ROBERT GORDON UNIVERSITY ABERDEEN

SCHOOL OF COMPUTING SCIENCE AND DIGITAL MEDI

MSc Big Data Analytics

Student Name: Rajapaksha Mudiyanselage Maneesha Indrachapa	Matriculation Number:
Supervisor: Mr.Pumudu Fernando	Second Marker:
Project Title: Autism Spectrum Disorder Classification on Electroencephalogram data using Neural Networks	
	Start Date: 18-02-2022
	Submission Date: 05-01-2023

CONSENT

I agree ■
I do not agree ☐

That the University shall be entitled to use any results, materials or other outcomes arising from my project work for the purposes of non-commercial teaching and research, including collaboration.

DECLARATION

I confirm:

- **That the work contained in this document has been composed solely by myself and that I have not made use of any unauthorised assistance.**
- **That the work has not been accepted in any previous application for a degree.**
- **All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.**

Student Signature: Maneesha Indrachapa	Date Signed: 05-01-2023

RGU SYMBOL OF SUCCESS

MSc Project Report

Autism Spectrum Disorder Classification on Electroencephalogram Data using Neural Networks

Maneesha Indrachapa

2022

A report submitted as part of the requirements for the degree of
MSc Big Data Analytics at Robert Gordon University, Aberdeen, Scotland

Declaration

I confirm that the work contained in this MSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Full Name: Rajapaksha Mudiyanselage Maneesha Indrachapa
RGU Student Number: 2016894
IIT Student Number: 20200399

Maneesha Indrachapa
…………………………………
Signature

05-01-2023
…………………………………
Date

STUDENT PROJECT ETHICAL REVIEW (SPER) FORM

The aim of the University's *Research Ethics Policy* is to establish and promote good ethical practice in the conduct of academic research. The questionnaire is intended to enable researchers to undertake an initial self-assessment of ethical issues in their research. Ethical conduct is not primarily a matter of following fixed rules; it depends on researchers developing a considered, flexible and thoughtful practice.

The questionnaire aims to engage researchers discursively with the ethical dimensions of their work and potential ethical issues, and the main focus of any subsequent review is not to 'approve' or 'disapprove' of a project but to make sure that this process has taken place.

The *Research Ethics Policy* is available at www.rgu.ac.uk/research-ethics-policy

Student Name	R.M.Maneesha Indrachapa
Supervisor	Mr. Pumudu Fernando
Project Title	Autism Spectrum Disorder Classification using Electroencephalogram data
Course of Study	Big Data Analytics
School/Department	School of Computing

PART 1: DESCRIPTIVE QUESTIONS			
1.	Does the research involve, or does information in the research relate to: [see Guidance Note 1]	**Yes**	**No**
	(a) individual human subjects		x
	(b) groups (e.g. families, communities, crowds)	x	
	(c) organisations		x
	(d) animals?		x
	(e) genetically-modified organisms www.rgu.ac.uk/hr/healthsafety/page.cfm?pge=26027#122628		x
	Please provide further details:		
	This research is trying to identify the early signs of autism spectrum disorder of children using neural networks.		
2.	Will the research deal with information which is private or confidential? [see Guidance Note 2]	**Yes**	**No**
			x

	Please provide further details:

PART 2: THE IMPACT OF THE RESEARCH			
3.	In the process of doing the research, is there any potential for harm to be done to, or costs to be imposed on: [see Guidance Note 3(i)]	**Yes**	**No**
	(a) research participants?		x
	(b) research subjects? [see Guidance Note 3(ii)]		x
	(c) you, as the researcher?		x
	(d) third parties? [see Guidance Note 3(iii)]		x
	Please state what you believe are the implications of the research:		
4.	When the research is complete, could negative consequences follow:	**Yes**	**No**
	(a) for research subjects		x
	(b) or elsewhere? [see Guidance Note 4]		x
	Please state what you believe are the consequences of the research:		

PART 3: ETHICAL PROCEDURES			
5.	Does the research require informed consent or approval from: [see Guidance Note 5(i)]	**Yes**	**No**
	(a) research participants?		x
	(b) research subjects? [see Guidance Note 5(ii)]		x
	(c) external bodies? [see Guidance Note 5(iii)]		x
	If you answered yes to any of the above, please explain your answer:		
6.	Are there reasons why research subjects may need safeguards or protection? [see Guidance Note 6]	**Yes**	**No**
			x

	If you answered yes to the above, please state the reasons and indicate the measures to be taken to address them:		
7.	Does the research involve any “regulated work with children” and/or “regulated work with protected adults”, therefore requiring membership of the *Protecting Vulnerable Groups (PVG) Scheme*? *[see Guidance Note 7]*	**Yes**	**No**
			x
	[Please note: if the research potentially involves “regulated work”, this MUST be raised with your HR Business Partner immediately. In this instance, the Human Resources Department will conduct a detailed assessment and will confirm whether or not PVG Membership is required.]		
	(a) PVG membership is not required.		x
	(b) PVG membership may be required for working with children.		x
	(c) PVG membership may be required for working with protected adults.		x
	(d) PVG membership may be required for working with both children and protected adults.		x
	If you answered yes to (b), (c) or (d) above, please give further information about the work you will be required to undertake and the nature of the contact with these groups. Please provide as much detail as possible:		
	Are you already a PVG member?	**Yes**	**No**
			x
	If yes, please provide your PVG Scheme number:		
8.	Are specified procedures or safeguards required for recording, management, or storage of data? *[see Guidance Note 8]*	**Yes**	**No**
			x
	If you answered yes to any of the above, please give details:		

PART 4: THE RESEARCH RELATIONSHIP			
9.	Does the research require you to give or make undertakings to research participants or subjects about the use of data? *[see Guidance Note 9]*	**Yes**	**No**
			x

	If you answered yes to the above, please outline the likely undertakings:		
10.	Is the research likely to be affected by the relationship with a sponsor, funder or employer? *[see Guidance Note 10]*	**Yes**	**No**
			x
	If you answered yes to the above, please identify how the research may be affected:		

Part 5: Other Issues			
11.	Are there any other ethical issues not covered by this form which you believe you should raise?	**Yes**	**No**
			x

Statement by Student			
I believe that the information I have given in this form is correct, and that I have addressed the ethical issues as fully as possible at this stage.			
Signature:	**Maneesha Indrachapa**	**Date:**	05-04-2022

If any ethical issues arise during the course of the research, students should complete a further Student Project Ethical Review (SPER) form.

