SCHIZOPHRENIA DIAGNOSIS USING BRAINWAVES AND CNN

Project Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

Bonnie Simon

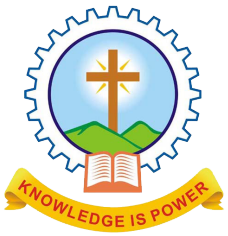
MAC18CS024

Kiran Johns

MAC18CS038

Neenu Chacko

MAC18CS045



Department of Computer Science & Engineering
Mar Athanasius College Of Engineering
Kothamangalam

JUNE 2022

SCHIZOPHRENIA DIAGNOSIS USING BRAINWAVES AND CNN

Project Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

Bonnie Simon

MAC18CS024

Kiran Johns

MAC18CS038

Neenu Chacko

MAC18CS045

Department of Computer Science & Engineering
Mar Athanasius College Of Engineering
Kothamangalam

JUNE 2022

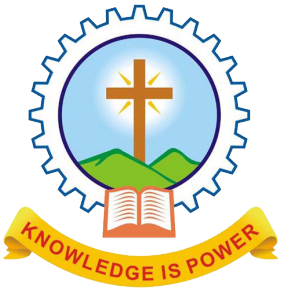
DECLARATION

We, undersigned hereby declare that the project report Schizophrenia Diagnosis using Brainwaves and CNN, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Dr.Surekha Mariam Varghese. This submission represents our ideas in my own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Date: 09.07.2022

Kothamangalam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM



CERTIFICATE

This is to certify that the report entitled **Schizophrenia Diagnosis using Brainwaves and CNN** submitted by **Mr. Bonnie Simon, Reg. No. MAC18CS024, Mr. Kiran Johns, Reg. No. MAC18CS038 and Ms. Neenu Chacko, Reg. No. MAC18CS045**towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering Engineering from APJ Abdul Kalam Technological University for JUNE 2022 is a bonafide record of the project carried out by them under our supervision and guidance.

Dr. Surekha Mariam Varghese	**Prof. Linda Sara Mathew**	**Prof. Joby George**
Project Guide	Project Coordinator	Head of Department

Internal Examiner (s) **External Examiner(s)**

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, We sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the project.

We express our sincere gratitude and thanks to Dr. Bos Mathew Jos, Principal and Prof. Joby George,Head of the Department for providing the necessary facilities and their encouragement and support.

We owe special thanks to my project guide Dr. Surekha Mariam Varghese for her guidance, constant supervision, encouragement and support throughout the period of this project work.

We are grateful to the staff-in-charge Prof. Linda Sara Mathew for her corrections, suggestions and sincere efforts to co-ordinate the project under a tight schedule.

We express our sincere thanks to staff members in the department of Computer Science and Engineering who have taken sincere efforts in helping us to conduct this project.

Finally, We would like to acknowledge the heartfelt efforts, comments, criticisms, cooperation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

Mental illnesses like Schizophrenia are hard to diagnose. The traditional diagnostic process involves interviewing the patient, acquiring their EEG signals visual inspection to identify schizophrenia. This process could yield inaccurate results due to the self reported nature of information from the patient. Furthermore, the complexity in non-linear feature extraction from the time series data makes traditional ML approaches less compatible. Deep learning (DL) techniques overcome the shortcomings of traditional ML approaches like SVM and random forests. DLs such as convolutional neural network (CNN), recurrent neural network (RNN) having long short-term memory network (LSTM) are used.The phase 1 part of the project aims to develop a one dimensional Convolutional Neural Network (CNN) to classify EEG signals from patients and healthy samples. The EEG signals are filtered to remove noise from body movements and other factors. A fast fourier transform (FFT) is applied to extract the five major bands of brain frequencies from the signal. Models with different kernel sizes and filters were experimented with. Classification is then done using a model and results of the classification determine whether the patient has Schizophrenia or not.

CONTENTS

List of Figures

List of Abbrevations

SVM	Support Vector Machines
TP	True Positive
TN	True Negative
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
FN	False Negative
FP	False Positive
KNN	K Nearest neighbors
EEG	Electroencephalogram
ReLU	Rectified Linear Unit
MRI	Root Mean Square Error
LSTM	Long Short Term Memory
EDF	European Data Format
MEG	MagnetoEncephaloGraphy
ECG	Electrocardiography
NIRS	Near-infrared spectroscopy
GRU	Gated Recurrent Units

CHAPTER 1

Introduction

Mental disorders can be hard to diagnose. Doctors can take time from months to years to accurately diagnose a mental illness because the symptoms of most mental disorders overlap. It is estimated that 13% of the world's population suffers from a mental disorder, accounting for approximately 100 crores of people worldwide. In India, almost 7.5% of the population suffers from a mental disorder. Diagnosing mental disorders quickly and efficiently is crucial for treating the disorder.

Schizophrenia is a fatal mental disorder that affects millions of people globally by disturbing their thinking, feelings, and behavior. The current diagnostic process is time consuming and heavily depends on several physical and mental examinations by the psychiatrist. The symptoms of several mental illnesses overlap with each other therefore making the current diagnostic procedures extremely difficult and less accurate. Poor diagnosis of schizophrenia may lead to the patient being untreated and can cause death in some extreme cases.

To speed up the diagnostic procedures we propose a CNN-LSTM model to efficiently classify the recorded EEG signal of a subject as having schizophrenia or not. This method is not only efficient but also speeds up the early diagnostic process of schizophrenia.

However, we need to find an optimal number of convolutional layers and find and combine a proper combination of datasets with healthy subjects and patients having both schizophrenia and other mental illnesses to do an efficient classification.

In the age of the internet of things assisted with cloud computing and machine learning techniques, the computer-aided diagnosis of schizophrenia is essentially required to provide patients with an opportunity to own a better quality of life.

1.1 Schizophrenia

Schizophrenia is a mental disorder in which people interpret reality abnormally. This disorder affects a person's ability to think, feel and behave clearly. The exact cause of schizophrenia is unknown, however scientists hypothesise that a combination of genetic, environmental and change in brain chemistry and brain structure might play a role. Schizophrenia may affect all areas of a person's life including, family, mental, physical, work and education.

Patients usually face stigma, discrimination and violation of their rights. Schizophrenia patients may have the following symptoms. Patients may have delusions and hallucinations which can cause serious physical injury as well as lead to suicide. The patient may also be disorganized when it comes to thinking or doing particular actions. People with schizophrenia often also experience persistent difficulties with their cognitive or thinking skills, such as memory, attention, and problem-solving.

The vast majority of people with schizophrenia around the world are not receiving mental health care. Approximately 50% of people in mental hospitals have a schizophrenia diagnosis. The diagnostic processes are extremely time consuming and mostly provide inaccurate diagnosis moreover only 31.3% of people with schizophrenia receive specialist mental health care.

1.2 Project Outline

The project intends to develop a portable device to assist doctors and other medical professionals in diagnosing mental disorders quickly and accurately. The project involves using an EEG headband to capture the patient's brainwaves. We then use a Raspberry Pi to train and deploy the CNN-LSTM model. The machine-learning algorithm then determines if the patient suffers from Schizophrenia or not. This data would be displayed to the doctor via a custom analytics dashboard. The dashboard will provide live information on the patient's brainwave, what mental disorder they are suffering from, and other patient medical data as needed. This can significantly speed up the diagnosis of schizophrenia, thus making it easier for doctors to treat them.

CHAPTER 2

Background

In this chapter we discuss the necessary technologies and features that we require to build the project.

2.1 Electroencephalogram

An electroencephalogram (EEG) is a test that detects electrical activity in your brain using small, metal discs (electrodes) attached to your scalp. Your brain cells communicate via electrical impulses and are active all the time, even when you're asleep. This activity shows up as wavy lines on an EEG recording.

An EEG is one of the main diagnostic tests for epilepsy. An EEG can also play a role in diagnosing other brain disorders.

An EEG can determine changes in brain activity that might be useful in diagnosing brain disorders, especially epilepsy or another seizure disorder. An EEG might also be helpful for diagnosing or treating the following disorders:

- Brain tumor
- Brain damage from head injury
- Brain dysfunction that can have a variety of causes (encephalopathy)
- Inflammation of the brain (encephalitis)
- Stroke
- Sleep disorders

EEGs are safe and painless. Sometimes seizures are intentionally triggered in people with epilepsy during the test, but appropriate medical care is provided if needed. You'll feel little or no discomfort during an EEG. The electrodes don't transmit any sensations. They just

record your brain waves.After the test, the technician removes the electrodes or cap. If you had no sedative, you should feel no side effects after the procedure, and you can return to your normal routine. Doctors trained to analyze EEGs interpret the recording and send the results to the doctor who ordered the EEG.

2.1.1 EDF File Format

The European Data Format (EDF) is a simple and flexible format for exchange and storage of multi-channel biological and physical signals. It was developed by a few European 'medical' engineers who first met at the 1987 international Sleep Congress in Copenhagen. EDF was published in 1992 in Electroencephalography and Clinical neurophysiology 82, pages 391-393. Since then, EDF became the de-facto standard for EEG and PSG recordings in commercial equipment and multi-center research projects. An extension of EDF, named EDF+, was developed in 2002 and is largely compatible with EDF: all existing EDF viewers also show EDF+ signals. But EDF+ files can also contain interrupted recordings, annotations, stimuli and events. Therefore, EDF+ can store any medical recording such as EMG, Evoked potentials, ECG, as well as automatic and manual analysis results such as delta-plots, QRS parameters and sleep stages. The specs are stricter than EDF which enables automatic localization and calibration of electrodes. And EDF+ fixed a few omissions in EDF such as the Y2K problem, little-endian integers, and comma vs dot.

2.2 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will

increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

2.3 Deep Learning

Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to imitate how humans think and learn.

Artificial neural networks, comprising many layers, drive deep learning. Deep Neural Networks (DNNs) are such types of networks where each layer can perform complex operations such as representation and abstraction that make sense of images, sound, and text.

2.3.1 Applications of Deep Learning

There are several applications of deep learning across industries:

- Self driving cars
- Virtual assistants
- Virtual Recognition
- Natural Language Processing

2.3.2 Advantages of Deep Learning

- The deep learning architecture is flexible to be fitted to new troubles in the future
- It has the proficiency to develop new features from the limited convenient training data sets

2.3.3 Disadvantages of Deep Learning

- Due to complicated data models, it is very expensive to train
- It is not easy to comprehend output based on mere learning and requires classifiers to do so.

2.4 Convolutional Neural Networks

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image. A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

1. Convolution layer - This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.
2. ReLU layer - ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer. ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map.
3. Pooling layer - Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map. The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.
4. Fully connected layer

2.4.1 One Dimensional Convolutional Neural Networks

CNNs are feed-forward Artificial Neural Networks (ANNs) with alternating convolutional and subsampling layers. Deep 2D CNNs with many hidden layers and millions of

parameters have the ability to learn complex objects and patterns providing that they can be trained on a massive size visual database with ground-truth labels. With proper training, this unique ability makes them the primary tool for various engineering applications for 2D signals such as images and video frames. Yet, this may not be a viable option in numerous applications over 1D signals especially when the training data is scarce or application specific. To address this issue, 1D CNNs have recently been proposed and immediately achieved state-of-the-art performance levels in several applications such as personalized biomedical data classification and early diagnosis, structural health monitoring, anomaly detection and identification in power electronics and electrical motor fault detection. Another major advantage is that a real-time and low-cost hardware implementation is feasible due to the simple and compact configuration of 1D CNNs that perform only 1D convolutions (scalar multiplications and additions)

2.5 Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

One reason why python is suitable for machine learning applications is that it has a huge number of libraries and frameworks: The Python language comes with many libraries and frameworks that make coding easy. This also saves a significant amount of time. The most popular libraries are NumPy, which is used for scientific calculations; SciPy for more advanced computations; and scikit, for learning data mining and data analysis. These libraries work alongside powerful frameworks like TensorFlow, CNTK, and Apache Spark. These libraries and frameworks are essential when it comes to machine and deep learning projects.

