e-09
   3.50000000e-09  3.50000000e-09]
 [ 9.15876627e-06  1.26682850e-05  1.61778038e-05 ...  3.50000000e-09
   3.50000000e-09  3.50000000e-09]
 [-7.16812525e-06 -4.57413314e-06 -2.89566765e-06 ...  3.50000000e-09
   3.50000000e-09  3.50000000e-09]
 ...
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

Figure 5 : Electrodes Signal Value Array

Our data can also be displayed in an array which contains the value of the signal for every electrode.

4 Data Preparation

Our goal in this part is to make the data ready for analysis to ensure that it will be accurate and consistent hence making our results valid later on.
We chose for this part to use two types of analytical processes :

- ICA based pre processing.

- Specter based pre processing.

4.1 Channel Selection

When looking at our data we concluded that there's a number of different channels that represent the electrodes used in the process of making the EEGs.

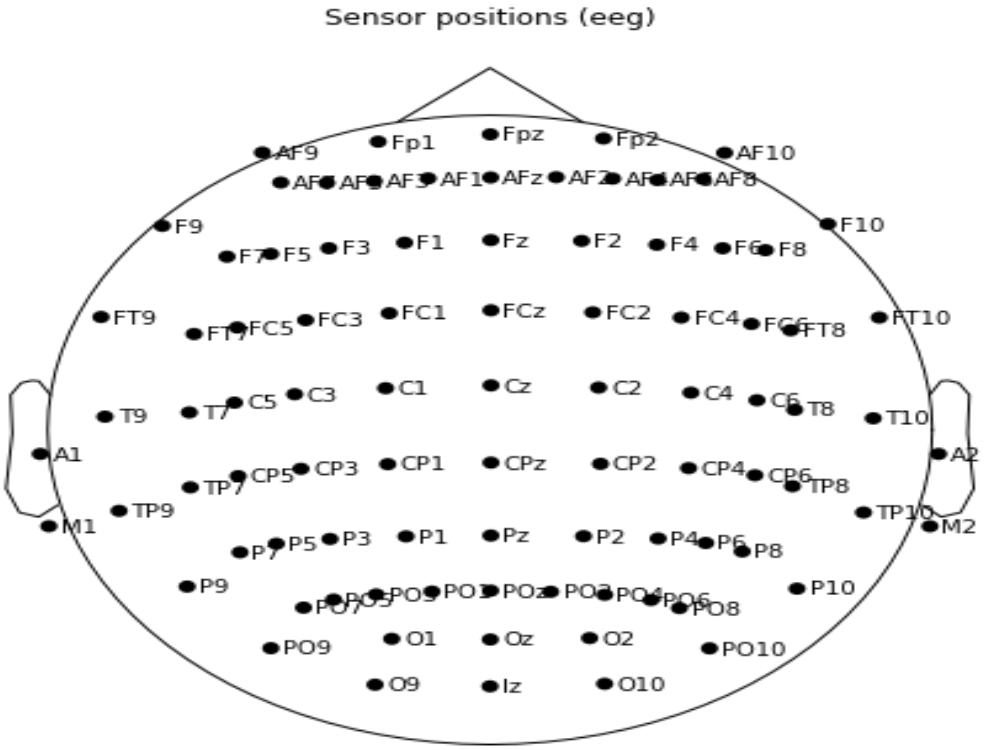


Figure 6 : Electrodes positions in the 10-20 System

But not all of these channels have a relationship with the epilepsy detection which is why we removed the unnecessary data making it 19 electrodes in total instead of 36.

```
                        EEG 30-REF , EEG T1-REF , EEG T2-REF , PHOTIC-REF , IBI , BURSTS , SUPPR ])
def renamechannels2(raw):
    raw.rename_channels({"EEG FP1-REF":"FP1","EEG FP2-REF":"FP2","EEG F3-REF":"F3","EEG F4-REF":"F4","EEG C3-REF":"C3","EEG C4-RE
                         "EEG P3-REF":"P3","EEG P4-REF":"P4","EEG O1-REF":"O1","EEG O2-REF":"O2","EEG F7-REF":"F7","EEG F8-REF":"F8",
                         "EEG T3-REF":"T3","EEG T4-REF":"T4","EEG T5-REF":"T5","EEG T6-REF":"T6",
                         "EEG FZ-REF":"FZ","EEG CZ-REF":"CZ","EEG PZ-REF":"PZ"})
    raw.drop_channels(['EEG ROC-REF','EEG LOC-REF','EEG EKG1-REF','EMG-REF','EEG 26-REF','EEG 27-REF','EEG 28-REF','EEG 29-REF',
                       'EEG 30-REF','EEG A1-REF','EEG A2-REF','EEG T1-REF','EEG T2-REF','PHOTIC-REF','IBI','BURSTS','SUPPR'])
```

Figure 7: Electrode Selection Function

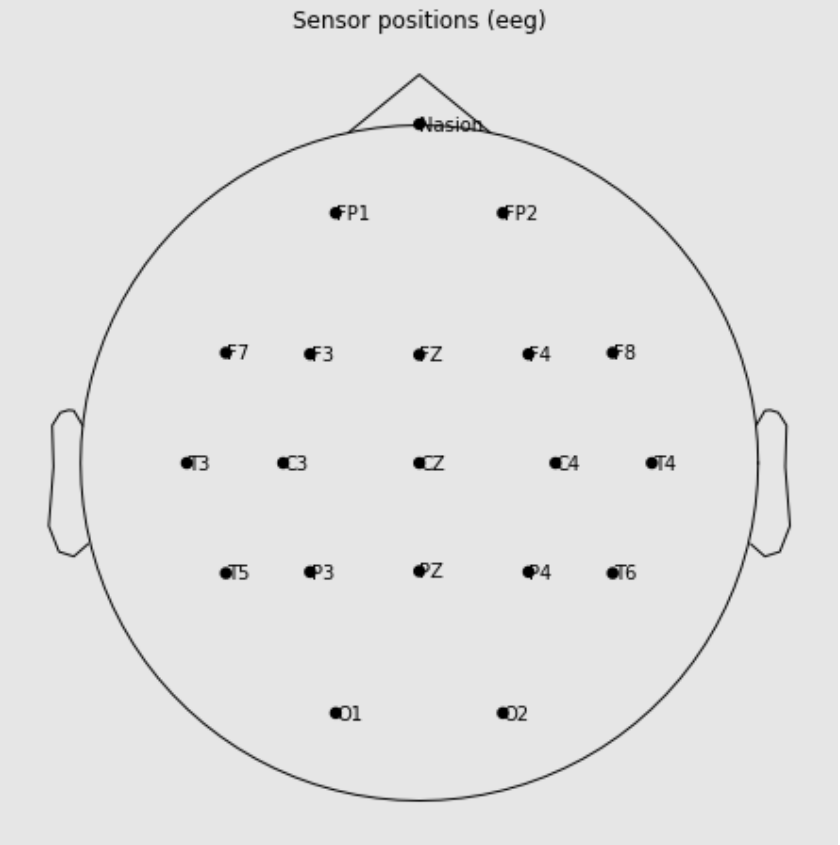


Figure 8: Electrodes used after the Channel Selection

4.2 Filtering

After removing the channels we don't need, we now have to clean the signals we have. Rhythmic brain activity contains multiple overlapping frequencies that can be separated through signal-processing techniques.

These are typically grouped into bands of :

- delta (2-4Hz)
- theta (4-8Hz)
- alpha (8-12Hz)
- beta (15-30Hz),
- lower gamma (30-80Hz).

In our case we have used a pass band to filter between 0.1 and 40 Hz.