The *Research Ethics Policy* is available at www.rgu.ac.uk/research-ethics-policy

PART 6: TO BE COMPLETED BY THE SUPERVISOR			
12.	Does the research have potentially negative implications for the University? *[see Guidance Note 11]*	**Yes**	**No**
			x
	If you answered yes to the above, please explain your answer:		

13.	Are any potential conflicts of interest likely to arise in the course of the research? *[see Guidance Note 12]*	**Yes**	**No**
			x
	If you answered yes to the above, please identify the potential conflicts:		
14.	Are you satisfied that the student has engaged adequately with the ethical implications of the work? *[see Guidance Note 13]*	**Yes**	**No**
		x	
	If you answered no to the above, please identify the potential issues:		

15.	**Appraisal:** Please select one of the following	
	i. The research project should proceed in its present form – no further action is required	x
	ii. The research project requires ethical approval by the School Ethics Review Panel (SERP) (or equivalent)	
	iii. The research project requires ethical review by the University's Research Ethics Sub-Committee	
	iv. The project needs to be returned to the student for modification prior to further action	
	v. The research project requires ethical review by an external body (N.B. Question 5 above). If this applies, please give these details:	
	Title of External Body providing ethical review	
	Address of External Body	
	Anticipated date when External Body may consider project	

AFFIRMATION BY SUPERVISOR			
I have read the student's responses and have discussed ethical issues arising with the student. I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.			
Signature:	Mr Pumudu Fernando	**Date:**	05-04-2022

Abstract

Autism spectrum disorder (ASD) is a neurodevelopmental disorder characterized by the presence of restricted interests, repetitive behaviours, and deficits in social communication. Diagnosis of Autism is currently a challenging and time-consuming process that involves observation of the behaviour of potential ASD individuals. However, the behavioural symptoms may vary in type and severity from person to person and they may also change with time which makes it even harder to properly diagnose by a behaviour observation. Despite being challenging, early diagnosis of ASD is considered very much important in effectively treating Autistic children and improving their lives.

Electroencephalogram (EEG) patterns of Autistic individuals are found to carry certain abnormalities compared to that of normal individuals. Therefore, EEGs can act as potential biomarkers of ASD prevalence and can be used in creating an effective ASD diagnosis system.

This research was aimed at designing and developing a low-cost, automated and behaviour-independent mechanism to diagnose the prevalence of ASD in children at an early age by using neural network algorithms. EEG data of 17 participants including both Autistic and Normal children aged between 5 to 17 years were considered in this research.

The author created a pre-processing pipeline to remove noise from EEG data before it was used in the classifications. This research attempted six different types of neural networks in classifying ASD prevalence. The best accuracy achieved was 93% and came from the ConvLSTM model. The CapsNet model gave the second-best accuracy of 83%.

Keywords – EEG data, Autism Spectrum Disorder, Deep Learning, Neural Networks, ConvLSTM, LSTM, CNN, CapsNet

Acknowledgement

I would like to extend my thanks to the below mentioned persons who were the main reason I was able to successfully complete this research.

My supervisor Dr Pumudu Fernando for taking the time to guide me through every step of the way and providing feedback.

My module leader, Mr Cassim Farook, for the endless support is given by him to successfully complete this project

Prof. Sampath Jayarathna for providing me with the dataset used in this research.

My loving wife, parents, family, and friends who encouraged me and supported me throughout the course of this research.

Table of Contents

List of Tables

List of Figures

List Of Abbreviations

Abbreviation	Definition
ASD	Autism Spectrum Disorder
EEG	Electroencephalography
EOG	Electrooculogram
ECG	Electrocardiography
Conv	Convolutional

LSTM	Long-Short-Term Memory
Bi-LSTM	Bidirectional Long-Short-Term Memory
GRU	Gated Recurrent Unit
ConvLSTM	Convolutional Long-Short-Term Memory
ICA	Independent Component Analysis
ADOS	Autism Diagnostic Observation Schedule
RF	Random Forest

Chapter 01 - Introduction

1.1 Chapter Overview

This chapter provides an understanding of the background of the problem statement of this project along with the importance of achieving and implementing the proposed solution. Furthermore, this chapter emphasizes the scope of the project and its aims and objectives which the project attempts to achieve.

1.2 Project Background

Autism spectrum disorder (ASD) is a neurodevelopmental disorder characterized by the presence of restricted interests, repetitive behaviours, and deficits in social communication. (Klintwall and Eikeseth 2014) It mainly impacts an individual's overall quality of life and the ability to function in school, the workplace, or other areas of life as it causes difficulty in communicating and interacting with others, repetitive behaviours, and restricted interests.

According to recent studies, it was found that the global prevalence of Autism has increased over the years (Matson and Kozlowski 2011). Studies have also stated that approximately 1 in every 100 children is diagnosed with autism spectrum disorder around the world.

Autism can be diagnosed at any given age of an individual. It is described as a "Developmental Disorder" because the characteristic symptoms usually start to appear during the first two years of life. The reason why it is referred to as a "Spectrum Disorder" is because of the wide variety of types and severity of the symptoms that are being shown by Autistic individuals. The symptoms can also change over time as they grow up. This very reason makes it highly challenging to diagnose Autism, especially during the early ages of toddlers as their social skills and communication skills are yet to develop.

Due to the complexity of symptom identification and the fact that currently, the diagnosis depends entirely on observation of behavioural patterns, most cases of Autism Diagnosis can go undetected for years. However, early diagnosis of Autism is considered highly important in effectively treating and improving the quality of life of Autistic children. Treatment methods such as EIBI (Early Intensive Behavioral Intervention), heavily depend on early diagnosis of

Autism in children. In addition to that, early diagnosis and information on the presence of Autism are vital to creating an Autism friendly environment around Autistic children which immensely helps their overall adaptation to their day-to-day life.

Studies have found that EEGs (Electroencephalography) can be used as a possible and effective tool to detect Autism in kids, and prediction of the presence of ASD is considered accurate when using EEGs even from as early as 3 months old babies. (Reiersen 2017) Therefore, EEGs have the ability and potential of acting as behaviour-independent digital biomarkers of Autism Diagnosis. In addition to diagnosis, novel approaches to facilitate EEG-based treatment plans have also begun to come to the attention of the medical world.

1.3 Problem Statement

Although several studies have been conducted based on EEG data to diagnose Autism using multiple classification techniques such as Random Forest, KNN, Decision Tree, etc., not many studies have been conducted to detect Autism using Electroencephalogram data (EEG) using neural networks, in early stages of children. However, given the importance of Autism diagnosis at an early age, automated and behaviour-independent diagnosis methods are in need.