Also python is a simple language offering reliable code. Machine learning is all about

complicated algorithms and versatile workflows. So the simplicity of Python helps the developers to deal with the complex algorithms. Also, it saves the time of developers as they only require concentrating on solving the ML problems rather than focusing on the technicality of language.

2.6 MNE

Open-source Python package for exploring, visualizing, and analyzing human neurophysiological data: MEG, EEG, sEEG, ECoG, NIRS, etc. The historical core functions of MNE were written by Matti Hämäläinen in Boston and originate in part from the Elekta software that is shipped with its MEG systems. MNE is nowadays developed mostly in Python by an international team of researchers from diverse laboratories and has widened its scope. MNE supports advanced sensor space analyses for EEG, temporal ICA, many different file formats and many other inverse solvers.

2.7 Tensorflow

Tensorflow is an open-source platform for machine learning. Tensorflow consists of a set of flexible tools and libraries that lets developers build and deploy machine learning applications with ease.

Tensorflow offers multiple levels of abstractions to build and train models using the high level Keras API

2.8 Keras

Keras is the most used deep learning framework which provides a high level abstraction to train and develop deep learning models. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code

2.9 Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2.10 Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. It offers data structures and operations for manipulating numerical tables and time series.

2.11 Arduino Mega 2560

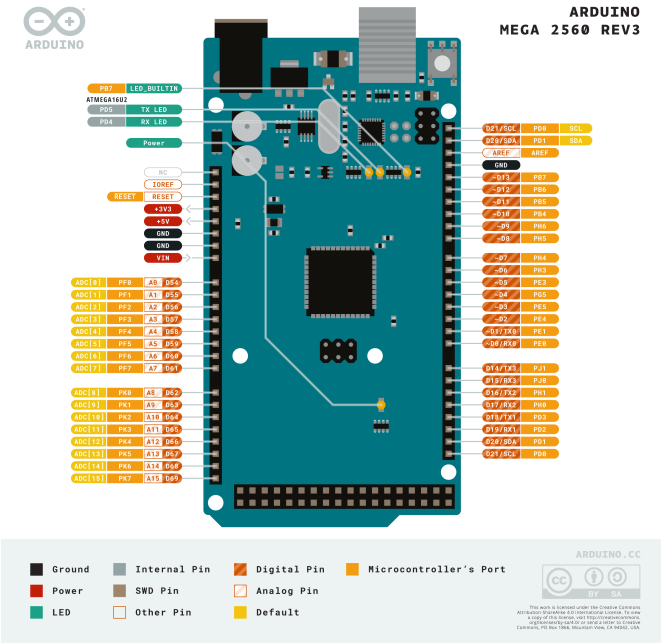


Fig. 2.1: Arduino Mega Pinout Diagram

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP

header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

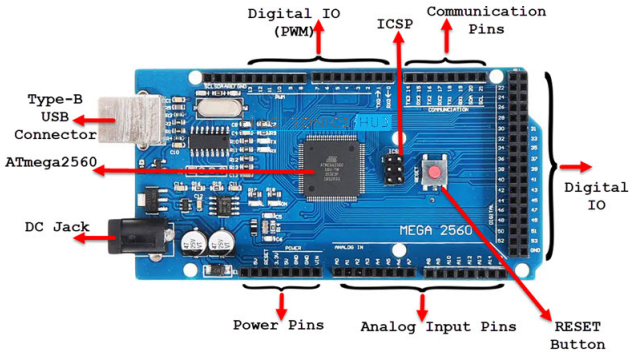


Fig. 2.2: Arduino Mega Layout Diagram

2.11.1 Tech Specs

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage	7-12V
Digital I/O pins	54
Analog pins	16
DC current per I/O pins	20mA
Flash memory	256KB , 8KB used by bootloader
SRAM	8KB
EEPROM	4KB
Clock Speed	16MHz
$\text{LED}_{BUILTIN}$	13
Length	101.52mm
Width	53.3mm
Weight	37gm

2.11.2 Communication Interfaces on Arduino Mega

Arduino Mega supports three different types of communication interfaces. They are:

1. Serial
2. I2C or I2C
3. SPI

Perhaps the most common communication interface in the Arduino universe is the Serial Communication. In fact, the Arduino boards (UNO or Nano or Mega) are programmed using the serial communication.Arduino Mega supports four hardware Serial Communication interfaces. Digital IO pins 0 and 1 are used as Serial RX0 and TX0 pins to receive and transmit serial data. These pins are connected to the serial pins of the on-board USB to Serial Converter IC.

Similarly. Digital IO pins 19 and 18 as RX1 and TX1, 17 and 16 as RX2 and TX2 and 15 and 14 as RX3 and TX3 respectively.Digital IO Pins 20 and 21 can be configured as SDA (20) and SCL (21) to support I2C or I2C or Two Wire Interface (TWI) communication.The final communication interface is the SPI. Digital IO Pins 50, 51 52 and 53 can be configured as SPI pins MISO, MOSI, SCK and SS respectively.

2.11.3 ATMega2560

It's an AVR RISC-based microcontroller that executes powerful instructions in a single clock cycle. This allows it to strike a fine balance between power consumption and processing speed.It can be programmed using Atmel Studio and a dedicated programmer, or using another ATMega microcontroller and using the Arduino development environment. The easiest to use is the Arduino Mega board, which contains the ATMega3560, and can be programmed over USB from the Arduino software. The GPIO pins are also mapped internally to the peripherals like SPI, USART, and SPI. The ATMega2560 uses 5V logic levels, so it might not be compatible with 3.3V sensors and other peripherals. This is not a major limitation, it can be overcome using level shifting circuits. Thanks to its large number of GPIO pins, the ATMega2560 overcomes the limitations of smaller microcontrollers like the ATMega328P and allows it to communicate

with more sensors and peripherals, while also leaving a large number of GPIO pins available for other tasks.

Pin Number	Pin Name	Description
10, 31, 61, 80	V_{CC}	IC Supply pins
11, 32, 62, 81, 99	GND	IC ground reference pins
98	A_{REF}	Reference supply for ADC
100	A_{VCC}	Supply pin for analog peripherals
33, 34	XTAL	Crystal oscillator pins
30	$\overline{RESET}$	Reset pin, active low
2 - 9	PE0 – PE7	GPIO Port E pins
12 – 18, 27	PH0 – PH6, PH7	GPIO Port H pins
19 - 26	PB0 – PB7	GPIO Port B pins
28 – 29, 51 – 52, 70,1	PG3 – PG4, PG0 – PG1, PG2,PG5	GPIO Port G pins
35 - 42	PL0 – PL7	GPIO Port L pins
43 - 50	PD0 – PD7	GPIO Port D pins
53 - 60	PC0 – PC7	GPIO Port C pins
63 – 69, 79	PJ0 – PJ6, PJ7	GPIO Port J pins
71 - 78	PA7 – PA0	GPIO Port A pins
82 - 89	PK7 – PK0	GPIO Port K pins
90 - 97	PF7 – PF0	GPIO Port F pins

Fig. 2.3: ATMega2560 Pinout

2.12 BioAmpEXG Pill

BioAmp EXG Pill is a small, powerful Analog Front End (AFE) biopotential signal-acquisition board that can be paired with any 5 V microcontroller unit (MCU) that has an analog-to-digital converter (ADC), such as an Arduino UNO, an Arduino Nano, or an ESP32 (voltage-divider), among others. It also works with any dedicated 5 V (input range) ADC like the Texas Instruments ADS1115 or ADS131M0x, among others.

BioAmp EXG Pill is capable of recording publication-quality biopotential signals like ECG, EMG, EOG, and EEG, without the inclusion of any dedicated hardware or software filters. Its small size allows easy integration into mobile and space-constrained projects, and its powerful noise rejection makes it usable even when the device is close to the AC mains supply. Any 1.5 mm diameter wire can be used as a strain-relieving electrode cable, making BioAmp

EXG Pill very cost-effective in comparison to other options.

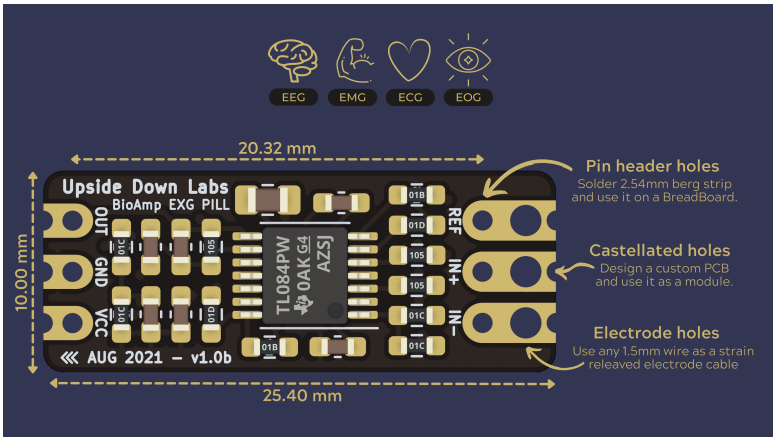


Fig. 2.4: BioAmp EXG Connection

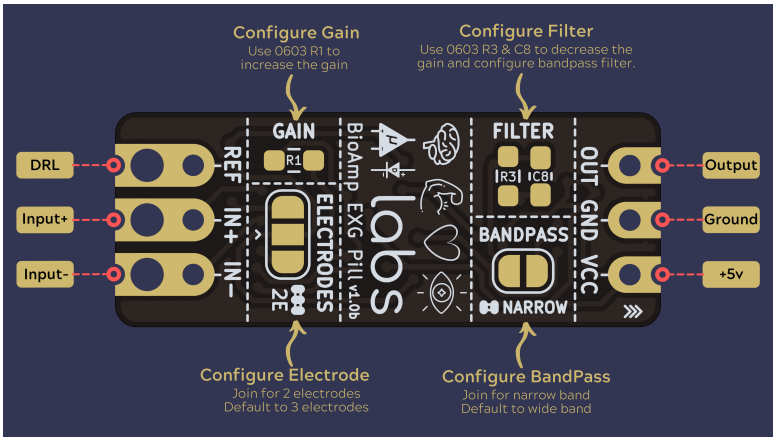


Fig. 2.5: BioAmp EXG Configuration

2.13 Analog to Digital Converter (ADC)

An analog to digital converter (ADC) as its name indicates is an electronic device which converts continuous time-varying analog signals into discrete-time digital signals so that they can easily be read by the digital devices. It has many applications in electronics projects. ADC converts the physical quantities of a real-world phenomenon into a digital language which is used in control systems, data computing, data transmission, and information processing. Mainly there are two steps for the analog to digital conversion:

1. S/H: Sampling and holding

2. Q/E: Quantizing and Encoding

2.13.1 Sampling and Holding

An analog signal continuously changes with time, in order to measure the signal we have to keep it steady for a short duration so that it can be sampled. We could measure the signal repeatedly and very fast, and then find out the right time scale. or we could measure the signal at different timings and then average it. Or preferably we can hold the signal for a specific duration and then digitize the signal and sample the value. This is done by a sample and hold circuit. For, at least the time required for digitization, it keeps the value stable.

2.13.2 Quantizing and Encoding

On the output of (S/H), a certain voltage level is present. We assign a numerical value to it. The nearest value, in correspondence with the amplitude of sampling and holding signal, is searched. And this value cannot be just any value, it should be from a limited set of possible values. It depends on the range of the quantizer and the range given in a power of 2 i,e 2n (28 = 256, 210=1024 etc). After identifying the closest value, a numerical value is assigned to it and it is encoded in the form of a binary number. The binary encoded numbers generated by quantizer are represented by ‘n’ bits. The resolution of an ADC can also be denoted by ‘n’ bit.