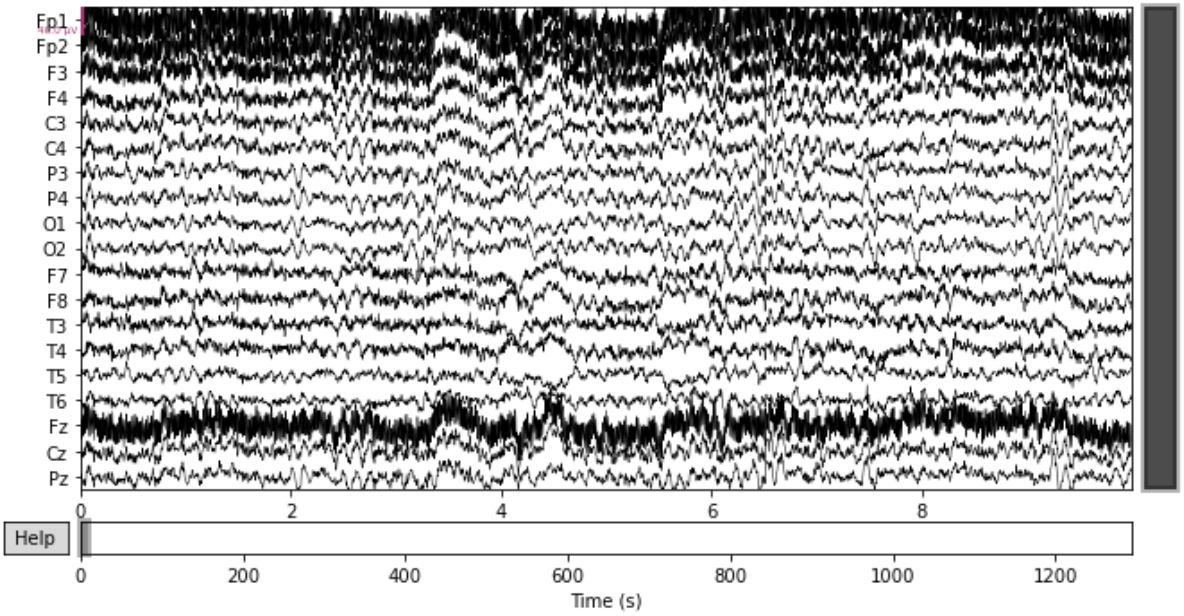


Figure 9 : EEG signal before filtering

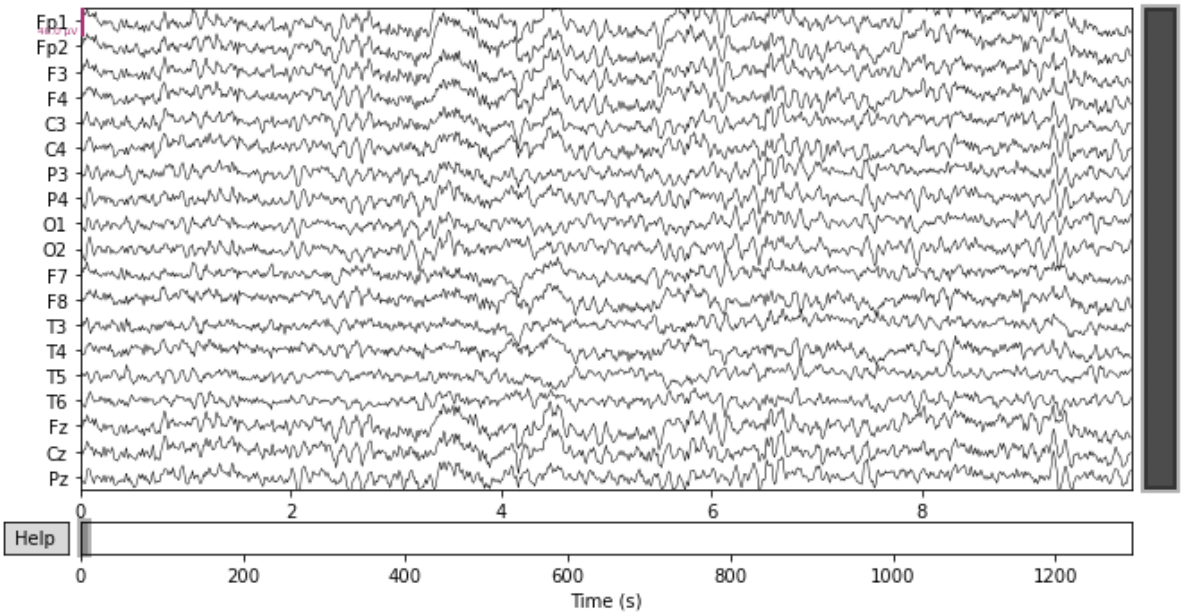


Figure 10 : EEG Signal after filtering

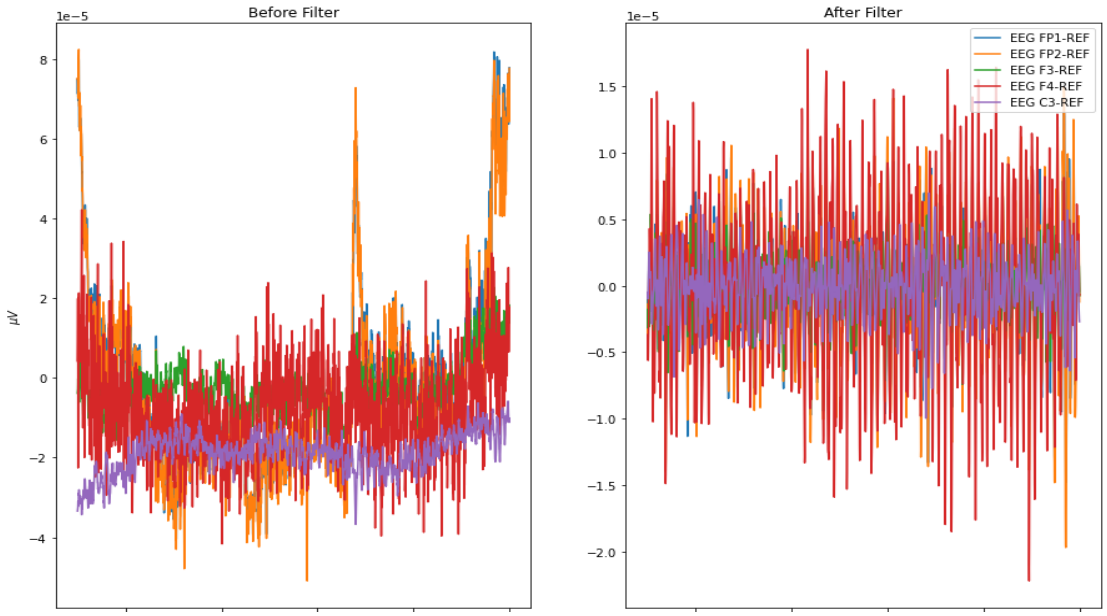


Figure 11 : Signal Comparison before and after filtering

4.3 Analysis Techniques

For this process we chose to use two different types of techniques :

4.3.1 ICA

Independent components analysis (ICA) is a technique for estimating independent source signals from a set of recordings in which the source signals were mixed together in unknown ratios.