1.4 Proposed Solution

This project proposes a low-cost, automated, and behaviour-independent approach based on optimized classification techniques to identify autism spectrum disorder at an early age in children. The suggested solution uses EEG data to identify the prevalence of Autism. The EEG data of a child is applied to the system and the system can predict whether the EEG of the child is showing ASD symptoms. Therefore, this system helps the detection of Autism in children at an early age

1.5 Research Aim

To design, develop, test and evaluate behaviour-independent mechanisms to diagnose the prevalence of autism spectrum disorder in children at an early age.

1.6 Research Objectives

I. To identify the most suitable dataset for this classification.

II. Clean and remove the noise in EEG data.
III. To identify the best approaches to implement the system by analyzing existing technologies used in the previous studies.
IV. To implement a classifier using neural networks to classify ASD using EEG data, which produces more accurate results with a better performance.
V. To evaluate the implemented system.

1.7 Operational Objectives

1. Prepare Project Proposal
    a. Research on the project background
    b. Identify aims and objectives
2. Conduct Literature Review
    a. Analyse the existing systems and the technologies used to implement them
    b. Identify the areas which we could improve and add to our system
    c. Research and identify the technologies and tools to be used in the proposed solution
3. Requirement gathering
    a. Gather requirements through questionnaires and interviews
    b. Analyse and evaluate the gathered information
    c. Identify stakeholders
    d. Identify functional and non-functional requirements
4. Design the system
    a. Architecture design
    b. Prototype designing
5. Document the design
6. Prototype development
    a. The prototype will be implemented to achieve the project's aim and objective.
    b. The prototype will be implemented according to the selected tools, technologies, and methodologies and also align with the identified requirements
7. Testing of the prototype
    a. Creation of test cases
    b. Execution of test cases
    c. Evaluation of the system

8. Completion of the project and submition of all deliverables

1.8 Research Questions

- How to classify autism spectrum disorder prevalence using neural networks based on EEG data at early ages?
- How to optimize current classification techniques provided for the detection of autism spectrum disorder using EEG using neural networks?

1.9 Scope of the Project

Current findings suggest that the prevalence of autism spectrum disorder can be detected using EEG data using classification techniques. The following describes the scope of the project by defining what falls in scope and out of scope from this project's scope.

1.9.1 In Scope

- Use of EEG data collected during ADOS-2 diagnostics tasks.
- Detection of Autism for early-age individuals.
- Use of different types of neural networks for classification.
- Evaluation of different neural network models used.
- A GUI for the user to upload EEG Data as a .csv file

1.9.2 Out of Scope

- Use of EEG data while engaging in other activities
- Use of any other form of data such as fMRI as input
- Detection of Autism in elderly individuals
- Use of any other classification technique instead of neural networks
- Use of Raw EEG data in any other format than .csv as the input file to the system

1.10 Chapter Summary

This chapter included a basic introduction to autism spectrum disorder, how it's current diagnosis mechanism which is based on behaviour observations consumes a lot of time and also how some cases can go undiagnosed. Then the author describes the importance of having a behaviour-independent autism detection mechanism, and how certain abnormalities in EEG data can act as a biomarker for Autism prevalence and therefore be used in such a mechanism. The next author describes the aim of this research which is to design and develop a behaviour-independent Autism Detection mechanism in children at an early age using EEG data, the research objectives and scope. Those aspects which were described in this chapter set the stage for the next chapters of this report.

Chapter 02 - Literature Review

2.1 Chapter Overview

The author utilizes this chapter to explore the domain of this project in depth. The author then reviews and summarizes existing literature in the considered domain while analyzing the previous approaches attempted by researchers to address similar problem areas. A technical analysis of possible approaches is then carried out by the author in this chapter to understand the best approaches to solving the research questions.

2.2 Problem Domain

Autism spectrum disorder is a developmental disorder that affects how people interact and behave with the rest of the world. Autism has many forms in such a way that it affects different people in different ways to different degrees. Literature suggests that approximately 67 million people are affected by autism spectrum disorder worldwide and over the past decades it has been showing a rising trend in prevalence across the globe. This has made it a major global issue and has taken the attention of the medical world. Various types of researches are carried out investigating Autism and its symptoms in the hope of finding a cure and preventive measures.

Being a spectrum disorder in nature and the symptoms being behavioural makes it challenging to diagnose ASD. The main symptoms of ASD are recognized to fall into 2 main areas.

- Difficulty in communication and maintaining social interaction
- Repetitive and restricted interests and behaviours

Currently, the diagnosis of Autism is mainly done through analysis of behaviours that fall into any of the above-mentioned areas. However, diagnosis based on behaviour observation can be far from accurate because of the multitude of symptoms shown by Autistic individuals which are complex and change over the years. Especially, noticing and recognizing symptoms falling into the above two areas can become very challenging for children, especially at an early age. Children have their unique ways of reacting to being introduced to society and this makes it harder to trace if some of their reactions are due to Autism or otherwise. On top of this, masking is also recognized as a challenge in Autism Diagnosis. Masking in Autism means suppression of Autistic characteristics or aspects of self to "appear normal". Masking is also referred to as “camouflaging” or “adaptive morphing” and is an emerging area in Autism research. (Miller, Rees and Pearson 2021) Masking makes it even more challenging to detect Autism and can cause it to go undetected until one’s adulthood. All these reasons call for the need of finding out a reliable and behaviour-independent Autism detection mechanism. To address this issue there is a lot of research carried out around the world. One such research area is related to the use of EEG analysis along with machine learning techniques to automate the detection of Autism. (Jamal et al. 2014)

Electroencephalography (EEG) is a medical imaging technique that reads scalp electrical activity generated by brain structures. The electroencephalogram (EEG) is defined as the electrical activity of an alternating type recorded from the scalp surface after being picked up by metal electrodes and conductive media (Kamel et al. 2012). EEG reading is a completely non-invasive procedure that can be applied repeatedly to patients, normal adults, and children, with virtually no risks or limitations. (Teplan and others 2002)

Researchers have observed that the EEG pattern of an Autistic individual may show certain abnormalities as opposed to that of a non-Autistic individual. Studies have been conducted to show that EEG signals are correlated with ASD prevalence and can be used as a reliable biomarker (Jamal et al. 2014). Electrical impulses used by brain cells to communicate with each other are captured in EEGs using electrodes that are attached to the scalp. The time series resulting in an EEG can be analysed to find the abnormalities and to detect Autism prevalence.