2.14 HTML

HTML stands for HyperText Markup Language. It is used to design web pages using the markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages and markup language defines the text document within the tag that define the structure of web pages. Other technologies besides HTML are generally used to describe a web page’s appearance/presentation (CSS) or functionality/behavior (JavaScript). ”Hypertext” refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.HTML uses ”markup” to annotate text, images,

and other content for display in a Web browser. HTML markup includes special ”elements” such as ¡head¿, ¡title¿, ¡body¿, ¡header¿, ¡footer¿, ¡article¿, ¡section¿, ¡p¿, ¡div¿, ¡span¿, ¡img¿, ¡aside¿, ¡audio¿, ¡canvas¿, ¡datalist¿, ¡details¿, ¡embed¿, ¡nav¿, ¡output¿, ¡progress¿, ¡video¿, ¡ul¿, ¡ol¿, ¡li¿ and many others.

An HTML element is set off from other text in a document by ”tags”, which consist of the element name surrounded by ”¡” and ”¿”. The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the ¡title¿ tag can be written as ¡Title¿, ¡TITLE¿, or in any other way. However, the convention and recommended practice is to write tags in lowercase.

2.15 CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, development of various parts of CSS specification was done synchronously, which allowed versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, CSS3. However, CSS4 has never become an official version.

From CSS3, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much that it became more effective to develop and release recommendations separately per module. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest stable state of the CSS specification.

2.16 JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic

language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about APIs that are specific to Web pages, please see Web APIs and DOM.

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402). The JavaScript documentation throughout MDN is based on the latest draft versions of ECMA-262 and ECMA-402.

Do not confuse JavaScript with the Java programming language. Both ”Java” and ”JavaScript” are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

CHAPTER 3

Related Works

Currently, there are different approaches and methodologies to diagnose schizophrenia but most of them are either time consuming or expensive. The conventional approach in diagnosing schizophrenia tends to be inaccurate because the symptoms of most mental illnesses seem to overlap. Below are the detailed explanations of different methods and their drawbacks.

3.1 Conventional Methods

Conventional diagnosis of schizophrenia involves ruling out other mental health disorders and determining that symptoms are not due to substance abuse, medication, or a medical condition. Determining a diagnosis of schizophrenia may include:

- Physical exam: This may be done to help rule out other problems causing symptoms and check for any related complications.
- Tests and screenings: These may include tests that help rule out conditions with similar symptoms, and screening for alcohol and drugs. The doctor may also request imaging studies, such as an MRI or CT scan.
- Psychiatric evaluation: A doctor or mental health professional checks mental status by observing appearance and demeanor and asking about thoughts, moods, delusions, hallucinations, substance use, and potential for violence or suicide. This also includes a discussion of family and personal history.
- Diagnostic criteria for schizophrenia: A doctor or mental health professional may use the criteria in the Diagnostic and Statistical Manual of Mental Disorders (DSM-5) to diagnose the disease.

3.2 Machine Learning & Deep Learning approaches and their drawbacks

Generally, the detection of schizophrenia is carried out using several widely accepted neuroimaging modalities for brain monitoring, which comprises magnetic resonance imaging (MRI), functional MRI (fMRI) and positron emission tomography (PET). These stand-alone techniques or their combinations are used to monitor and diagnose schizophrenic patients.

There have been researches on identifying schizophrenia from fMRI time series data, which investigates a spectrum of similar disorders using neuroimaging-based measures, structural and diffusion MRI based schizophrenia classification using 2D pre-trained and 3D naive Convolutional Neural Networks that can detect microstructural abnormalities in patients with schizophrenia and detection of schizophrenia by co-relating EEG signals and face activity recognition.

But these solutions are constrained by various limitations such as the cost of imaging equipment, high operational skills, and poor quality of images due to artifacts caused by the motion of the patients. And co-relating EEG with face recognition raises privacy concerns related to medical data. Therefore, there exists an urgent need for cost-effective and easy-to-use brain monitoring techniques for the computer-aided diagnosis of schizophrenia. This requirement could be fulfilled by the electroencephalogram (EEG) technique, which can monitor any variation in electrical activities occurring in the patient's brain using non-invasive scalp electrodes. The other significant reasons for preferring the EEG technique over MRI or fMRI methods for brain monitoring of patients with schizophrenia include its high temporal and improved spatial resolutions, ability to directly measure brain activities, the requirement of less technical skills for deployment, and fully or semi- portability. This technique monitors and diagnoses different neurological and mental disorders, comprising epilepsy, Parkinson's disease, depression, and Alzheimer's disease, etc. It has also gained the attention of researchers and neurologists for the diagnosis of schizophrenia.

Traditionally, the acquired EEG signals are analyzed by neurologists through visual inspection to identify schizophrenic patients, which is a slow task and may also lead to errors

in the detection of patients with schizophrenia. Therefore, machine learning (ML) techniques could be applied to make this process automated, which would be beneficial in the precise and speedy identification of patients with schizophrenia. This automated approach could assist the neurologists or experts by providing them second-opinion service for ensuring accurate detection of schizophrenia from EEG signals. Since the acquired EEG signals have a dynamically changing and non-stationary nature, these ML techniques are also accompanied by suitable feature extraction methodologies. But the task of non-linear feature extraction from these signals in time and/or frequency domain is a laborious and complex process. This problem makes traditional ML approaches less compatible in handling big data of EEG signals to detect schizophrenic patients in the real-time scenario.

Nowadays, deep learning (DL) techniques are gaining popularity to overcome the shortcomings of traditional ML approaches. These techniques can handle big data of EEG signals with less processing and classification time, making them suitable for real-time deployment in mobile or hospital scenarios. The popular deep learning algorithms include convolutional neural network (CNN), recurrent neural network (RNN) having long short-term memory network (LSTM) and gated recurrent unit (GRU) variants, and autoencoder models.

CHAPTER 4

Design and Implementation

In the previous chapters we discussed why schizophrenia is an illness that needs to be diagnosed at the earliest and why the diagnostic procedures are very slow. We also saw the advantages that machine learning and deep learning algorithms can provide to accelerate the diagnosis of schizophrenia.

In this chapter we will discuss the architecture of the proposed product to speed up the early diagnosis of schizophrenia, and the functionalities implemented in phase 1 is also discussed below.

The proposed product is a portable device capable of capturing EEG signals from an EEG headband. The deep learning CNN-LSTM algorithm deployed onto the device as an application classifies the incoming EEG signals as healthy or with signs of schizophrenia. The computed result is then displayed on a web dashboard with other relevant patient information for doctors and medical personnel to view. The dashboard will be made available on the network so that doctors and medical personnel can connect to the network to view it. This ensures a better user experience and guarantees that the data is safe and can be later transferred to the hospital database.

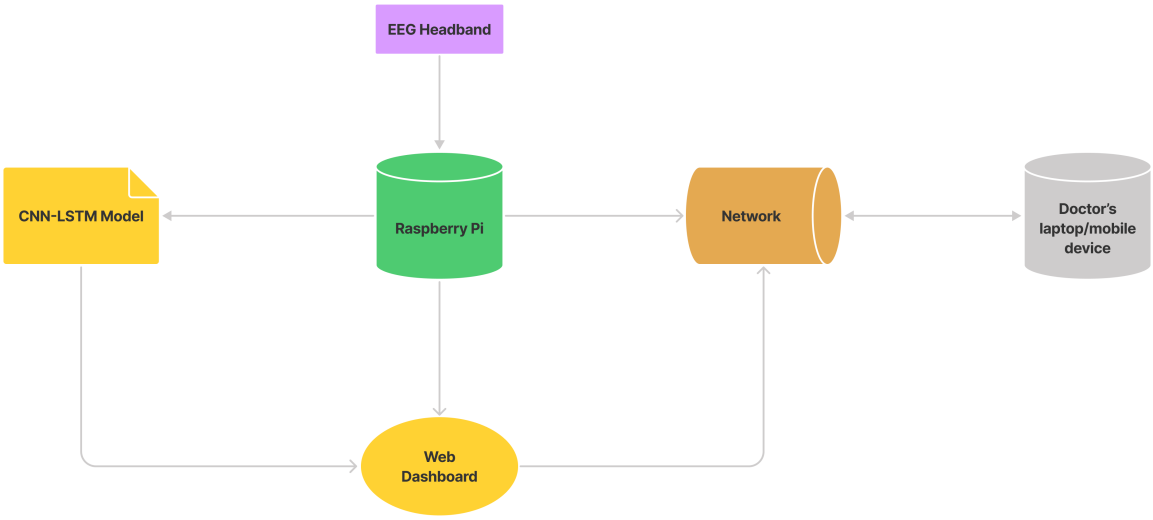


Fig. 4.1: Architecture of the proposed product

The CNN-LSTM model will be responsible for extracting the features from the dataset and classifying the patient and healthy data

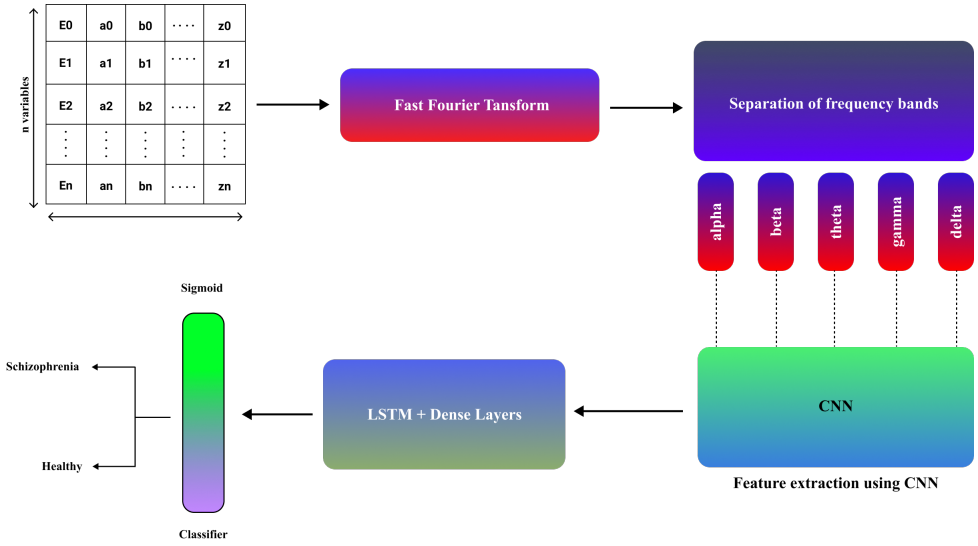


Fig. 4.2: Architecture of the proposed model

4.1 Understanding the dataset and creating a model

Phase 1 of the project included the collection and analysis of two datasets and the development and training of a deep learning model based on convolutional neural networks to classify the illness.

4.1.1 Dataset

The project involved two sets of data Dataset A, consisting of 19 channel EEG data sampled at a frequency of 250 hz from 14 patients and 14 healthy subjects. The data was obtained from the Institute of Biocybernetics and Biomedical Engineering, Poland hosted online at the Repository for Open Data. Dataset B consists of 16 channel EEG data sampled at a frequency of 128 hz from 45 patients and 39 healthy subjects. The data was obtained from Moscow State University hosted online in their EEG Database.

The available datasets A and B being in two different sampling frequencies, different channels and file types, have to be preprocessed separately before combining into a single dataset.

4.1.2 Dataset A - Preprocessing

The dataset A is obtained as an EDF file format and has a sampling rate of 250 hz. The dataset has data from 19 EEG electrode channels [Fp2, F8, T4, T6, O2, Fp1, F7, T3, T5, O1, F4, C4, P4, F3, C3, P3, Fz, Cz, Pz] .The dataset contains samples of 14 healthy and 14 schizophrenia subjects. Each individual dataset has a shape of 19 x (250 * T), where T is the duration of the sampling in seconds.