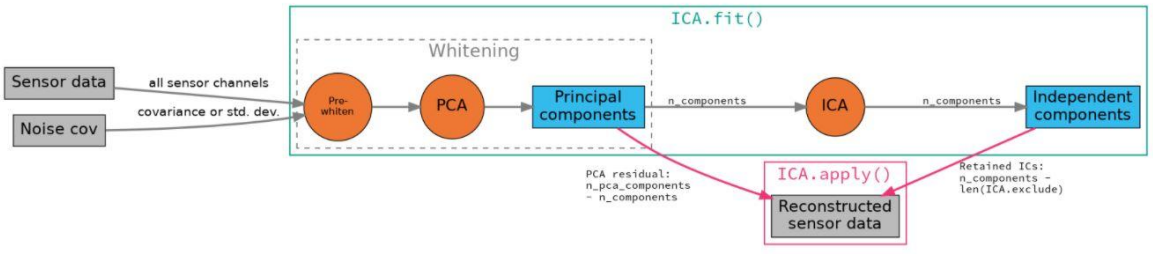


Figure 12 : ICA Technique explanation

During the ICA technique, we have the step concerning PCA, which gives as output the Principal Component for each Electrode.

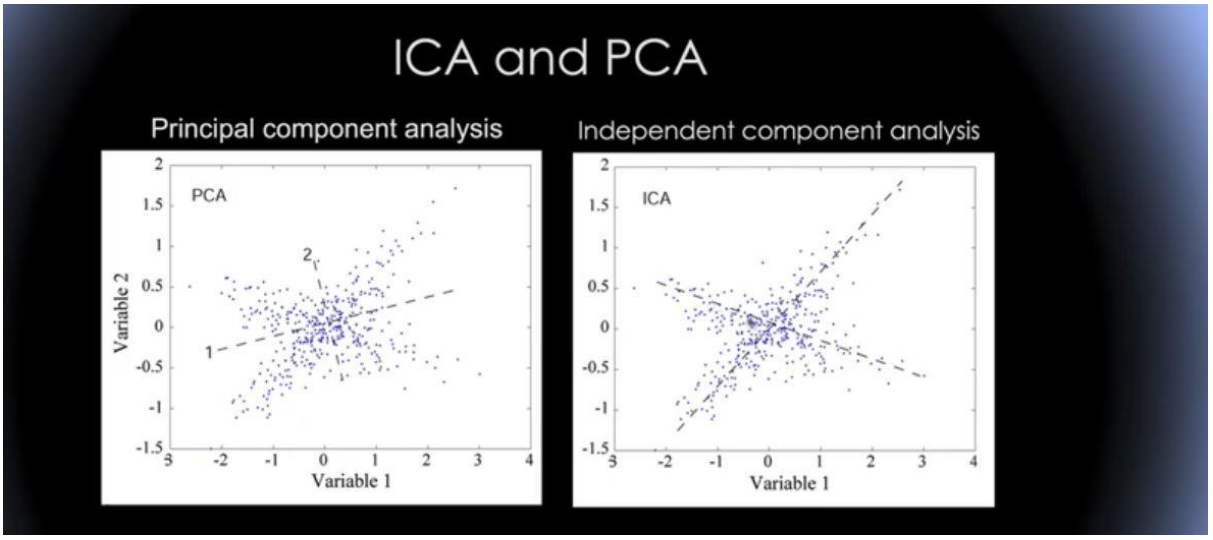


Figure 13 : ICA and PCA principle.

Here we can show the difference in how our data looks like before and after the ICA process

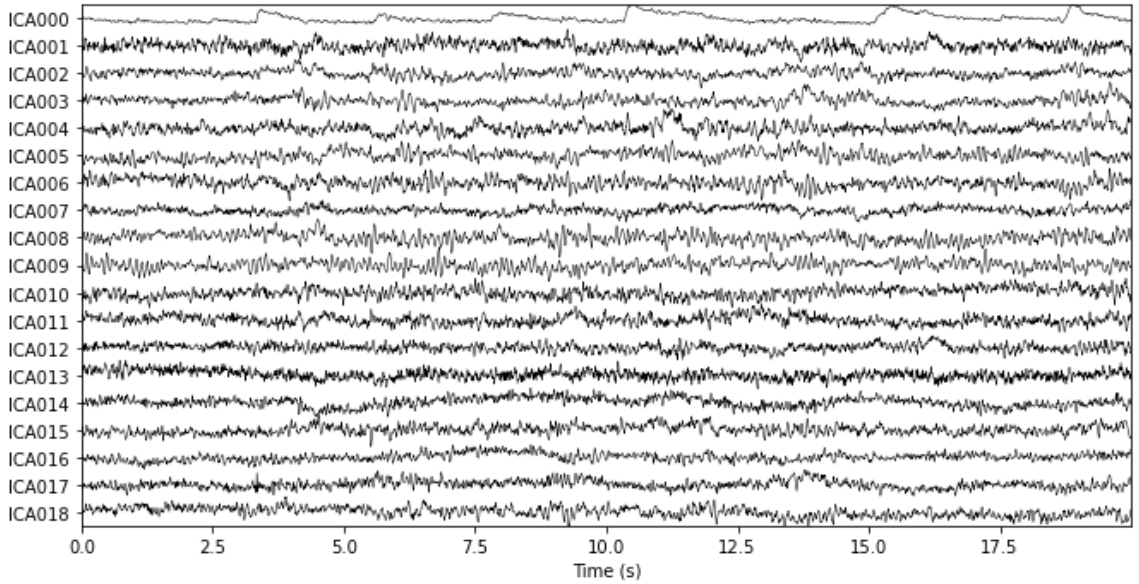


Figure 14 : EEG Signal after the ICA Process

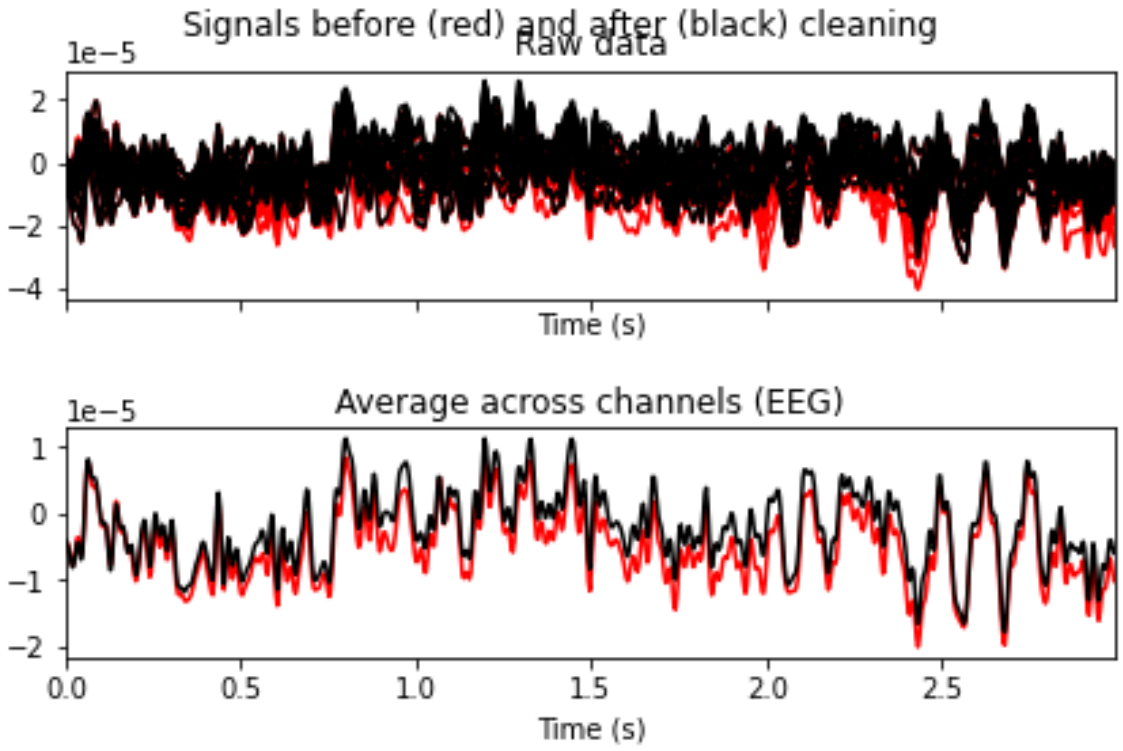


Figure 15: Signal comparison after the processes

Finally, we now have data that is better suited for the models we will use.