Neural networks, which are a subsidiary of the machine learning domain can be used to automate the detection process of Autism using EEG data. But when using EEG data in learning models, it is important to properly pre-process the EEG data to eliminate noise in EEG data. This is because EEG is nonstationary and is highly vulnerable to a variety of noises. The mechanisms used for noise reduction will be discussed later in the possible noise-removing approaches section. Also, the different types of neural network approaches are discussed in the possible approaches section below.

2.3 Possible Noise Removing Approaches

According to Artifacts and noise removal for electroencephalogram (EEG) (Lai et al. 2018), The EEG signals obtained from the scalp have relatively little amplitude since the skull is such a poor conductor of electricity. The amplitude of noise-related signals, particularly those originate from the muscles and eyes, are frequently one order of magnitude or more. Therefore, it would be doubtful that it would identify any brain activity at all if these well-known and most common sources of noise were not eliminated from the data. So EEG data cleaning is a necessary first step that should be taken before applying the EEG data in deep learning techniques.

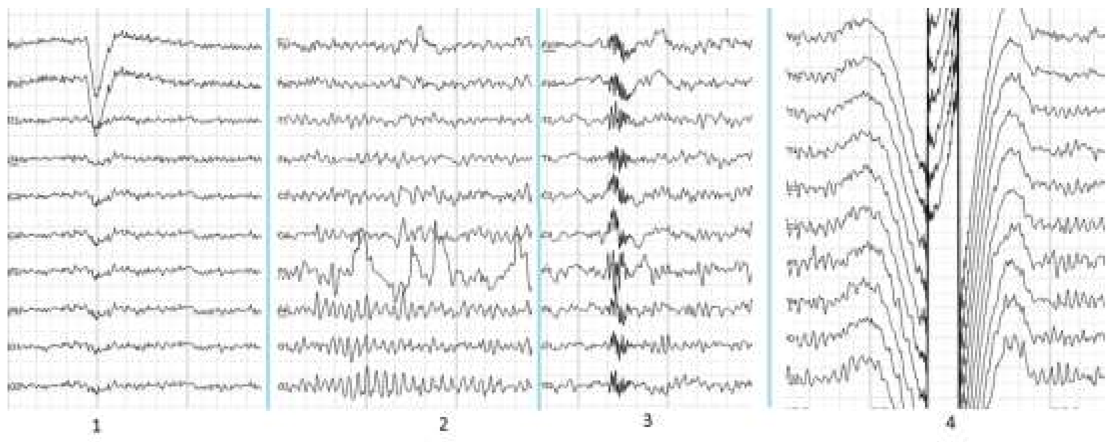


Figure 2.1-Visual patterns of different artefacts. (Deneke, 2022)

As figure 2.1 shows there can be different visual patterns of EEG data due to noise; according to figure 1 there are

1. Electrooculogram (EOG) blinking artefact. Big amplitude, slow, positive wave prominent in frontal electrodes.
2. Electrode artefact caused by unstable contact (bigger impedance) between P3 electrode and skin.
3. Swallowing artefact.
4. Common reference electrode artefacts are caused by unstable contact between the reference electrode and skin. The huge wave was similar in all channels.

So, removing noise before using EEG data is mandatory. In the next phase of this chapter, the author is illustrating the findings that can be used to remove the noise in EEG data.

2.3.1 Low and High Pass Filters

The simplest way to eliminate low-frequency signals from an EEG signal is by using high-pass filtering. It is quite challenging to distinguish between baseline wander and EEG since their spectrums are mixed. Baseline wanders noise can be reduced using adaptive filtering and the cubic spline approximation. Butterworth is shown filtering to reduce high-frequency noise from EEG signals with the use of Low and High Pass Filter design approximation. (Gaikwad and Chavan 2014)

2.3.2 Independent Component Analysis (ICA)

Independent component analysis is a computational method that can be used in signal processing, to separate distinct sources that have been linearly combined in several sensors. For instance, ICA can identify artefacts in data when scalp electroencephalograms (EEG) are recorded, since they are usually independent of each other. (Comon 1994)

2.3.3 Artifact Subspace Reconstruction (ASR)

Several eminent scientific organizations have recommended utilizing Artifact Subspace Reconstruction (Chang et al. 2018). ASR is a component-based, online technique for efficiently removing transient or significant amplitude artefacts. In multichannel data sets, the method can operate in real-time and uses statistical anomaly detection to distinguish artefacts from EEG signals. It is predicted that non-brain signals significantly increase the variance of the data collection and can be identified statistically.

2.3.4 Fully Automated Statistical Thresholding for EEG artefact Rejection

Five separate statistical criteria are used by Fully Automated Statistical Thresholding for EEG Artifact Rejection (FASTER) to identify "poor" sensors. The Hurst exponent, variance, correlation, kurtosis, and line noise make up this list. If a sensor's z-score for a given criterion is greater than 3, it is considered to be poor. (Nolan, Whelan and Reilly 2010)

2.4 Possible Approaches

2.4.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of Deep Neural Network that is fed forward and able to study concurrently. The natural visual perception mechanism of living creatures has been the inspiration for this deep-learning architecture. CNN is widely known as a recognition algorithm that excels in pattern recognition and image processing and therefore is used in CNN is known for its adaptability, simple structure, and the use of fewer training parameters. The general structure of CNN consists of two layers namely the feature extraction layer and the feature map layer. (Wu 2017)

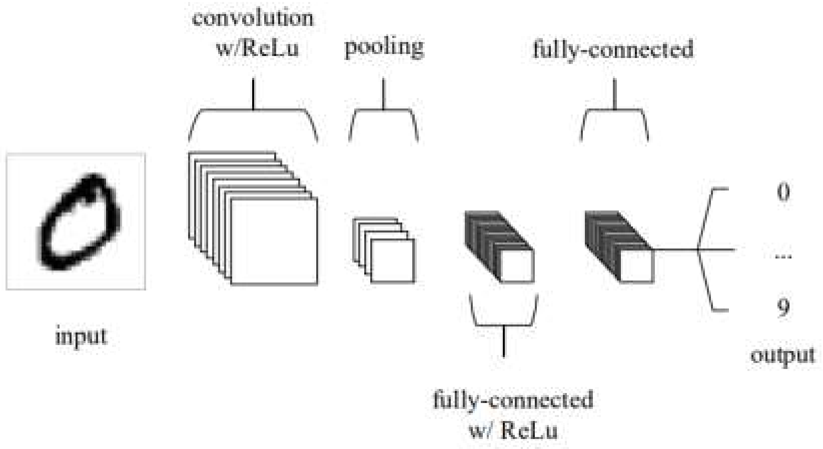


Figure 2.2-A simple CNN architecture, comprised of just five layers (Wu 2017)