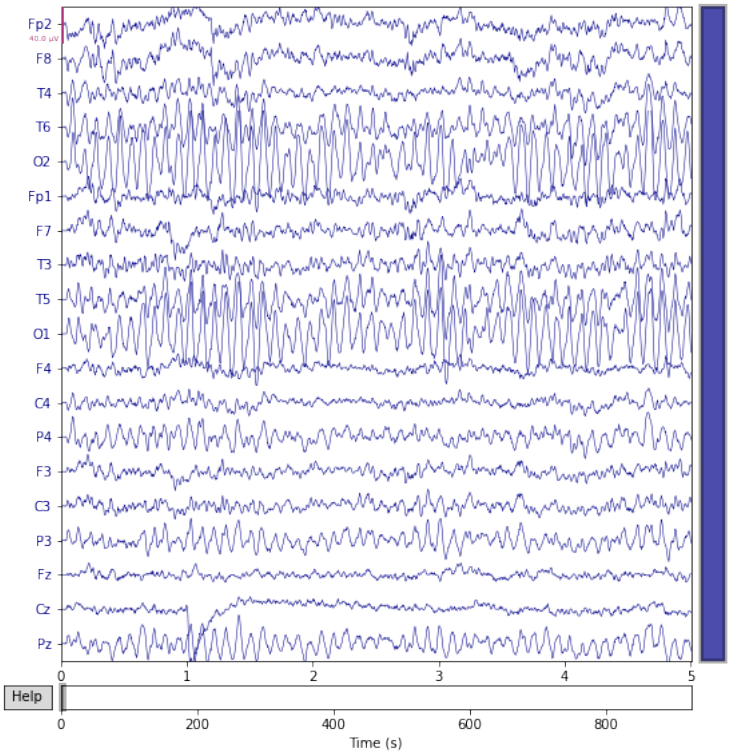


Fig. 4.3: Raw EEG Plot of healthy data

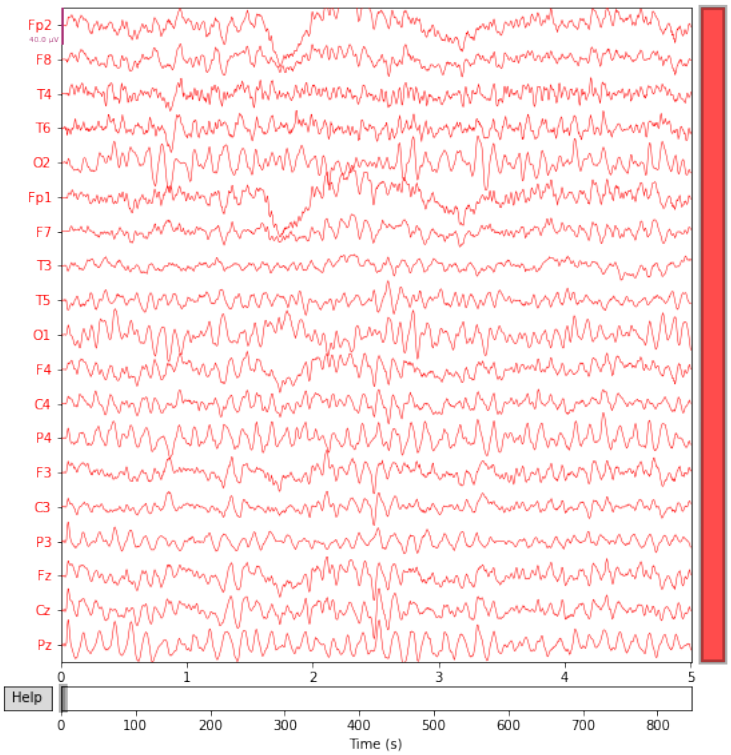


Fig. 4.4: Raw EEG Plot of patient data

The figures above show a plot of a healthy and patient sample of 5 seconds duration. This data has a lot of noise arising due to movement of the body, equipment noise etc. For a proper classification this noise has to be eliminated. The reduction of noise is done by applying a band pass filter over each sample in the dataset.

The preprocessing of EDF files can be achieved by utilising the MNE python package.

```
data=mne.io.read_raw_edf(PATH_TO_EDF_FILE)
```

Once an EDF file is read MNE has a built-in method to apply a filter to reduce the noise here we set the lower bound to a frequency of 0.5 hz and an upper limit of 45 hz.

```
data.filter(l_freq=0.5, h_freq=45)
```

The figures below show the EEG plot of patient and healthy samples after applying the filter to remove the noise.

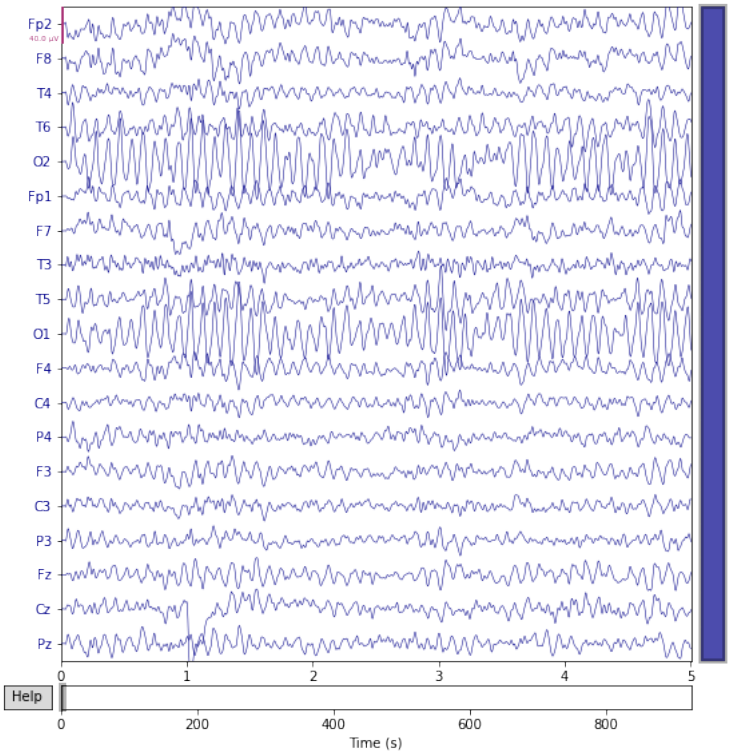


Fig. 4.5: Filtered EEG Plot of healthy data

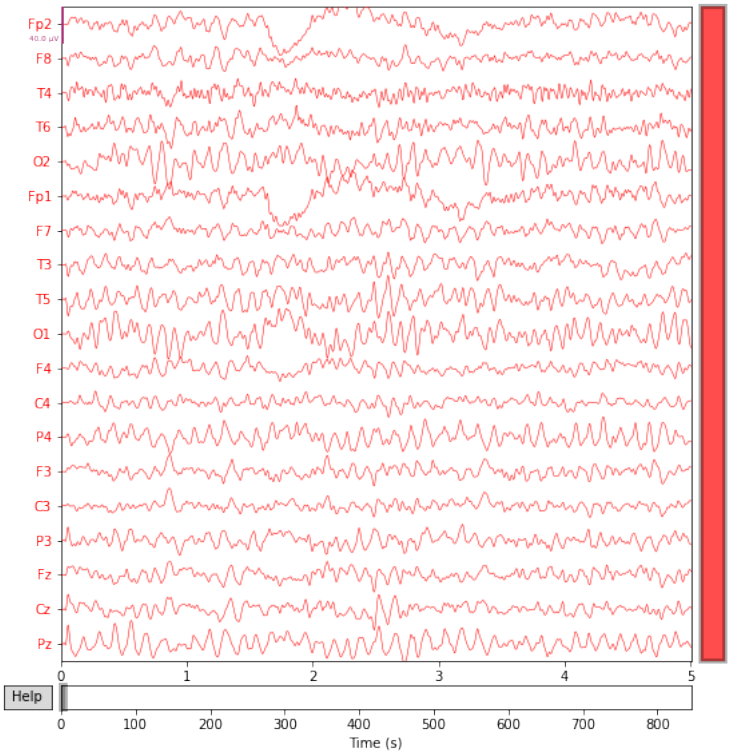


Fig. 4.6: Filtered EEG Plot of patient data

4.1.3 Dataset B - Preprocessing

The Dataset B is obtained an EEA file format, the data is sampled at a frequency of 128 hz from 16 different EEG electrode channels [F7, F3, F4, F8, T3, C3, Cz, C4, T4, T5, P3, Pz, P4, T6, O1, O2]. The dataset contains samples of 39 healthy subjects and 45 patient subjects. Each individual dataset has a dimension of 16 x 7680. Where 16 is the number of electrode channels and 7680 is the value obtained as a result of sampling for 60 seconds (128x60). The EEA file format is not easy to handle as all the EEG data of the patient is stored as a single file of 16x7680 values. The EEA data is converted to CSV data for easy manipulation.

The figure below shows the values of EEG data for 60 seconds obtained from the first five electrode channels.

	0	1	2	3	4	5	6	7	8	9	...	7670	7671	7672	7673	7674	7675	7676	7677	7678
F7	108.01	208.82	388.84	446.45	446.45	349.24	86.41	7.20	7.20	86.41	...	46.81	7.20	-111.61	-190.82	-352.84	-471.65	-532.86	-532.86	-410.44
F3	176.42	237.63	309.63	417.65	439.25	356.44	108.01	46.81	147.62	266.43	...	259.23	237.63	7.20	-151.22	-392.44	-471.65	-550.86	-572.46	-532.86
F4	-90.01	-72.01	-111.61	-162.02	-122.41	-72.01	46.81	86.41	118.81	158.42	...	187.22	79.21	-291.63	-493.25	-500.45	-432.05	-370.84	-493.25	-702.08
F8	7.20	18.00	-82.81	-151.22	-140.42	-32.40	118.81	129.61	28.80	7.20	...	118.81	39.60	-280.83	-410.44	-381.64	-313.23	-331.24	-471.65	-792.09
T3	39.60	129.61	277.23	356.44	309.63	208.82	-10.80	-43.20	7.20	79.21	...	0.00	-32.40	-180.02	-291.63	-471.65	-572.46	-583.26	-550.86	-360.04

Fig. 4.7: CSV EEG Data

The figure below shows a five second duration of a healthy sample and a patient sample from dataset B