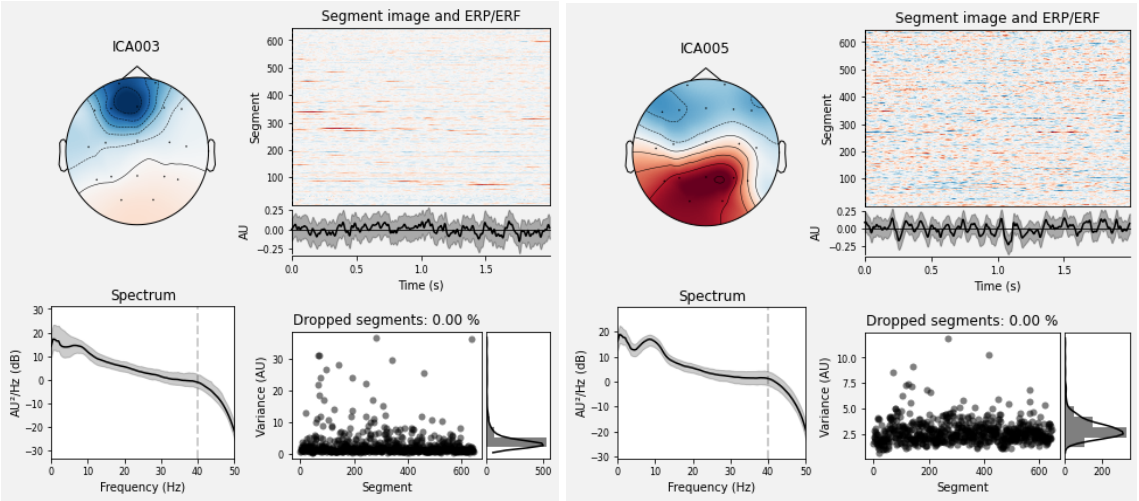


Figure 16 : ICA Samples on 2 electrodes.

4.3.2 Spectrum transformation

We also tried to change our EEGs into specters which in the end helped us have better results for our prediction.

5 Conclusion

Finally, after refining and cleaning our data we can move on to the next step.

III. MODELING & EVALUATION

1 Introduction

The next step in our project is Modeling. We are going to use all the preprocessed data we have collected, and we are going to make models utilizing different algorithms and techniques, such as transfer learning and CNN.

2 Models

2.1 CNN on ICA images

After applying ICA on all the signals, the first step we did was creating a directory where we will store all the images. That will form the train set we will use later.
Our dataset will consist of normalized ICA Images as shown below

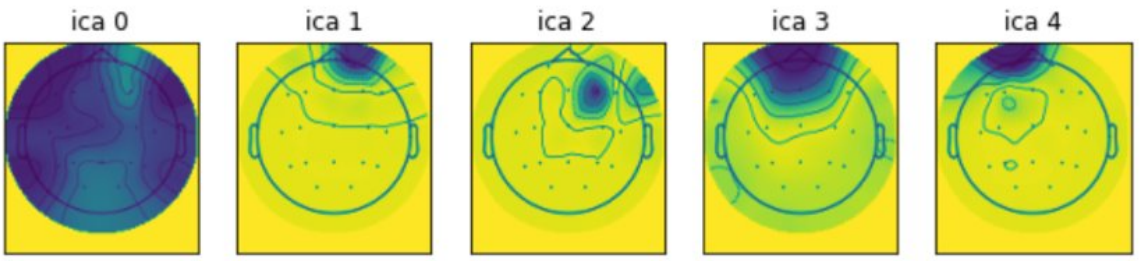


Figure 17 : ICA Images after normalization

After collecting all of our images, we are going to create our Alexnet inspired model, which will consist of 5 convolutional layers and 3 pooling layers.

```
def create_cnn_model(input_img):
  model = Conv2D(96,(3,3),padding='same',input_shape=input_shape)(input_img)
  model = LeakyReLU(alpha=0.1)(model)
  model = MaxPooling2D(pool_size=(2,2),padding='same')(model)
  model = Dropout(0.25)(model)

  model = Conv2D(256,(3,3),padding='same')(model)
  model = LeakyReLU(alpha=0.1)(model)
  model = MaxPooling2D(pool_size=(2,2),padding='same')(model)
  model = Dropout(0.25)(model)

  model = Conv2D(384,(3,3),padding='same')(model)
  model = LeakyReLU(alpha=0.1)(model)
  model = Dropout(0.25)(model)

  model = Conv2D(384,(3,3),padding='same')(model)
  model = LeakyReLU(alpha=0.1)(model)
  model = Dropout(0.25)(model)

  model = Conv2D(256,(3,3),padding='same')(model)
  model = LeakyReLU(alpha=0.1)(model)
  model = MaxPooling2D(pool_size=(2,2),padding='same')(model)
  model = Dropout(0.4)(model)

  return model
```

Figure 18 : CNN Model on ICA Images

After creating the models for each channel, we will have to fuse all of them into one final CNN model.

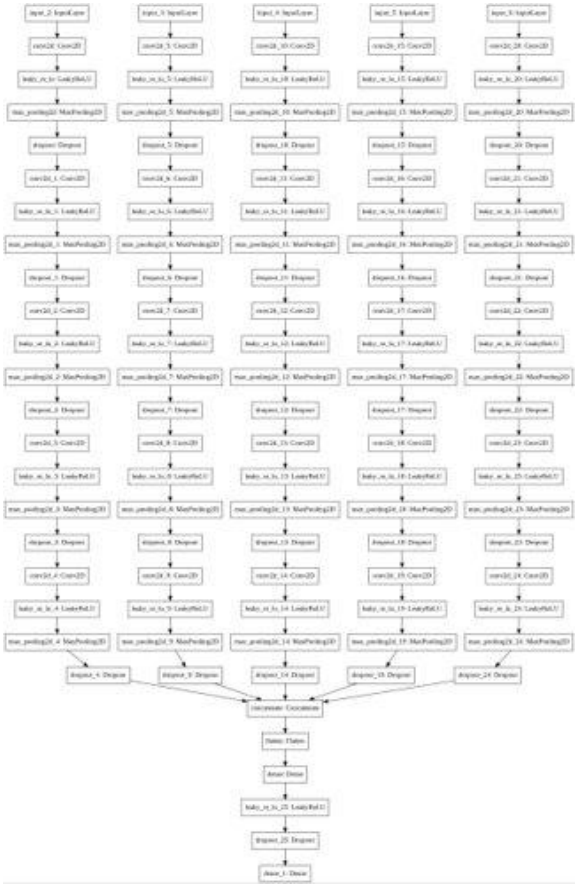


Figure 19 : CNN’s Model Description

Finally, and after splitting our dataset into a training and a test set, we are going to fit the model and then evaluate it through its accuracy

```
history = model.fit_generator(generator_five_img(
    X_train_ica['X_train_ica_0'],X_train_ica['X_train_ica_1'],X_train_ica['X_train_ica_2'],
                X_train_ica['X_train_ica_3'],X_train_ica['X_train_ica_4'],y_train,batch_size
),
          epochs = 10,
          callbacks=callbacks,
          steps_per_epoch = 18,
          verbose=1,
          validation_data=(
              [
                  X_test_ica['X_test_ica_0'],X_test_ica['X_test_ica_1'],X_test_ica['X_test_ica_2'],
                  X_test_ica['X_test_ica_3'],X_test_ica['X_test_ica_4']
              ], y_test),
          shuffle=True)


```

```
C:\Users\gth\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\training.py:1844: UserWarning: `Mode
l.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/10
18/18 [==============================] - 885s 50s/step - loss: 0.8478 - accuracy: 0.5161 - val_loss: 0.7754 - val_accuracy: 0.4
071
```

Figure 20 : Model Training and Evaluation

2.2 Transfer learning on ICA images

2.2.1 Introduction

Transfer learning consists in using a pre-trained model that has already been trained in solving a different problem.
Instead of starting the learning process and building layers from scratch, you start from patterns that have been learned from another task.