The basic functionality of the example CNN above can be broken down into the below key areas. (O'Shea and Nash 2015) The input layer will store the image's pixel values, as is typical of other ANN types. Then the convolutional layer will calculate the scalar product between the weights of the input volume-connected region and the neurons whose output is connected to particular regions of the input. The goal of the rectified linear unit, also known as ReLu, is to activate the output of the previous layer's activation by applying an "elementwise" activation function, such as sigmoid. To further reduce the number of parameters in that activation, the pooling layer simply applies down sampling along the spatial dimensionality of the input. The fully connected layers will then carry out the corresponding functions seen in conventional. The feature extraction is happening in the first layers and the feature map is happening in the final fully connected layer and output layer. (i.e figure 2.2)

In the analysis of Temporal Relationships between ASD and Brain Activity through EEG and Machine Learning by Jayawardana, Jaime and Jayarathna 2019 et al.; they were able to use the CNN model which has the architecture of figure 2.3. After processing the EEG data an algorithm was devised to calculate the power matrix for a time series; using that algorithm final power matrix was generated and fed to the convolutional NN.

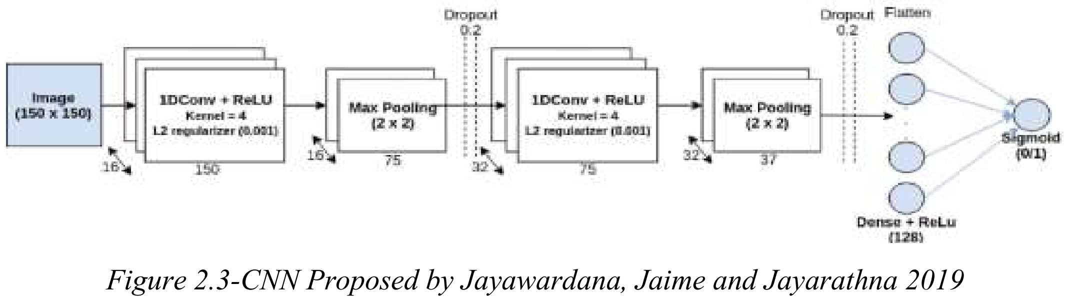


Figure 2.3-CNN Proposed by Jayawardana, Jaime and Jayarathna 2019

2.4.2 Long-Short-Term Memory

Long-Short Term Memory (LSTM) is an Advanced Recurrent Neural Network (Hochreiter and Schmidhuber 1997) which is a sequential network that can handle long-term dependencies. LSTM s have feedback connections and are considered to work well with sequence-based tasks with long-term dependencies. LSTMs can eliminate shortcomings of the classical “Vanishing” and “Exploding” problems that RNNs usually suffer from.

According to LSTM-based Electroencephalogram Classification on Autism Spectrum Disorder, (Ali et al. 2021) the input, forget, and output gates are three gates that are contained within a single LSTM block. The input gate decides to specify whether the block's current state condition is changed by a series of inputs. The output gate monitors the information output and determines whether the current concealed state is shifted to the next iteration, while the forget gate is employed to hold or delete the preceding condition on the LSTM block. Figure 2.4 shows the structure of the LSTM block. In (Ali et al. 2021) author created an LSTM network to analyze the EEG data consisting of 117 channels.

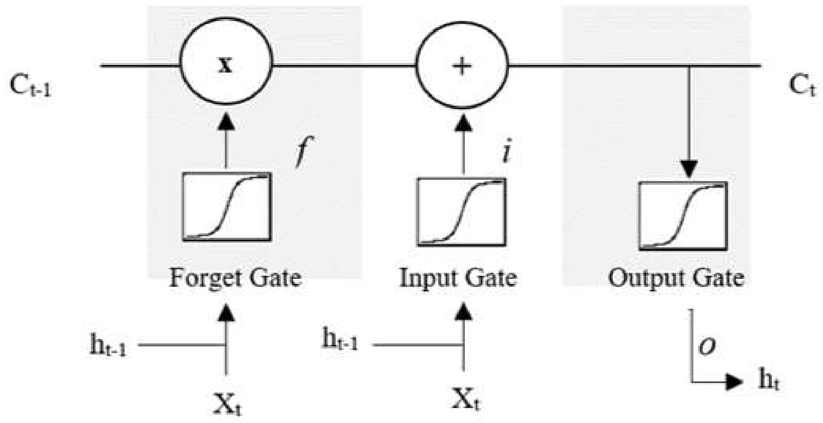


Figure 2.4-LSTM internal block structure (Ali et al. 2021)

2.4.3 The Gated Recurrent Unit

The Gated Recurrent Unit or the GRU can be considered as a simplified version of the LSTM which is improved than standard recurrent neural networks. Introduced by Cho et al. in 2014, GRUs are created to solve the common vanishing gradient problem faced by standard recurrent neural networks by using an update gate and reset gate.

According to Universal EEG Encoder for Learning Diverse Intelligent Tasks by Jolly et al.; (i.e figure 2.5) the Recurrent Neural Networks (RNNs) are anticipated to perform better because EEG is a time-series data. The output can be fed back into the network by them. The Gated Recurrent Unit (GRU), which makes use of update and reset gates, was suggested as a solution to a similar problem. The reset gate defines how much of the past knowledge is to be forgotten, while the update gate helps the model decide how much of the past information needs to be passed on to subsequent iterations.