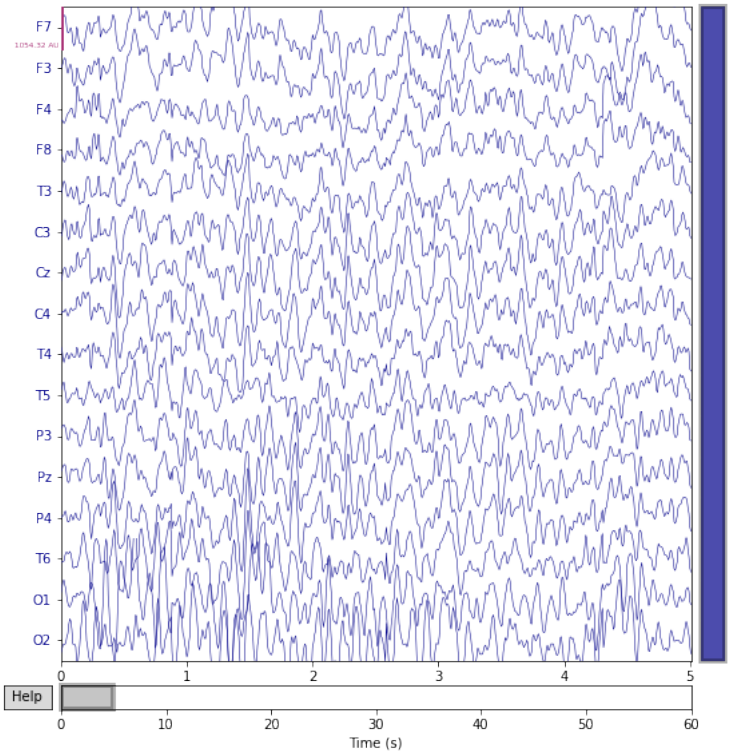


Fig. 4.8: Raw EEG Plot of healthy data

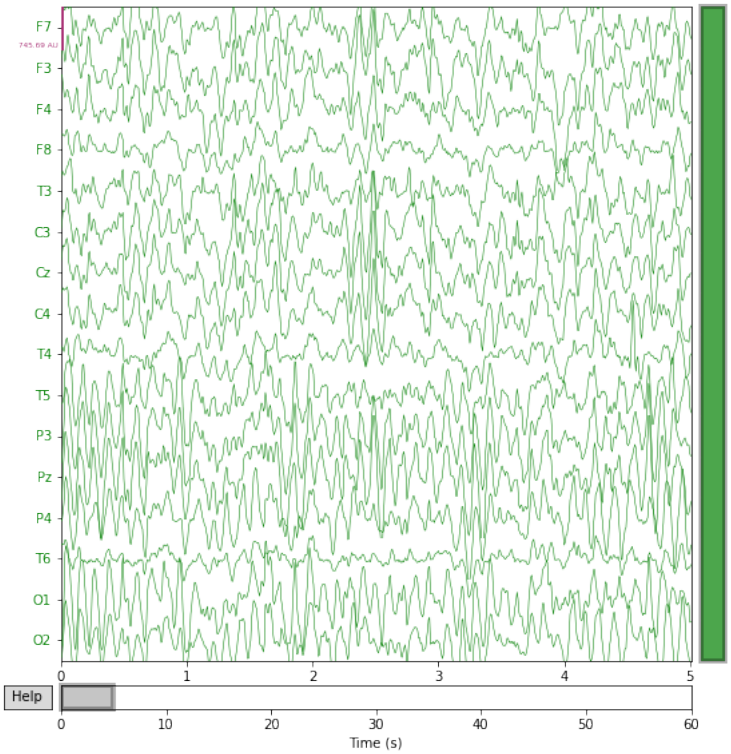


Fig. 4.9: Raw EEG Plot of patient data

Fourier transform is one of the most important concepts in mathematics and signal processing. A fourier transform provides a frequency domain representation of the input signal in it's time domain. Symptoms of schizophrenia are more prominent to distinguish when we take data to frequency domain.

The equation of the fourier transform is as shown below:

$$\mathrm{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{j\omega t} \delta t$$

The data we have in our time domain is to be converted into the frequency domain using the fast fourier transform. The code for implementing a fast fourier transform on our dataset A is as shown below

```
def FFT(df):
    fs = 128 # Sampling rate (128 Hz)
    band_data = pd.DataFrame()
```

```
for index in list(df.index):
    data = df.loc[index]
    fft_vals = np.absolute(np.fft.rfft(data))
    fft_freq = np.fft.rfftfreq(len(data), 1.0/fs)
    eeg_bands = {'Delta': (0, 4),
                 'Theta': (4, 8),
                 'Alpha': (8, 12),
                 'Beta': (12, 30),
                 'Gamma': (30, 45)}

    eeg_band_fft = dict()
    for band in eeg_bands:
        freq_ix = np.where((fft_freq >= eeg_bands[band][0])
        & (fft_freq <= eeg_bands[band][1]))[0]
        eeg_band_fft[band] = np.mean(fft_vals[freq_ix])

    ser = pd.Series(eeg_band_fft,name=index)
    band_data = band_data.append(ser)
return band_data
```

The fast fourier transformation extracts five major bands of brain frequencies namely, delta, theta, alpha, beta and gamma. The delta frequency lies between 0 hz and 4 hz. The theta frequency lies between 4 hz and 8 hz. Alpha between 8 hz and 12 hz. Beta between 12 hz and 30 hz while gamma ranges between 30 hz and 45 hz. The FFT function takes a pandas series as an argument and returns a new series containing the mean amplitude of each frequency band.

	Alpha	Beta	Delta	Gamma	Theta
F7	42692.993502	15103.204440	88473.312110	3880.282604	55976.578362
F3	41736.950926	14636.463205	97836.526504	3876.517404	58941.917438
F4	42715.947709	15751.899840	99231.109599	4339.468244	56509.780705
F8	38186.460041	15248.166183	81601.657846	4414.481169	51678.254634
T3	45563.641902	14794.040704	74862.183478	3750.273170	54752.429378

Fig. 4.10: CSV EEG data in frequency domain

A bar graph of the mean amplitudes of various frequencies are shown below

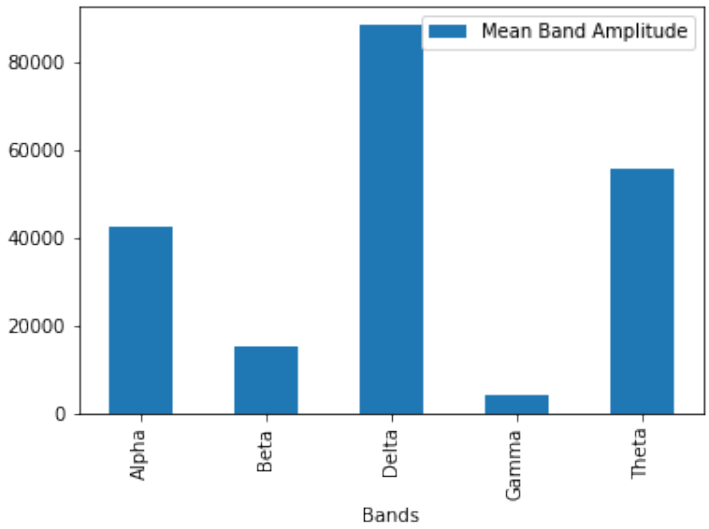


Fig. 4.11: Mean Amplitude v/s Frequency Plot

4.1.4 Convolutional Neural Network

A convolutional neural network is a deep learning algorithm that proves very useful in extraction of features from human activity such as EEG.

Grouping

Prior to developing the model, the rows of each 5 second matrix of the dataset are stacked together and grouped together along with their label. Each segment of the dataset is assigned a label 0 or 1 based on their class. Label 0 indicates that the segment belongs to a healthy subject while 1 indicates that the segment belongs to a patient.

Classification

We apply a done dimensional convolution on dataset A and create a classification model. One of the major challenges is deciding the number of convolutional layers for the model. For the phase 1 implementation, two different convolutional models have been implemented.

Model 1 which has five one dimensional convolutional layers of kernel size 3 and 5 filters is activated using a sigmoid function. The details of entire layers of the model are shown below

CNN Model 1		
Layer (type)	Output Shape	Param
Conv1D	None, 1248, 5	290
BatchNormalization	None, 1248, 5	20
LeakyReLU	None, 1248, 5	0
MaxPooling1D	None, 624, 5	0
Conv1D	None, 622, 5	80
LeakyReLU	None, 622, 5	0
MaxPooling1D	None, 311, 5	0
Dropout	None, 311, 5	0
Conv1D	None, 309, 5	80
LeakyReLU	None, 309, 5	0
AveragePooling1D	None, 154, 5	0
Dropout	None, 154, 5	0
Conv1D	None, 152, 5	80
LeakyReLU	None, 152, 5	0
AveragePooling1D	None, 76, 5	0
Conv1D	None, 74, 5	80
LeakyReLU	None, 74, 5	0
GobalAveragePooling1D	None, 5	0
Dense	None, 1	6

The above model has been implemented using tensorflow

```
def CNN_Model():
    clear_session()
    model = Sequential()
    model.add(Conv1D(filters=5,kernel_size=3,
                strides=1,input_shape=(1250,19))) #1
    model.add(BatchNormalization())
    model.add(LeakyReLU())
    model.add(MaxPool1D(pool_size=2, strides=2))#2
```

```
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#3
model.add(LeakyReLU())
model.add(MaxPool1D(pool_size=2,strides=2))#4
model.add(Dropout(0.5))
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#5
model.add(LeakyReLU())
model.add(AveragePooling1D(pool_size=2,strides=2))#6
model.add(Dropout(0.5))
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#7
model.add(LeakyReLU())
model.add(AveragePooling1D(pool_size=2,strides=2))#8
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#9
model.add(LeakyReLU())
model.add(GlobalAveragePooling1D())#10
model.add(Dense(1,activation='sigmoid'))#11

model.compile('adam',loss='binary_crossentropy',
                metrics=['accuracy'])
return model
```

The Model 1 implemented returned a training accuracy of 83% and a validation accuracy of 67%.

Model 2 which has one one dimensional convolutional layers of kernel size 3 and 32 filters is activated using a sigmoid function. The details of entire layers of the model are shown below

CNN Model 1		
Layer (type)	Output Shape	Param
Conv1D	None, 1248, 32	1856
BatchNormalization	None, 1248, 32	128
MaxPooling1D	None, 624, 32	0
GlobalAveragePooling 1D	None, 32	0
Dense	None, 1	33

The Model 2 was implemented using tensorflow

```
def CNN_Model_2():
  clear_session()
  model = Sequential()
  model.add(Conv1D(filters=32,kernel_size=3,
            strides=1,input_shape=(1250,19))) #1
  model.add(BatchNormalization())
  model.add(MaxPool1D(pool_size=2, strides=2))#2
  model.add(GlobalAveragePooling1D())#10
  model.add(Dense(1,activation='sigmoid'))
  model.compile('adam',loss='binary_crossentropy',
                metrics=['accuracy'])
  return model
```

The model 2 returned a training accuracy of 67% and a validation accuracy of 69%

4.2 CNN-LSTM Model

The CNN-LSTM Model below returned an accuracy of 87.90% while training and 73.89% during validation

CNN-LSTM Model 1		
Layer (type)	Output Shape	Param
conv1d	None, 1248, 5	290
layer normalization	None, 1248, 5	10
leaky relu	None, 1248, 5	0
max pooling1d	None, 624, 5	0
conv1d	None, 622, 5	80
leaky relu	None, 622, 5	0
max pooling1d None, 311, 5	0	
dropout	None, 311, 5	0
conv1d	None, 309, 5	80
leaky relu	None, 309, 5	0
average pooling	None, 154, 5	0
dropout	None, 154, 5	0
conv1d	None, 152, 5	80
leaky relu	None, 152, 5	0
average pooling1d	None, 76, 5	0
conv1d	None, 74, 5	80
dropout	None, 74, 5	0
lstm	None, 100	2400
leaky relu	None, 100	0
dense	None, 1	101

The above model has been implemented using tensorflow

```
def CNN_LSTM_Model():
  clear_session()
  model = Sequential()
  model.add(Conv1D(filters=5,kernel_size=3,strides=1,input_shape=(1250,19
  model.add(LayerNormalization())
  model.add(LeakyReLU())
```

```
model.add(MaxPool1D(pool_size=2, strides=2))#2
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#3
model.add(LeakyReLU())
model.add(MaxPool1D(pool_size=2,strides=2))#4
model.add(Dropout(0.5))
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#5
model.add(LeakyReLU())
model.add(AveragePooling1D(pool_size=2,strides=2))#6
model.add(Dropout(0.5))
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#7
model.add(LeakyReLU())
model.add(AveragePooling1D(pool_size=2,strides=2))#8
model.add(Conv1D(filters=5,kernel_size=3,strides=1))#9
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=False))

model.add(LeakyReLU())
# model.add(GlobalAveragePooling1D())#10
model.add(Dense(1,activation='sigmoid'))#11

model.compile('adam',loss='binary_crossentropy',metrics=['accuracy'])
return model
```

4.3 FFT-CNN-LSTM Model

A fourier transform applied to the CNN-LSTM model resulted in a training accuracy of 94% and on validation provided 75.7%