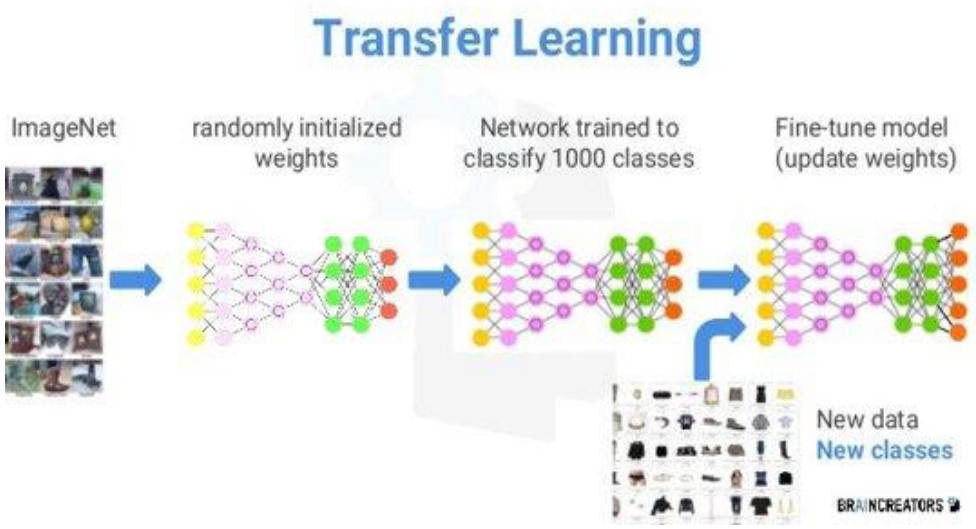


Figure 21 : Transfer Learning Technique explanation

When creating a model for Image classifiers, we take a lot of resources and a lot of time to get the best results possible. Which is why through the use of techniques like transfer learning we can take pre-created models and architectures and try to adapt them to our problem .
In our case we will try and adapt the followings :

- VGG16
- Mxnet
- XCeption

2.2.2 Model Comparison

For the next part we move to the comparison of our pre-trained models and choose the best one of the three.

Pre-trained model name	Accuracy
VGG16	0.5029
Mxnet	0.582
XCeption	0.7905

Table 1 : Transfer Learning Models comparison

2.2.3 Conclusion

Finally, we can conclude that our best pre-trained model is XCeption although it isn't the best result possible and it isn't a conclusive result.

2.3 CNN on spectrums

2.3.1 Model crafting

For this model, we used the Fourier transform in order to obtain spectrograms from the signals we had.

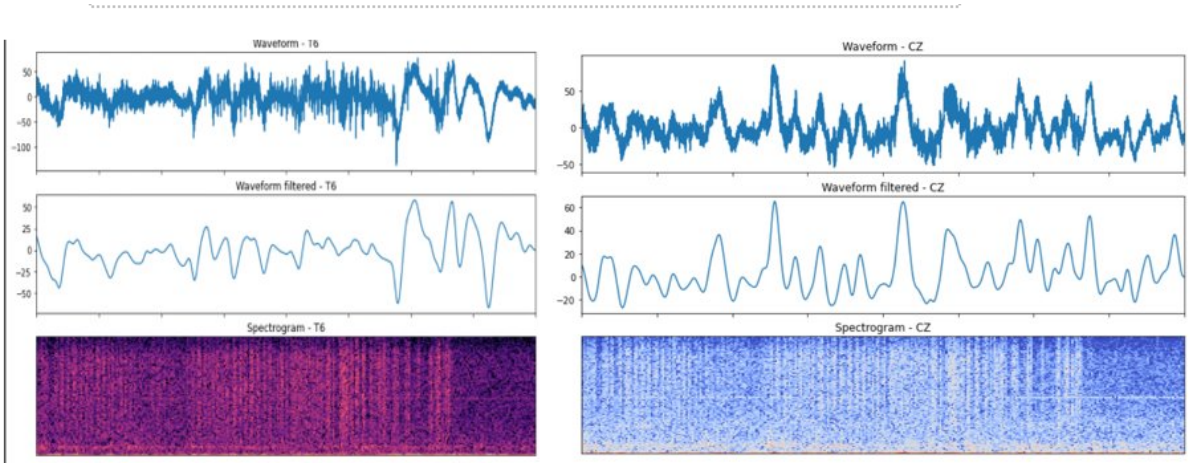


Figure 22 : Signals after Fourier Transform technique

Our data had an uncommon input shape (311,129,19), and similarly to the ICA Image model, we crafted a neural network, as shown below

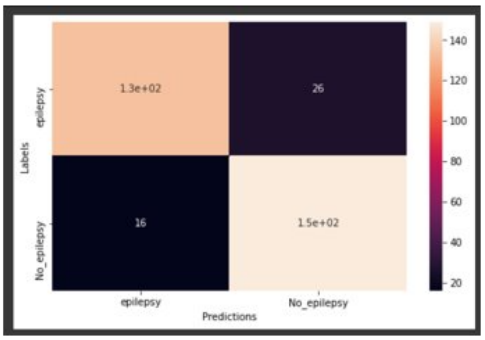
```
model = models.Sequential([
    layers.Input(shape=input_shape),
    norm_layer,
    layers.Conv2D(8, 2, activation='relu'),
    layers.Conv2D(16, 2, activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 2, activation='relu'),
    layers.Conv2D(64, 2, activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 2, activation='relu'),
    layers.Conv2D(64, 2, activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 2, activation='relu'),
    layers.Conv2D(64, 2, activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(6528, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(2304, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(1200, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(500, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(150, activation='relu'),
    layers.Dropout(0.25),
    layers.Dense(1,activation='sigmoid'),
])
```

- (311,129,19) Input_shape
- 8 Convolution layers
- 3 MaxPooling layers
- 6 Dense layers

22

Figure 23 : Fourier Transform's CNN Model

Then we evaluated the model, using the precision, recall and accuracy, and having a look at the confusion matrix to see the performance it had.