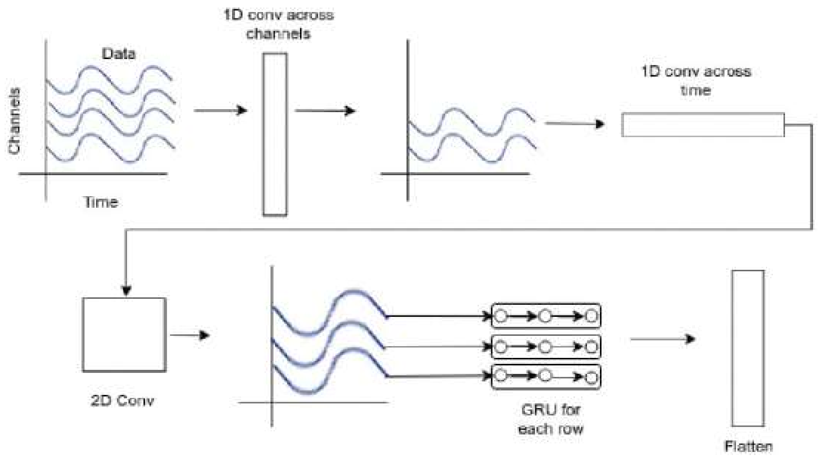


Figure 2.5-GRU architecture for EEG data (Jolly et al. 2019)

2.4.4 Capsule Network

To compensate for the shortcomings of Convolutional Neural Networks, a network structure called the capsule network has been proposed (Sabour, Frosst and Hinton 2017). The CapsNet is a deep network model consisting of capsules. Compared to alternative deep neural networks, CapsNet consist of capsule layers where each node is a capsule containing a series of neurons. The responsibility of each capsule is to determine a single component in the object and therefore all capsules as a whole can jointly identify the structure of the entire object. Compared to other deep neural networks, CapsNet can preserve information on objects, components and other spatial information. An encoder (i.e., Figure 2.6) and a decoder, each with a set of three layers, make up the CapsNet architecture. Convolutional, PrimaryCaps, and DigitCaps layers make up the encoder, whereas three fully connected layers make up the decoder. (Sabour, Frosst and Hinton 2017)

The CapsNet has been used in many areas such as NLP, medical image processing, speech recognition, etc. In Multi-channel EEG-based emotion recognition via multi-level features, guided capsule networks by Liu et al. used CapsNet for EEG data and generated a model to find emotion recognition.

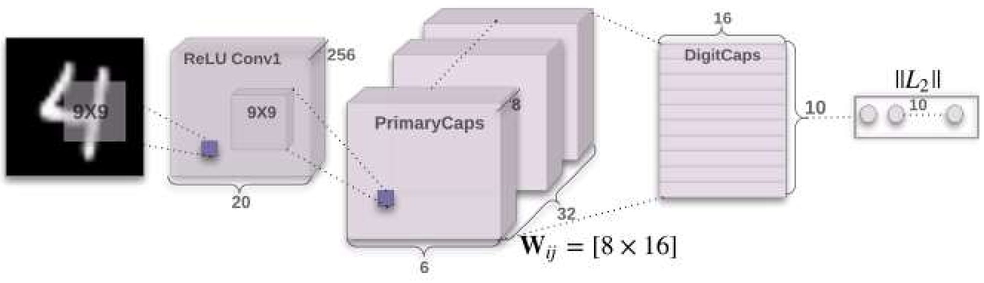


Figure 2.6-CapsNet encoder architecture (Liu et al. 2020)

2.4.5 ConvLSTM

Introduced by Shi et al. in Convolutional LSTM Network: A machine learning approach for precipitation nowcasting; ConvLSTM is a kind of recurrent neural network used for spatiotemporal prediction that incorporates convolutional structures in both input-to-state and state-to-state transitions. By using the inputs and previous states of its local neighbours, the ConvLSTM predicts the future state of each cell in the grid. Using a convolution operator in the state-to-state and input-to-state transitions makes this simple to accomplish. Figure 2.7 shows the inner structure of the ConvLSTM layer.

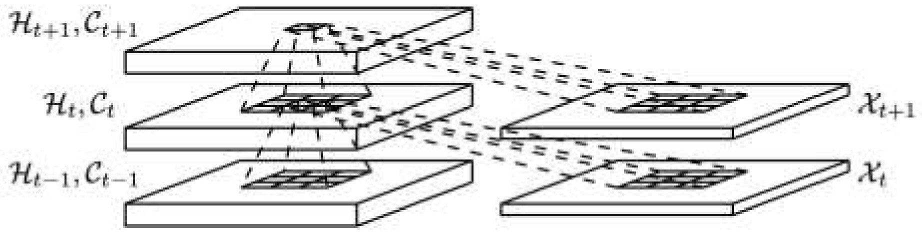

$\mathcal{H}_{t+1}, \mathcal{C}_{t+1}$
$\mathcal{H}_t, \mathcal{C}_t$
$\mathcal{H}_{t-1}, \mathcal{C}_{t-1}$
$\mathcal{X}_{t+1}$
$\mathcal{X}_t$

Figure 2.7-Inner structure of ConvLSTM (Shi et al. 2015)

Introduced by M. N. A. Tawhid et al. in A Convolutional Long Short-Term Memory-Based Neural Network for Epilepsy Detection from EEG (Tawhid, Siuly and Li 2022) used a 2-D convolutional LSTM (ConvLSTM) model to perform the classification of 2-D EEG data for Epilepsy Detection.

2.4 Related Research

Several studies have been conducted in the recent past to detect the correlation between ASD and EEG data.

In Haputhanthri et al. 2019, the authors have been able to present a low-cost and straightforward diagnostic approach to detect ASD based on EEG data and learning models.

They have explored the possibility of using a minimum number of EEG channels. The data set used consisted of EEG data from 15 participants, 9 male and 6 female, between 5 and 17 years of age. 10 of them had a prior ASD diagnosis and 5 had not. The authors have implemented a 3-phase process methodology namely pre-processing, feature extraction and classification. They have compared 4 different learning models in this research namely Logistic Regression, SVM, Naive Bayes and Random Forest. Based on the results obtained, the Random Forest model couples with CFS gave the highest accuracy of 93%. The Logistic Regression model also gave promising results with an accuracy of 87%.

Jiang et al. 2022, authors present an ASD classification network defined as CNNG which is a combination of convolutional neural networks (CNN) and gated recurrent units (GRU). However, this research was done based on fMRI data. According to this study, CNNG has given an accuracy of 72.46% in extracting spatiotemporal features of fMRI. The rs-fMRI data that was used in this paper were from the international publicly available Autism Brain Imaging Data Sharing Project which consisted of data from 1112 individuals from 17 sites around the world, out of which 539 were Autistic and 573 TCs.

Jiao, Li and Fan 2020, the authors have attempted to improve autism diagnosis by analyzing the Functional Connectivity Patterns (FC) based on fMRI data from 1035 individuals. They used Random Forest, SVM, DNN and CapsNet to classify and they obtained the best accuracy of 71% from CapsNet.