FFT-CNN-LSTM Model 1		
Layer (type)	Output Shape	Param
conv1d	None, 4, 64	320
layer normalization	None, 4, 64	128
leaky relu	None, 4, 64	0
max pooling1d	None, 2, 64	0
conv1d	None, 1, 64	8256
leaky relu	None, 1, 64	0
max pooling1d	None, 1, 64	0
dropout	None, 1, 64	0
average pooling1d	None, 1, 64	0
dropout	None, 1, 64	0
lstm	None, 100	66000
leaky relu	None, 100	0
dense	None, 100	10100
dense	None, 1	101

The model has been implemented using tensorflow as below

```
def FFT(df):
    fs = 250 # Sampling rate (128 Hz)
    band_data = pd.DataFrame()
    for index in range(1):
        data = df[index]  # 2 sec of data b/w 0.0-100.0

        # Get real amplitudes of FFT (only in postive frequencies)
        fft_vals = np.absolute(np.fft.rfft(data))
        # Get frequencies for amplitudes in Hz
        fft_freq = np.fft.rfftfreq(len(data), 1.0/fs)
        # Define EEG bands
```

```
        eeg_bands = {'Delta': (0, 4),
                     'Theta': (4, 8),
                     'Alpha': (8, 12),
                     'Beta': (12, 30),
                     'Gamma': (30, 45)}

        # Take the mean of the fft amplitude for each EEG band
        eeg_band_fft = dict()
        for band in eeg_bands:
            freq_ix = np.where((fft_freq >= eeg_bands[band][0])
            (fft_freq <= eeg_bands[band][1]))[0]
            eeg_band_fft[band] = np.mean(fft_vals[freq_ix])

        ser = pd.Series(eeg_band_fft,name=index)
        band_data = band_data.append(ser)
    return band_data.to_numpy()

def FFT_CNN_LSTM_Model():
 clear_session()
 model = Sequential()
 model.add(Conv1D(filters=64,kernel_size=2,
 strides=1,input_shape=(5,2))) #1
 model.add(LayerNormalization())
 model.add(LeakyReLU())
 model.add(MaxPool1D(pool_size=1, strides=2))#2
 model.add(Conv1D(filters=64,kernel_size=2,strides=1))#3
 model.add(LeakyReLU())
 model.add(MaxPool1D(pool_size=1,strides=2))#4
 model.add(Dropout(0.5))
 model.add(AveragePooling1D(pool_size=1,strides=1))#6
 model.add(Dropout(0.2))
```

```
model.add(LSTM(100, return_sequences=False))

model.add(LeakyReLU())
model.add(Dense(100))
model.add(Dense(1,activation='sigmoid'))#11

model.compile('adam',loss='binary_crossentropy',
metrics=['accuracy'])
return model
```

4.4 EEG Headset Design

The BioAmp EXG pill is powered by an Arduino Mega 2560 Microcontroller to set up a circuit of a fourth order Band-Pass Butterworth IIR digital filter, with sampling rate 256 Hz and a frequency of 0.5 - 29.5 Hz. It takes input from the frontal cortex of the human head, which is in position to the standard EEG Headset electrodes Fp1 and Fp2 with the third electrode used for noise cancellation. The electrodes detect tiny electrical charges that result from the activity of your brain cells which are then amplified to appear on the computer screen. It is typically non-invasive, with the electrodes placed along the scalp.

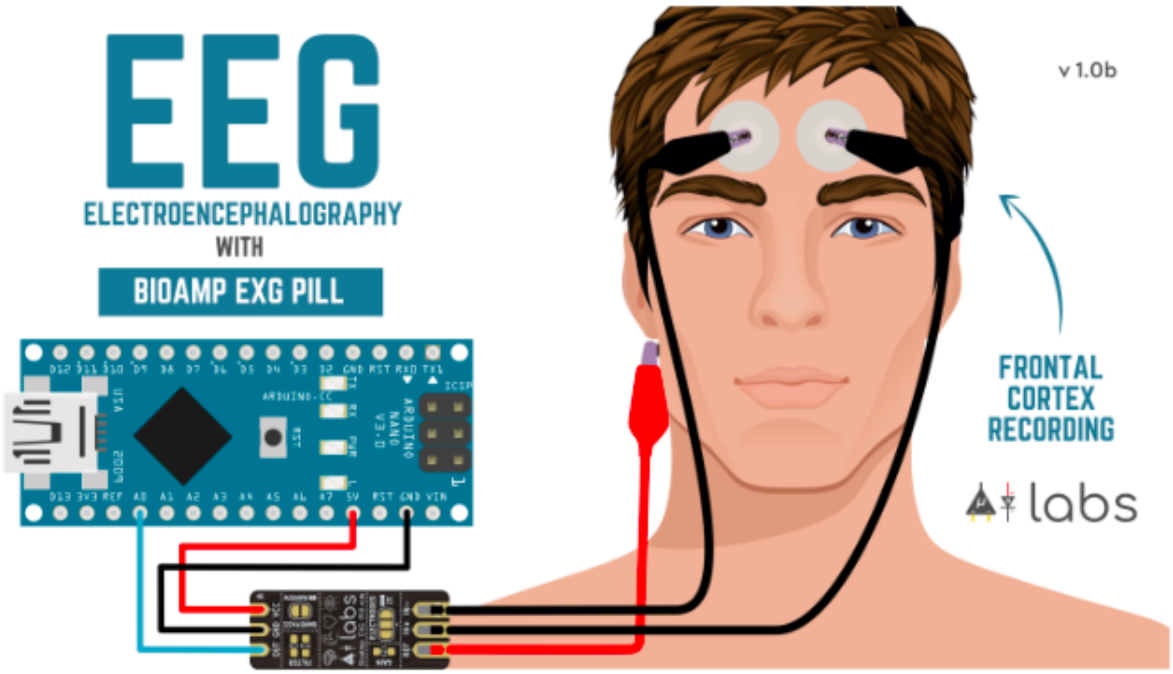


Fig. 4.12: EEG Acquisition Circuit using BioAmp EXG Pill

4.4.1 Serial Plotter Output

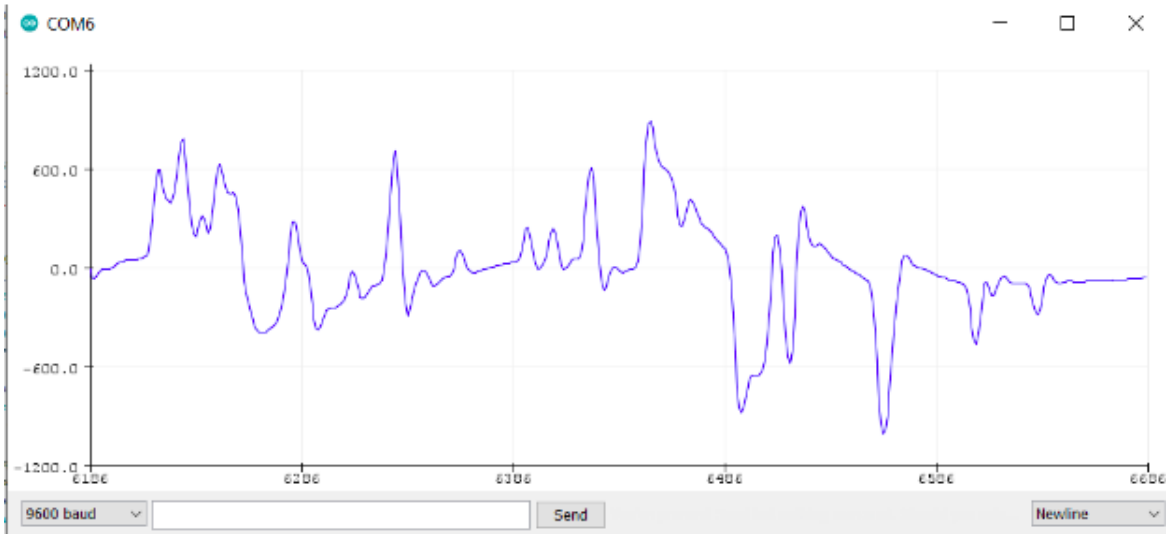


Fig. 4.13: EEG Signal Plot

4.5 Two Electrode Model

The machine learning model that was previously implemented cannot be integrated with our custom EEG headset to detect the disease. This is because the model we implemented utilizes 19 electrode channels whereas our custom EEG device only has two channels. Therefore we need to reduce the number of channels that are given as input to the model.

We train the model using the data available from Fp1 and Fp2 electrodes so that integration of the model and the hardware is possible. Here we only use a CNN-LSTM model with the following layers to train the model.

Two Electrode CNN-LSTM Model 1		
Layer (type)	Output Shape	Param
conv1d	None, 1998, 5	35
layer normalization	None, 1998, 5	10
max pooling 1d	None, 999, 5	0
conv1d	None, 997, 5	80
dropout	None, 498, 5	0
conv1d	None, 496, 5	80
average pooling 1d	None, 248, 5	0
dropout	None, 248, 5	0
conv1d	None, 246, 5	80
average pooling 1d	None, 123, 5	0
conv1d	None, 121, 5	80
dropout	None, 121, 5	0
lstm	None, 100	42400
leaky relu	None, 100	0
dense	None, 100	10100
dense	None, 1	101

The model produced an accuracy of 77.8% while training and 73.7% while using a validation dataset.

The model implemented using tensorflow

```
def CNN_LSTM_2E_Model():
 clear_session()
 model = Sequential()
 model.add(Conv1D(filters=5,kernel_size=3,strides=1,
 input_shape=(2000,2))) #1
 model.add(LayerNormalization())
 model.add(MaxPool1D(pool_size=2, strides=2))#2
 model.add(Conv1D(filters=5,kernel_size=3,strides=1))#3
 model.add(MaxPool1D(pool_size=2,strides=2))#4
 model.add(Dropout(0.25))
 model.add(Conv1D(filters=5,kernel_size=3,strides=1))#5
 model.add(AveragePooling1D(pool_size=2,strides=2))#6
 model.add(Dropout(0.25))
 model.add(Conv1D(filters=5,kernel_size=3,strides=1))#7
 model.add(AveragePooling1D(pool_size=2,strides=2))#8
 model.add(Conv1D(filters=5,kernel_size=3,strides=1))#9
 model.add(Dropout(0.1))
 model.add(LSTM(100, return_sequences=False))
 model.add(LeakyReLU())
 model.add(Dense(100))#11
 model.add(Dense(1,activation='sigmoid'))#11

 model.compile('adam',loss='binary_crossentropy',metrics=['accuracy'])
 return model
```

4.6 Dashboard

Dashboard acts as a visual representation, for easy interpretation by medical personnel. The dashboard mainly serves few purposes. First, it acts as the frontend for the user input and the data capture. Also, it shows a graphical representation of EEG signal for quick analysis by medical personnels. Adding to that, it also displays the output of the model by sending the

input data to a backend using REST.

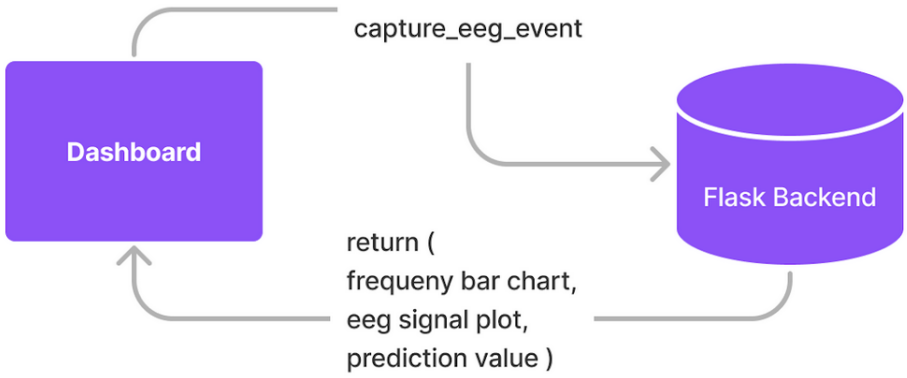


Fig. 4.14: Dashboard Data Flow

The EEG data is uploaded as a log file. Then we send a request to the backend along with the EEG data. The response will be the output of the model which will be displayed on the screen.