- 325 for test

	precision	recall	f1-score	support
epilepsy	0.8933333	0.8375000	0.8645161	160
No_epilepsy	0.8514286	0.9030303	0.8764706	165
accuracy			0.8707692	325
macro avg	0.8723810	0.8702652	0.8704934	325
weighted avg	0.8720586	0.8707692	0.8705853	325

- 87.23% Precision
- 87.02% Recall
- 87.07% Accuracy

Figure 24 : Confusion matrix and evaluation scores

2.3.2 Hyperparameters Tuning :

After evaluating the model, we are looking right now to perfect it and polish it by looking at its hyperparameters.
To do so, we are going to use a commonly used and very effective method : RandomSearch

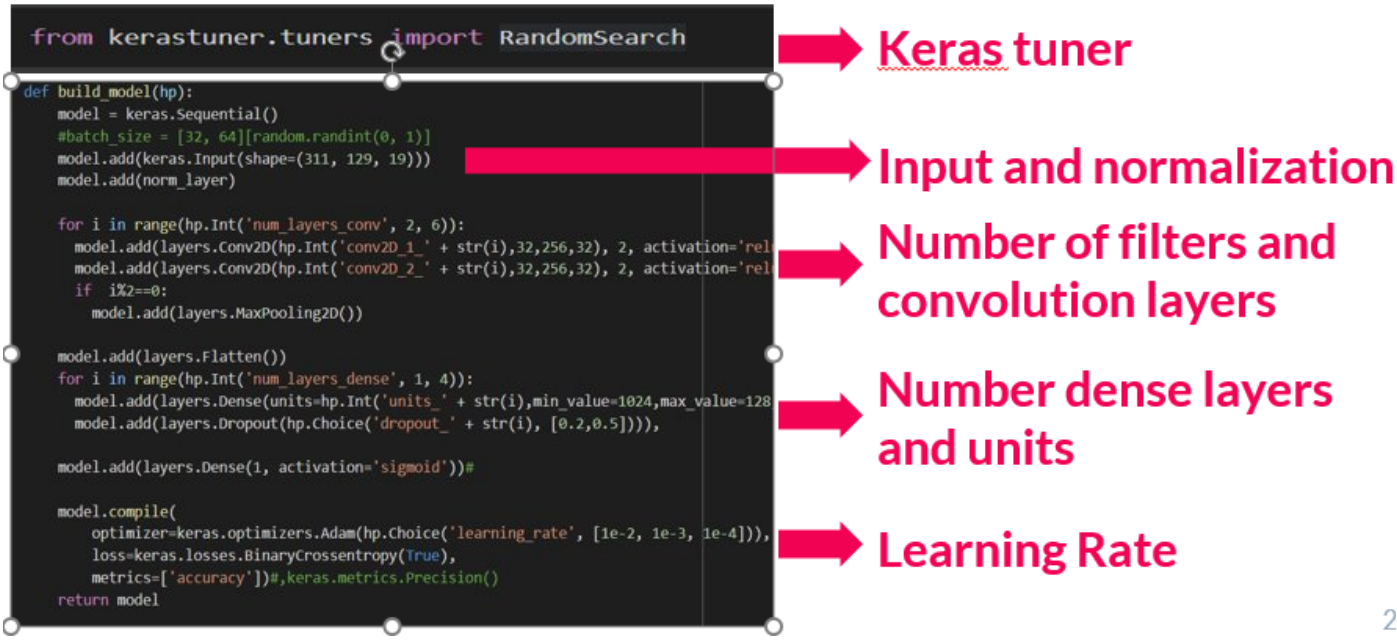


Figure 25 : Hyperparameters Tuning

3 Conclusion

We are going to conclude the modeling part by comparing the performance of the different models we implemented throughout this phase.

Conclusion: Model Comparison

Model	CNN on ICA images	Transfer Learning	Spectrogram based model
Accuracy	0.78	0.7905	0.8723

Table 2 : Final Models Comparison

Based on the accuracy we had, the best model is clearly the Spectrogram based model.

IV. Deployment

1 Introduction

After choosing the best model, we can move on to the last part of our project which is the deployment task which consists of choosing a dashboard available to the user.

2 Features

Our final product will have two types of interfaces :

- Pro : This interface is primarily used by sanitary professionals in this case neurologists. It will give them detailed information as well as accurate prediction.

- Casual : This interface is targeted at the normal users in this case patients. It will only allow them to get an accurate prediction on if they are epileptic or not by inserting their EEG files.

3 Tools & Technologies

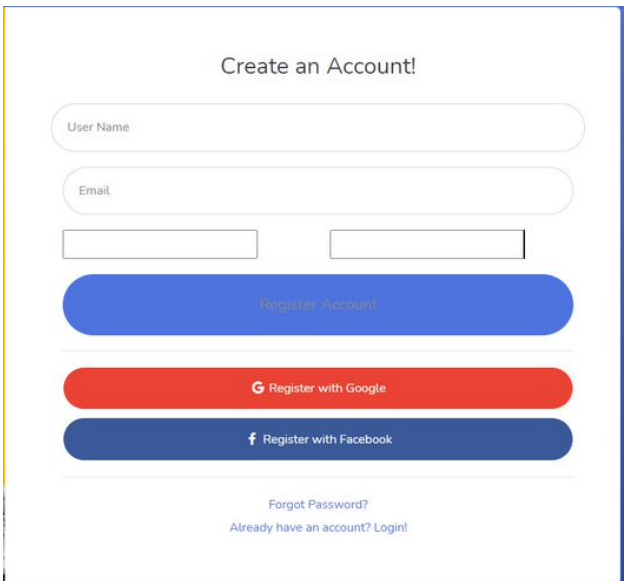
3.1 Python

Figure 26 : Python's Logo

3.2 Django

Figure 27 : Django's Logo

4 Screenshots



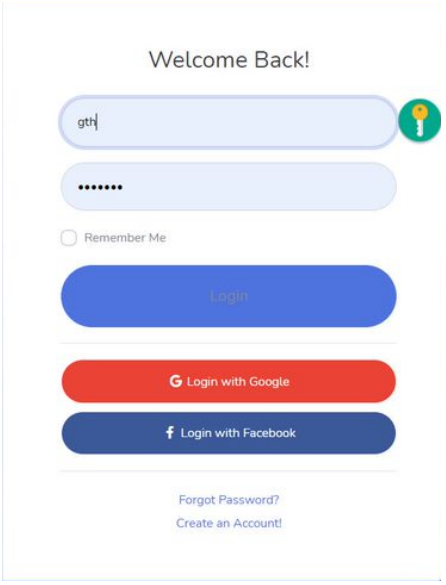


Figure 28: Registration Interface

Figure 29: Login Interface

In our application, the user will use these interfaces to create an account, and then login using their username and password previously created.
After the login process, the customer will be redirected to the home page.