Ali et al. 2020 the authors have obtained an accuracy of 80% in classifying autism spectrum disorder on EEG data by using a deep learning model. The data set tested in this research consisted of EEG data from only 20 individuals. The deep learning model is based on CNN architecture with a total of 6 layers consisting of 3 convolutional layers, one flattened layer and 2 dense (fully connected) layers. Batch Normalization is applied after each layer and the model also uses Gaussian dropout weights. The ReLu activation function is used in each convolution layer and the Softmax activation function is used in the final dense layer. The dataset distribution is 80% for the training set, 10% for the validation set and the remaining 10% for the test set.

In Ali et al. 2021, LSTM-based Electroencephalogram Classification on Autism Spectrum Disorder, the researchers have proposed a novel Autism Diagnosis method based on a bidirectional Long-short-term-memory (LSTM) network's deep learning algorithm. The dataset for this study was taken from the Simon Foundation Autism Research Initiative

(SFARI). An accuracy of 99.6% has been obtained for the 90:10 train: test data distribution whereas an accuracy of 97.3% was achieved for the 70:30 distribution of the dataset. They have concluded that a bidirectional LSTM method has been able to upgrade the efficiency of Autism classification compared to that of a single LSTM network method.

In Kang et al. 2020 the authors use the SVM approach on EEG and eye tracking data to identify the prevalence of autism spectrum disorder. The data set used in this study consisted of rest state EEG and eye tracking data from 97 children aged 3 to 6 years. The MRMR (Minimum Redundancy Maximum Relevance) feature selection method combined with the SVM classifier was used for the classification of Autistic vs normal. They obtained a classification accuracy of 85.44% with an AUC of 0.93 when 32 features were selected.

In Thapaliya, Jayarathna and Jaime 2018 the researchers evaluated the EEG and eye movements for autism spectrum disorder using data from 34 individuals. The data set distribution has been 80% training and 20% testing so 27 out of 34 were used in training data and the remaining 7 in test data. The 7-test data included 4 individuals diagnosed with ASD and the other 3 were not. They then pre-processed the data and fed it into 4 different models namely SVM, Logistic Regression, Deep Neural Networks (DNN) and Gaussian Naive. They have been able to get 100% accuracy with Logistic Regression using a combination of EEG standard deviation and eye data without Principal Component Analysis (PCA). The highest performing SVM has been with an accuracy of 71% using Shannon Entropy with all the features without PCA.

The author found no substantial literature created previously that has generated an Autism Detection classification in children based on EEG data using a ConvLSTM model.

Research	**Used Technologies**	**Details**	**Limitations, Future suggestions**
Haputhanthri et al. 2019	Logistic Regression, SVM, Naive Bayes, Random Forest	EEG data from 15 Participants between 5 to 17 years of age and able to acquire 93% accuracy using Random Forest	Suggests the use of advanced learning models
Jiang et al. 2022	CNN+GRU	fMRI data from 1112 Participants and able to obtain	Uses fMRI data which is more

		72.46% accuracy	costly than EEG
Jiao, Li and Fan 2020	Random Forest, SVM, DNN and CapsNet	fMRI data from 1035 individuals and able to get 71% accuracy from CapsNet	Uses fMRI data which is more costly than EEG
Ali et al. 2020	CNN	Used EEG data from 20 individuals and was able to obtain an accuracy of 80%	Suggests improving accuracy
Ali et al. 2021	Bi-LSTM	Used EEG data from SFARI involving 57 individuals (but 4 cases were excluded) and was able to obtain 99.60% accuracy	Suggest if the train test split is 70% and 30% there is an accuracy drop and needs improving that
Kang et al. 2020	SVM	EEG and Eye Tracking data from 97 children ages 3 to 6 years were used and were able to obtain an accuracy of 85.44%	Suggests improving accuracy
Thapaliya, Jayarathna and Jaime 2018	SVM, Logistic Regression, RF and Gaussian Naive	Used EEG and Eye Movements of 34 Individuals and obtained100% accuracy	Limitation due to low participant count. Suggests the use of a higher participant count
Jayawardana, Jaime and Jayarathna 2019	Random forest, Linear regression and CNN	Used EEG data from 17 Participants Between 5 to 17 years of age and able to obtain 97% accuracy using Random Forest	Limitation due to sample size. Suggest using bio-behavioural measures

Table 2.1- Related Research

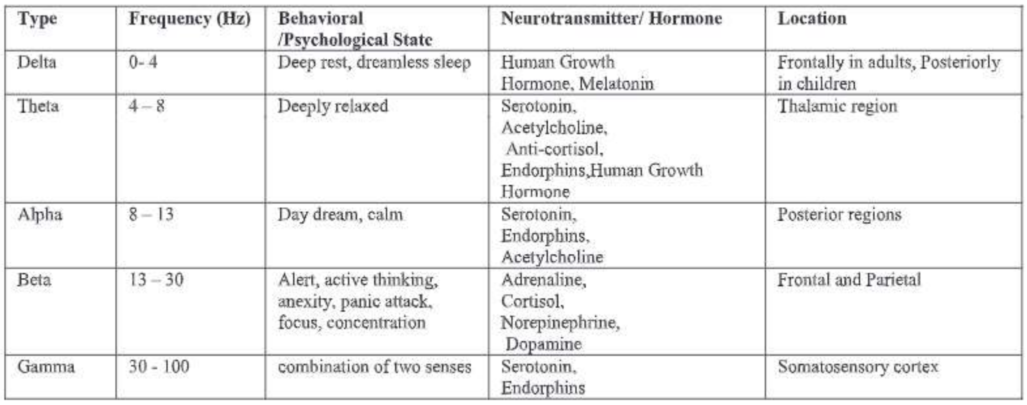
2.5 Technical Framework

2.5.1 Understanding EEG data

Electroencephalography, or EEG, is a method for capturing brain electrical activity. Although it can also be recorded from electrodes positioned directly on the surface of the brain, it is normally non-invasively recorded using electrodes positioned on the scalp (typically for clinical neurological purposes). (Olejniczak 2006) Even at the upper end of that spectrum, there are still much fewer electrodes than the approximately 80 billion neurons in the brain, according to estimates. EEG normally uses between one and 250 electrodes. The skull also has weak electrical conductivity. This means that the aggregated activity of many neurons acting in concert is always what we record with EEG.