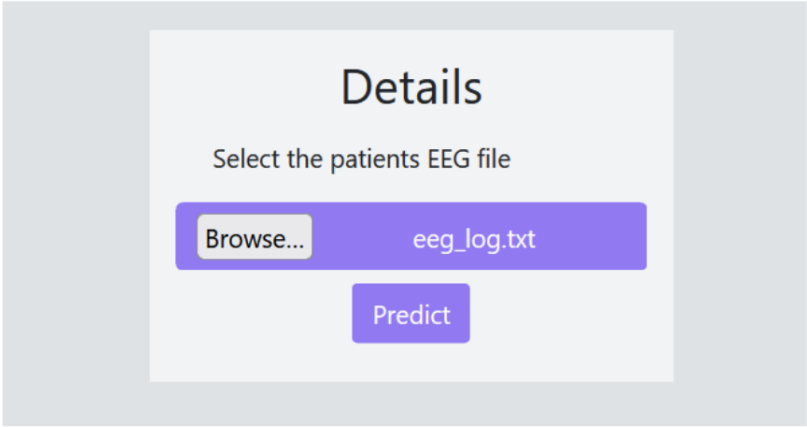


Fig. 4.15: Dashboard File Selector

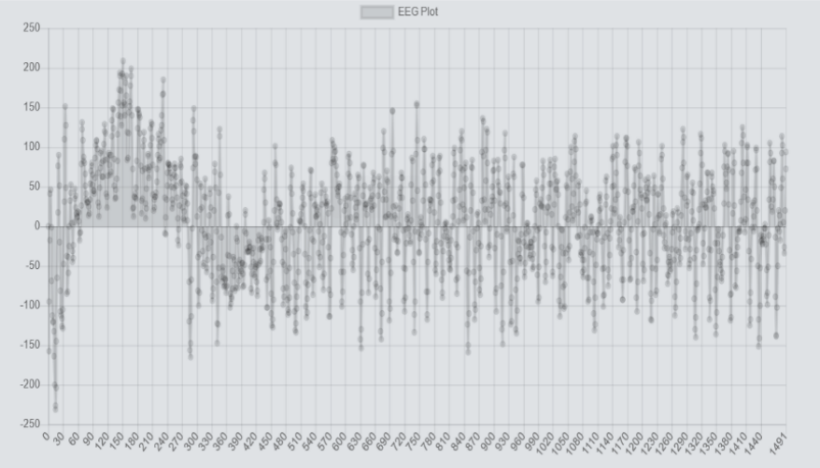


Fig. 4.16: Dashboard EEG Plot

CHAPTER 5
Evaluation and Analysis

5.1 Confusion Matrix

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. A confusion matrix is a tabular summary of the number of correct and incorrect predictions made by a classifier. It is used to measure the performance of a classification model. It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score. The following 4 are the basic terminology which will help us in determining the metrics we are looking for.

- True Positives (TP): when the actual value is Positive and predicted is also Positive.
- True negatives (TN): when the actual value is Negative and prediction is also Negative.
- False positives (FP): When the actual is negative but prediction is Positive. Also known as the Type 1 error
- False negatives (FN): When the actual is Positive but the prediction is Negative. Also known as the Type 2 error

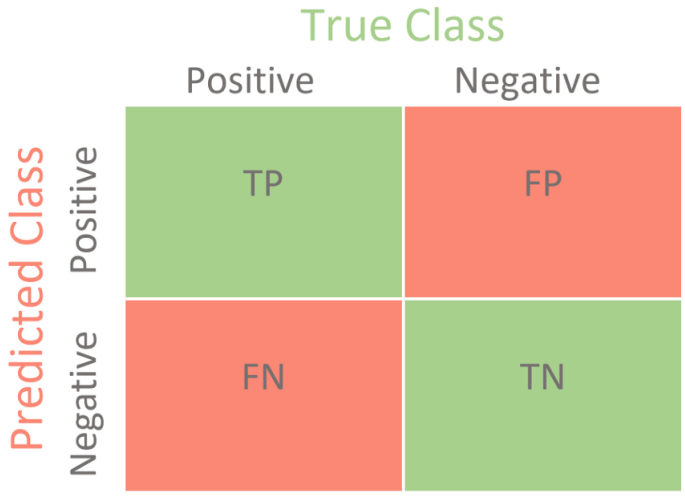


Fig. 5.1: Confusion Matrix

In our implementation, the 1D CNN which has five one dimensional convolutional layers of kernel size 3 and 5 filters, activated using a sigmoid function had a training accuracy of 83% and a validation accuracy of 67%.

Also the one 1D convolutional layer with kernel size 3 and 32 filters, activated using a sigmoid function had a training accuracy of 67% and a validation accuracy of 69%.

The CNN-LSTM model four convolutional layers with kernel size 3 and filter count 5, activating using a sigmoid function produced a training accuracy og 87.9% and a validation accuracy of 73.89%.

The FFT-CNN-LSTM model two convolutional layers with kernel size 2 and filter count 64, activating using a sigmoid function produced a training accuracy og 94% and a validation accuracy of 75.7%.

The Two Electrode CNN-LSTM model four convolutional layers with kernel size 3 and filter count 5, activating using a sigmoid function produced a training accuracy og 77.8% and a validation accuracy of 73.7%.