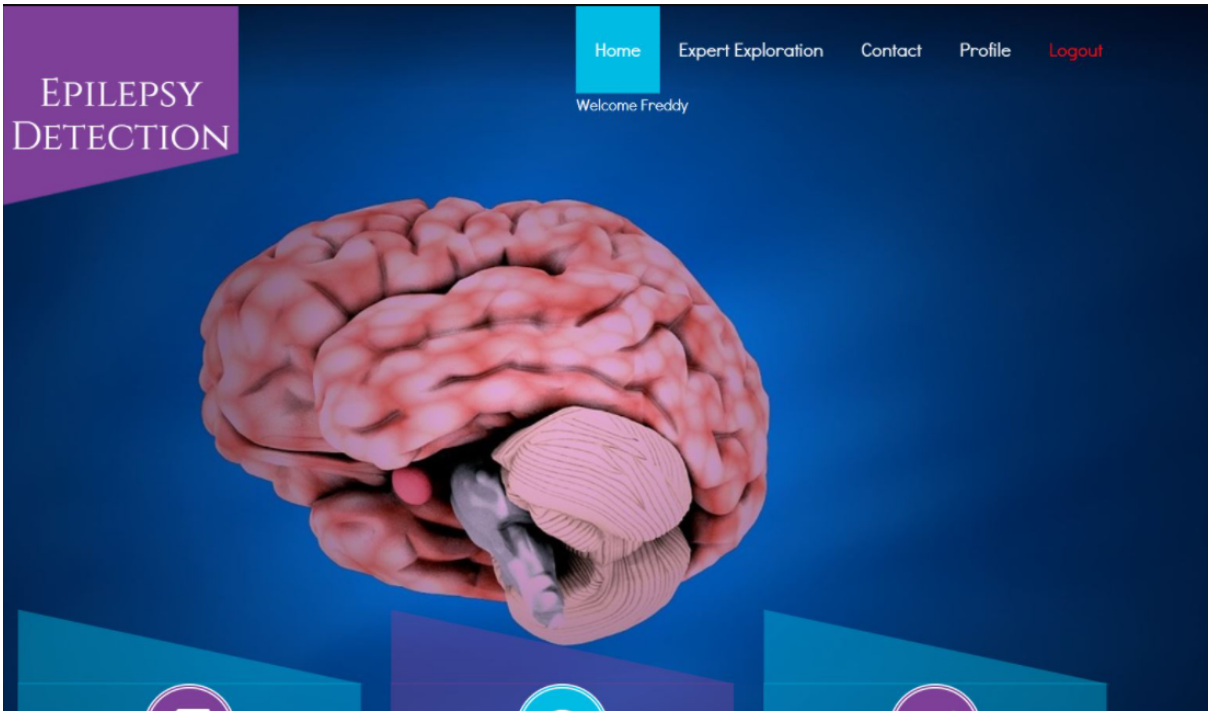


Figure 30 : Home Page Interface

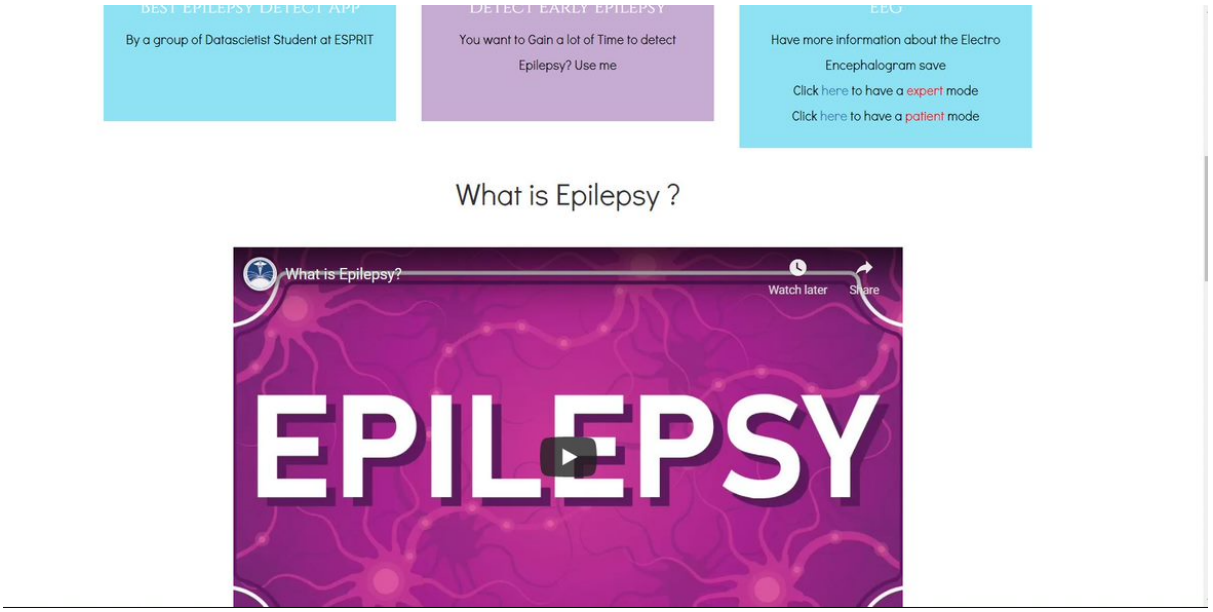


Figure 31: Home Page Interface (2)

As you can see, the user can either purchase the patient mode, which grants him some privileges related to patients , or the

expert mode, which grants him benefits related to doctors, such as analysis and diagnoses.

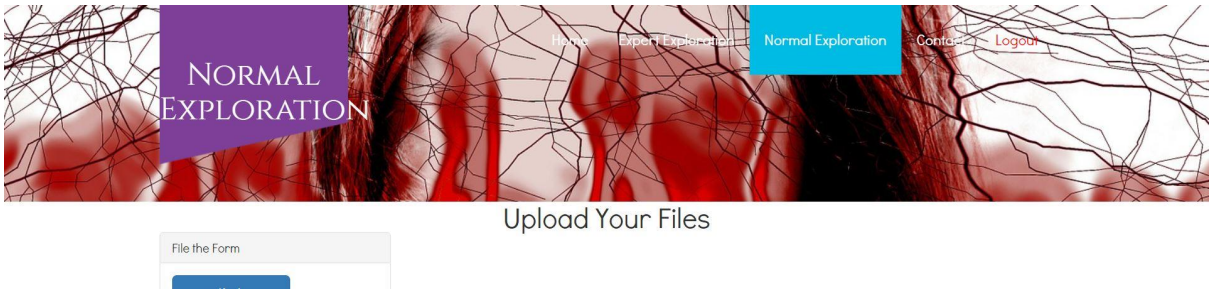


Figure 32 : Patients' File Uploading Page

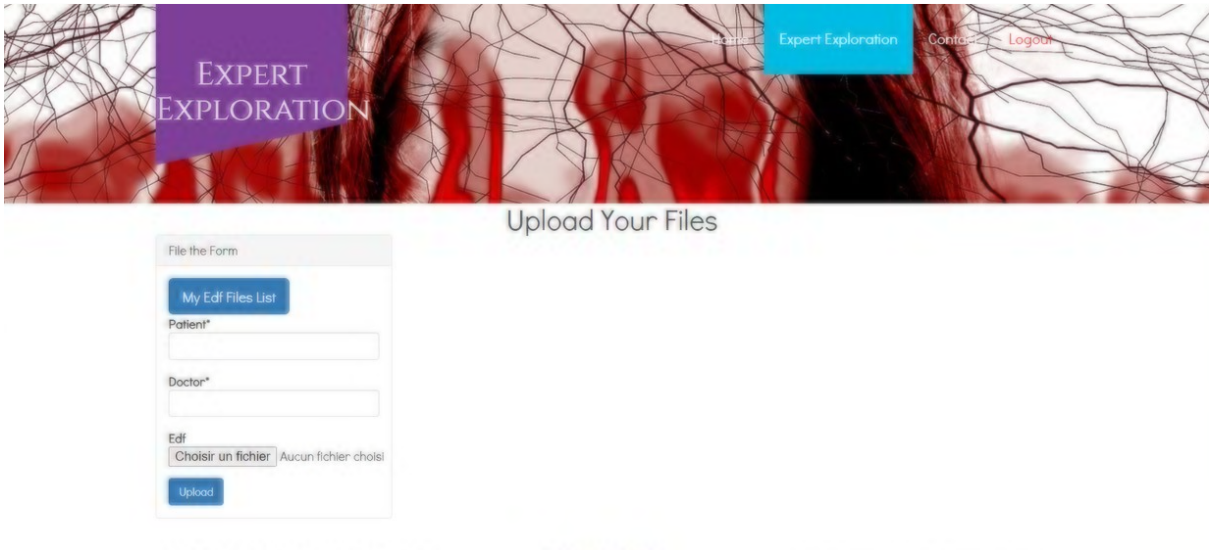


Figure 33 : Experts' File Uploading Page

For experts, they are going to navigate through the Expert Exploration.
Then, in order to use our application, they need to upload their patients files (which needs to be in an .EDF format) and fill the form we have above.

After the uploading process, our algorithms within the application will make all the analysis

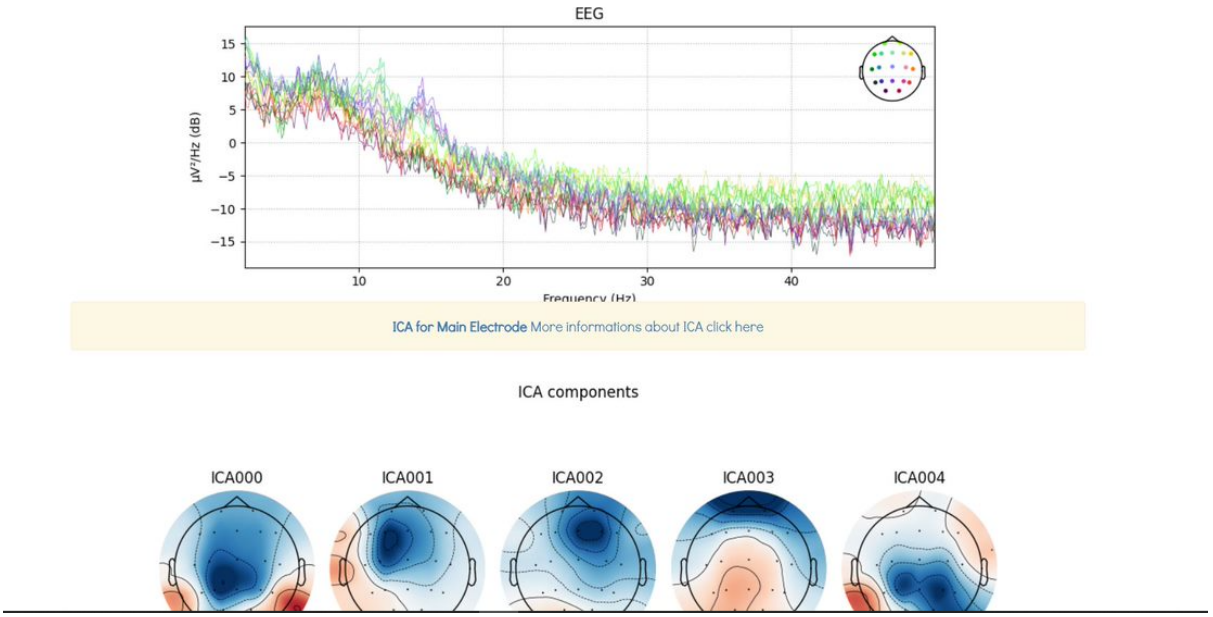


Figure 34 : EDF File analysis (1)

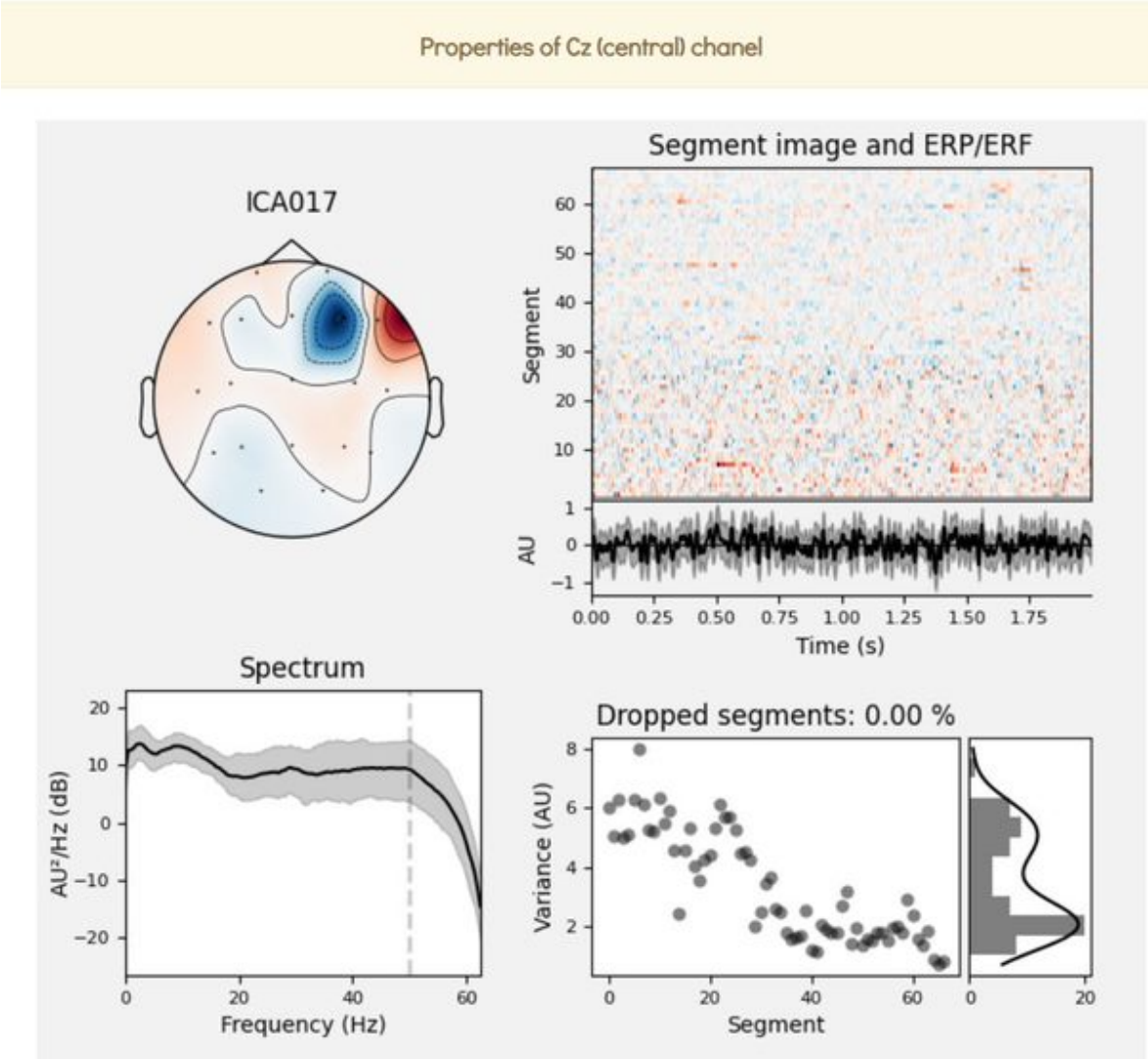


Figure 35 : EDF File Analysis (2)

The doctor will have access to all the analysis, all the graphs, the ICA Components and even more details about the EDF File uploaded.

After that, the expert can have access to the results and the final diagnosis (prediction) about whether the patient is epileptic or not.

Figure 36 : Decision Page Interface

V.Conclusion

The problem that we handled in this project was trying to help doctors with epilepsy using Data Science tools and technologies.
Following the IBM Master Plan Methodology, we managed to design an application that could support them with the signals analysis and the predictions.
Using multiple Machine Learning and Deep Learning techniques such as the Transfer Learning and the CNN, we succeeded in creating a good predictive model that will help the doctors with the exhausting tasks and processes they have to manipulate.

Extended Table of Contents

Report for Data Science Project

Epilepsy Detection

Under the supervision of:

Ncib Lotfi
Mlaouah Ayoub

Written by :

AMORRI Houssein
BEN NEJI Mohamed
DRISS Mohamed Ahmed
GAYAP Hadrien
LONTCHI TABOUA Freddy
SBOUI Mohamed Kadhem