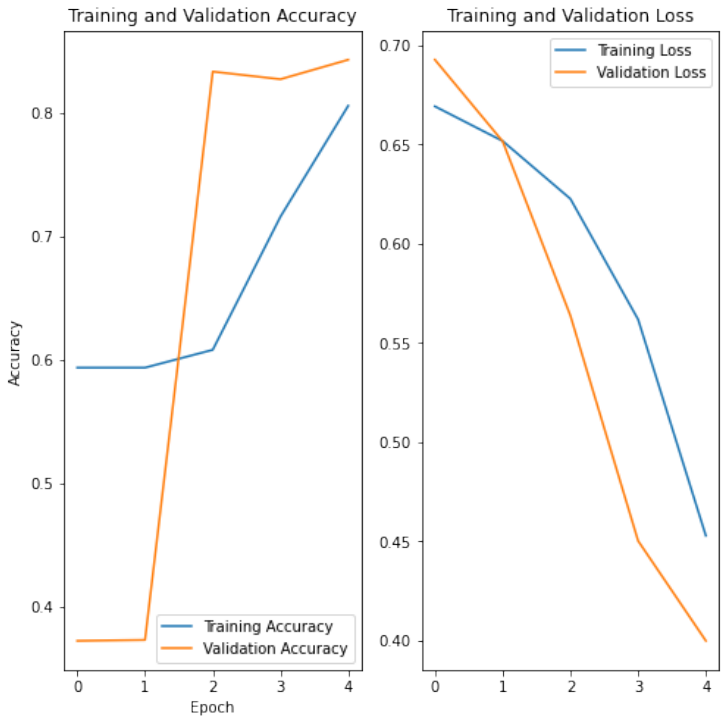


Fig. 5.2: 5 layer CNN

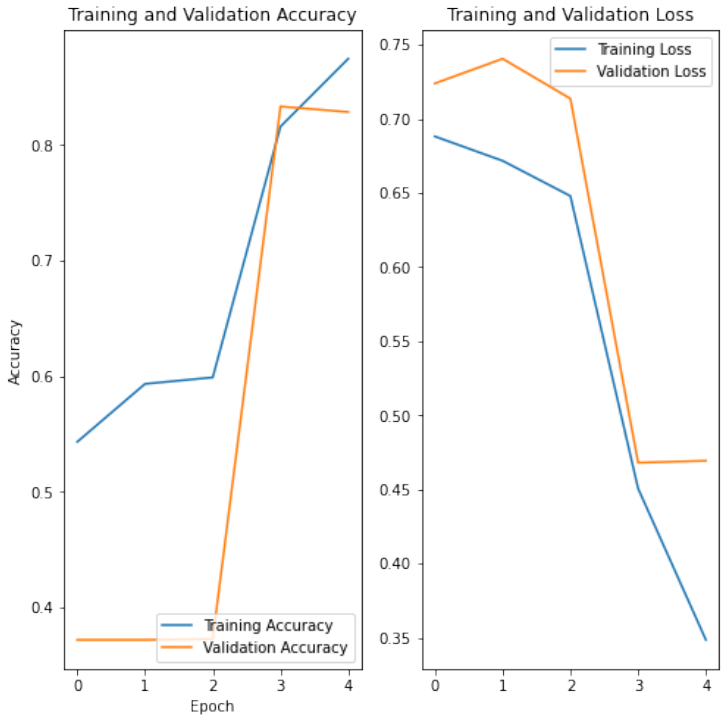


Fig. 5.3: 1 layer CNN

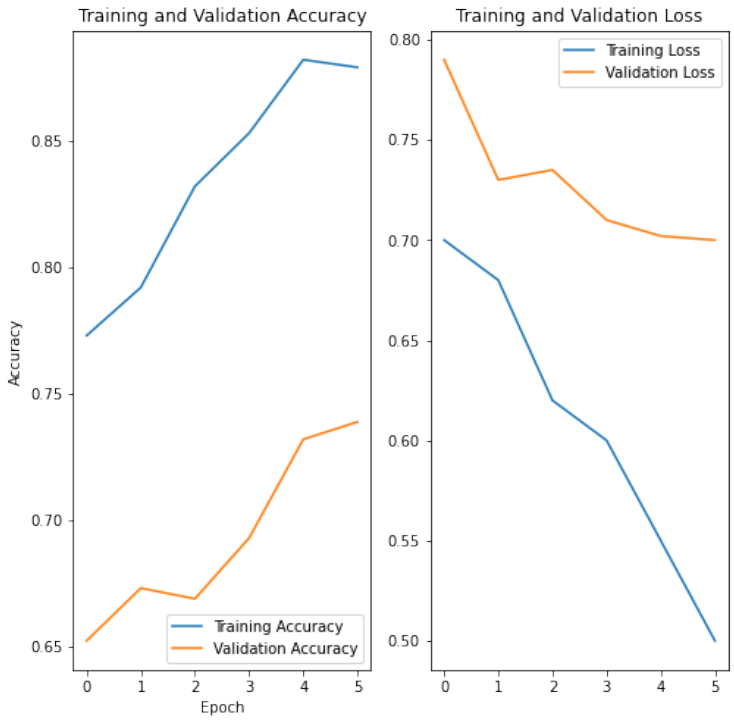


Fig. 5.4: CNN LSTM

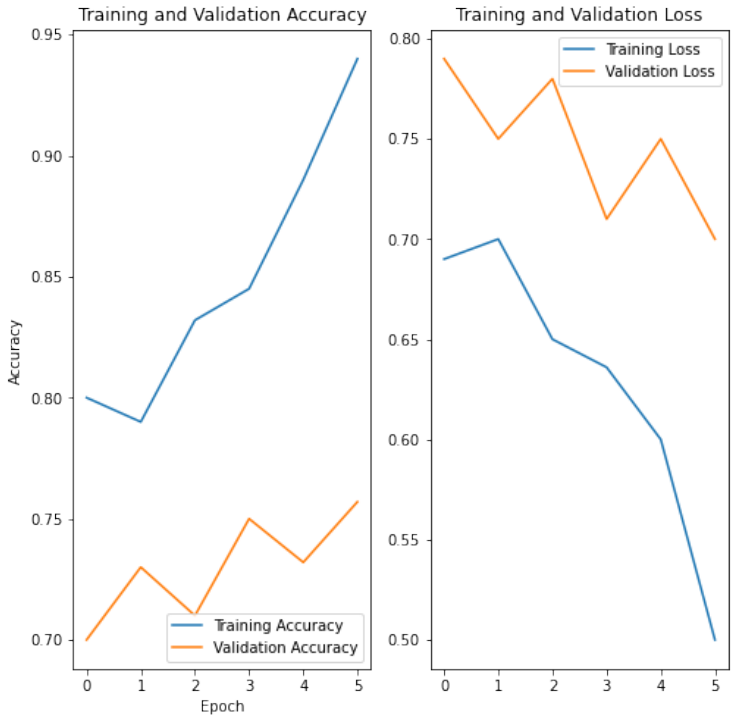


Fig. 5.5: FFT CNN LSTM

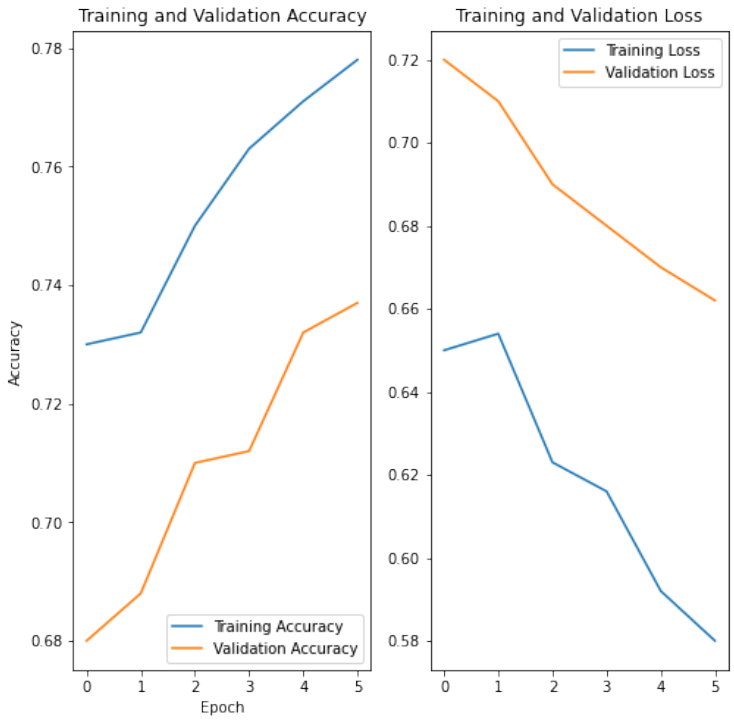


Fig. 5.6: 2 Electrode CNN LSTM

CHAPTER 6

Future Scope

Presently the project uses a two electrode model, taking in input signals from the Fp1 and Fp2 frontal lobes of the brain. The BioAmp EXG Pill Circuit can be modified and expanded to take in ERP averages from other signal points corresponding to the normal EEG headset (Fz, FCz, Cz, FC3, FC4, C3, C4, CP3, CP4 etc). This would give a higher accuarcy in classification, with more parameters taken into consideration for analysing the brain waves.

The research can also be expanded to classify a variety of mental illness like Alzheimer's, Parkinson's disease etc. using the same EEG signals. This can be used for a faster general diagnosis of the patient's mental health condition. The model will be deployed to Raspberry Pi, building a portable device that directly receives input in the form of EEG signals, detect anomalies in the brainwave to identify or predict if the patient suffers from a mental disorder and predicts the probability of the person suffering from a particular disorder.

CHAPTER 7

Conclusion

The project developed a easy to use device which can accelerate the diagnosis of schizophrenia. However due to lack of the number of electrodes in the custom EEG headset, the proposed model was pivoted. The project considers only the required number of electrodes as needed.

The Fast Fourier Transform component from the pre-processing stage was removed because it does not create much impact for a two electrode system. We experimented with five models. The 1D CNN model with 5 one dimensional convolution layers, kernel size 3, 5 filters activated using sigmoid function returned a training accuracy of 83% and a testing accuracy of 67%.

Alternatively, the 1D CNN model with 1 convolutional layer with kernel size 3 and 32 filters activated using sigmoid function returned a training accuracy of 67% and a testing accuracy of 69%.

The CNN-LSTM model with 4 convolutional layers, kernel size 3, filter count 5 activated with a sigmoid function produced a training accuracy of 87.9% and validation accuracy of 73%.

The FFT-CNN-LSTM model with 2 convolutional layers, kernel size 2, filter count 64 activated with sigmoid function produced a training accuracy of 94% and a validation accuracy of 75.7%.

Finally, the two electrode CNN-LSTM model with 4 convolutional layers, kernel size 3, filter count 5 activated with sigmoid function returned a training accuracy of 77.8% and a validation accuracy of 73.7%.

It is observed that the FFT-CNN-LSTM model had the highest validation accuracy followed by the CNN-LSTM model.

ANNEXURE I

EEG Acquisition Circuit Arduino Code

```
#define SAMPLE_RATE 256
#define BAUD_RATE 9600
#define INPUT_PIN A0

void setup() {
  // Serial connection begin
  Serial.begin(BAUD_RATE);
}

void loop() {
  // Calculate elapsed time
  static unsigned long past = 0;
  unsigned long present = micros();
  unsigned long interval = present - past;
  past = present;

  // Run timer
  static long timer = 0;
  timer -= interval;

  // Sample
  if(timer < 0){
    timer += 1000000 / SAMPLE_RATE;
    float sensor_value = analogRead(INPUT_PIN);
    float signal = EEGFilter(sensor_value);
```

```
    Serial.println(signal);
  }
}


float EEGFilter(float input) {
  float output = input;
  {
    static float z1, z2; // filter section state
    float x = output - -0.95391350*z1 - 0.25311356*z2;
    output = 0.00735282*x + 0.01470564*z1 + 0.00735282*z2;
    z2 = z1;
    z1 = x;
  }
  {
    static float z1, z2; // filter section state
    float x = output - -1.20596630*z1 - 0.60558332*z2;
    output = 1.00000000*x + 2.00000000*z1 + 1.00000000*z2;
    z2 = z1;
    z1 = x;
  }
  {
    static float z1, z2; // filter section state
    float x = output - -1.97690645*z1 - 0.97706395*z2;
    output = 1.00000000*x + -2.00000000*z1 + 1.00000000*z2;
    z2 = z1;
    z1 = x;
  }
  {
    static float z1, z2; // filter section state
    float x = output - -1.99071687*z1 - 0.99086813*z2;
```

```
    output = 1.00000000*x + -2.00000000*z1 + 1.00000000*z2;
    z2 = z1;
    z1 = x;
  }
  return output;
}
```

Web Dashboard Code

HTML code

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Schizophrenia Dashboard</title>
<!-- CSS Styles -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3
dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8t
T94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">

<!-- Custom styles for this template -->
<link href="./css/colors.css" rel="stylesheet">
```

```
<!-- JavaScript -->
<script defer src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/d
ist/js/bootstrap.min.js"
integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+Zn
N4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
crossorigin="anonymous"></script>
</head>
<body>
<nav class="navbar bg-gray-9 text-indigo-1
position-sticky top-0">
<div class="container-fluid">
<a class="navbar-brand text-indigo-1
fw-bold p-2" href="#">Patient Dashboard</a>
</div>
</nav>
<main class="main-content container-fluid bg-gray-3
text-dark pt-5 min-vh-100">
<section id="details-section" class="row d-flex
align-items-center">
<div class="col-sm-4 mx-auto d-flex
flex-column bg-gray-1 p-3">
<h2
class="text-center">Details</h2>
<div class="container mt-2">
<!-- <p>Name: </p>
<p>Age</p> --><p>
Select the patients EEG file</p>
</div>
<input class="btn bg-violet-4
mx-auto text-white" type="file"
id="filetoRead" /><button class="btn bg-violet-4
```

```
mx-auto my-2
text-white">Predict </button>
</div>
<!-- <textarea
id="editor"></textarea><br> -->
</section>
<section id="timeseries-plot" class="mt-4">
<canvas class="p-5 w-75 mx-auto"
id="timeseries-chart"></canvas>
</section>
<!-- <section id="plot-section" class="mt-4">
<canvas class="p-5 w-75 mx-auto"
id="myChart"></canvas>
</section> -->
</main>



<!-- Icons -->
<script src="https://unpkg.com/feather-icons/dist/feather
.min.js"></script>
<script>
feather.replace()
</script>

<!-- Graphs -->
<script src="https://cdn.jsdelivr.net/npm/chart.js@2.7.1/
dist/Chart.min.js"></script>
<script type="module" src="./js/chart.js">
</script>
</body>
```

```
</html>
```

Javascript Code

```
const ctx = document.getElementById("myChart");

const convertTextToNumArray = (string) => {
    const numString = string.replace( /\r\n/g, " "
    ).replace(/\r/g, " ").split(" ");
    const nums = numString.map(item => Number(item));
    return nums;
}

const timeseriesChartCtx =
document.getElementById("timeseries-chart");
document.getElementById("filetoRead").addEventListener("change",f
unction(){
var file = this.files[0];

if (file) {
var reader = new FileReader();

reader.onload = function (evt) {
data = [...data,
convertTextToNumArray(evt.target.result)];

const datasets1 = [{
 label: 'EEG Plot',
 data: convertTextToNumArray(evt.target.result),Axis
}]
```

```
const data1 = {
 labels:
 [...Array(convertTextToNumArray(evt.target.result).l
 ength)].map((a, b) => b), // X Axis
 datasets: datas

const myTimeseriesChart = new Chart(timeseriesChar{
 type: 'line',
 data: data1,
 options: {
 responsive: true,
 plugins: {
  legend: {
  position: 'top',
  },
  title: {
  display: true,
  text: 'Chart.js Line Chart'
  }
 }
 },
})};

reader.onerror = function (evt) {
  console.error("An error ocurred reading the file",evt);
};
reader.readAsText(file, "UTF-8");
    }
},false);
```

REFERENCES

[1] Diagnosis of Schizophrenia," 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2019, pp. 4521-4524, doi: 10.1109/EMBC.2019.8857946.

[2] Afshin Shoeibi, Delaram Sadeghi, Parisa Moridian, Navid Ghassemi, Jonathan Heras, Roohallah Alizadehsani, Ali Khadem, Yinan Kong, Saeid Nahavandi, Juan M. Gorriz. (2021). Automatic Diagnosis of Schizophrenia using EEG Signals and CNN-LSTM Models.

[3] Y. Luo, Q. Tian, C. Wang, K. Zhang, C. Wang and J. Zhang, "Biomarkers for Prediction of Schizophrenia: Insights From Resting-State EEG Microstates," in IEEE Access, vol. 8, pp. 213078-213093, 2020, doi: 10.1109/ACCESS.2020.3037658.

[4] W. Yan, M. Zhao, Z. Fu, G. D. Pearlson, J. Sui, and V. D. Calhoun, "Mapping relationships among schizophrenia, bipolar and schizoaffective disorders: A deep classification and clustering framework using fmri time series," Schizophrenia Research, 2021, Frontiers in Human Neuroscience.

[5] Mengjiao Hu, Xing Qian, Siwei Liu, Amelia Jialing Koh, Kang Sim, Xudong Jiang, Cuntai Guan, Juan Helen Zhou, Structural and diffusion MRI based schizophrenia classification using 2D pretrained and 3D naive Convolutional Neural Networks,Schizophrenia Research,2021.

[6] Murashko AA, Shmukler A. EEG correlates of face recognition in patients with schizophrenia spectrum disorders: A systematic review. Clin Neurophysiol. 2019 Jun;130(6):986-996. doi: 10.1016/j.clinph.2019.03.027. Epub 2019 Apr 10. PMID: 31003117.

[7] Wen, Tingxi Zhang, Zhongnan. (2017). Effective and Extensible Feature Extraction Method Using Genetic Algorithm-Based Frequency-Domain Feature Search for Epileptic EEG Multi-classification. Medicine.