EEG is captured as a continuous, time-varying signal, which is an essential contrast between it and other types of data we have so far worked with. As a result, the information is a time series. The signal was discrete rather than continuous. The only possible values for each time point in data from a single unit were 0 (no spike) and 1. (spike). EEG, however, records electrical voltage, which is continuously variable (e.g., the voltage could be 0 or -13.352, or -82.43, etc.). Due to the placement of the electrodes over the scalp and the fact that each electrode will pick up a distinct signal based on its location, EEG data frequently also has a spatial dimension. (Kumar and Bhuvaneswari et al.,2012)

EEG can be seen, examined, and interpreted in two different ways, or domains, because it is a time-varying signal. EEG data is often displayed in the time domain, with voltage on the Y-axis and time plotted on the X-axis. EEG can also be observed in the frequency domain. This is accomplished by using a rapid Fourier transform, a common mathematical transformation (FFT). (Olejniczak 2006) When using EEG data. One explanation is that the brain seems to produce aperiodic ("one-off") peaks and troughs in addition to periodic (oscillating sine wave) activity. Since aperiodic signals are often best viewed and analyzed in the time domain, Event-Related Potentials are frequently used in cognitive neuroscience studies. However, visualizing and examining periodic signals is better done in the frequency domain. It appears that the brain uses several frequency bands to carry out various functions and send various information across different brain regions. The following table 2.2 lists those.



Type	Frequency (Hz)	Behavioral /Psychological State	Neurotransmitter/ Hormone	Location
Delta	0- 4	Deep rest, dreamless sleep	Human Growth Hormone, Melatonin	Frontally in adults, Posteriorly in children
Theta	4 – 8	Deeply relaxed	Serotonin, Acetylcholine, Anti-cortisol, Endorphins,Human Growth Hormone	Thalamic region
Alpha	8 – 13	Day dream, calm	Serotonin, Endorphins, Acetylcholine	Posterior regions
Beta	13 – 30	Alert, active thinking, anexity, panic attack, focus, concentration	Adrenaline, Cortisol, Norepinephrine, Dopamine	Frontal and Parietal
Gamma	30 - 100	combination of two senses	Serotonin, Endorphins	Somatosensory cortex

Table 2.2-Brain wave frequency list (Olejniczak 2006)

2.5.2 Understanding ADOS-2

According to Dolan et al.2013 ADOS-2 stands for The Autism Diagnostic Observation Schedule-Second Edition which is a standardized assessment tool that helps providers diagnose autism spectrum disorders in children and adults. The ADOS involves a semi-structured play or interview session determined by the age and communication level of the individual. The ADOS offers the examiner opportunity to observe behaviours that are directly related to the diagnosis of ASD using standardized tasks and questions. The ADOS-2 uses planned social tasks that are created for various developmental stages and chronological ages and that offer contexts where social interactions, communication, and specific sorts of behaviours are likely to occur.

According to the participant’s age, linguistic proficiency, and developmental stage, there are five different modules available. The evaluation typically takes between 40 and 60 minutes to finish. Based on observations made throughout the session about many facets of social behaviour, the examiner assesses the ADOS-2. The ADOS-2 score reveals whether the person's presentation is aligned with an ASD diagnosis.

2.5.3 Understanding Neural Networks

Deep learning techniques are based on neural networks, commonly referred to as artificial neural networks (ANN), a branch of machine learning. They take their cues from the way biological neurons communicate with one another, emulating the name and structure of the human brain. (Kumar and Bhuvaneswari 2012)

As in figure 2:8, ANN consists of an input layer, one or more hidden layers, and an output layer. The artificial neurons, or nodes, are interconnected and each one has a threshold and weight that go with it. An individual node is activated and begins sending data to the network's next layer when its output exceeds the predetermined threshold. Otherwise, no data is transmitted to the network's next tier. (Kumar and Bhuvaneswari 2012)

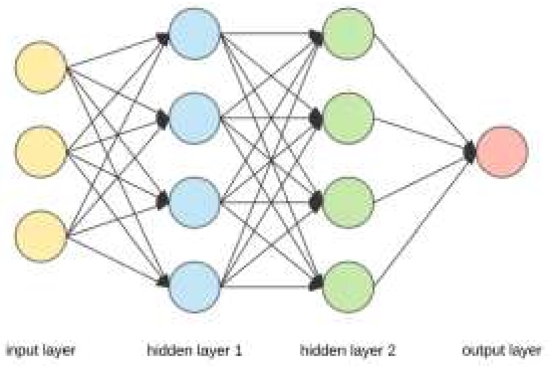


Figure 2.8-Artificial Neural Network structure

Both supervised and unsupervised methods can use ANNs. They can be distinguished depending on their operational flow and architectural design. Two of them are the Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) types.

2.5.4 Understanding Activation Functions

The activation function of a node in artificial neural networks determines that node's output given an input or group of inputs. The formula below can be used to define a neuron's output, which can vary from negative infinity to positive infinity. To obtain the desired forecast or generalization findings, it is therefore important to bound the output.

$$\mathbf{Y = \sum (weights * input + bias)}$$

In the functions, output(Y) is the result of adding the inputs' weights and biases. Popular activation functions for use with neural networks include the following. More details about activation functions can be found in Appendix A: A.1

2.6 Chapter Summary

In this chapter, the author discusses the problem domain in further detail emphasizing how observational Autism detection is not efficient and thus the necessity of a reliable and behaviour-independent detection mechanism. Then the author discusses how the EEG data were pre-processed by removing noise. Next, the possible approaches to classify EEG data are discussed in the chapter. Then the author reviews previous literature available related to the same problem domain area followed by a technical framework.

Chapter 03 - Requirement Analysis

3.1 Chapter Overview

Before development of a project, it is very crutial to understand the requirement elicitation techniques that need to be used in the project. In this chapter, the author discusses a context diagram of the system and then the work on stakeholder analysis. Later, the author discusses the process utilized for gathering requirements, data gathering techniques, use cases, and functional and non-functional requirements of the system. Having an overall image of these areas will give a proper foundation for the proposed system.

3.2 Stakeholder Analysis

A stakeholder can be identified as a person or a group who has a legitimate interest in the outcome of a project or a business strategy. The stakeholders are two main types namely, internal and external. Internal stakeholders are those who are interested in the project directly and the external stakeholders are those who are affected by the outcomes of the project. These outcomes can be positive or negative. So before developing a project or a strategy it is important to identify the stakeholders which helps to get a clear understanding as to whether the outcomes or results of the project or strategy are in the best interest of its stakeholders.

3.3.1 Onion Model

This model is a conceptual framework for describing the hierarchical order of the parties that are involved with a project. This is important because using this helps to identify the stakeholders in different hierarchies and their affecting levels in the project. In the onion model, both positive and negative stakeholders are listed.

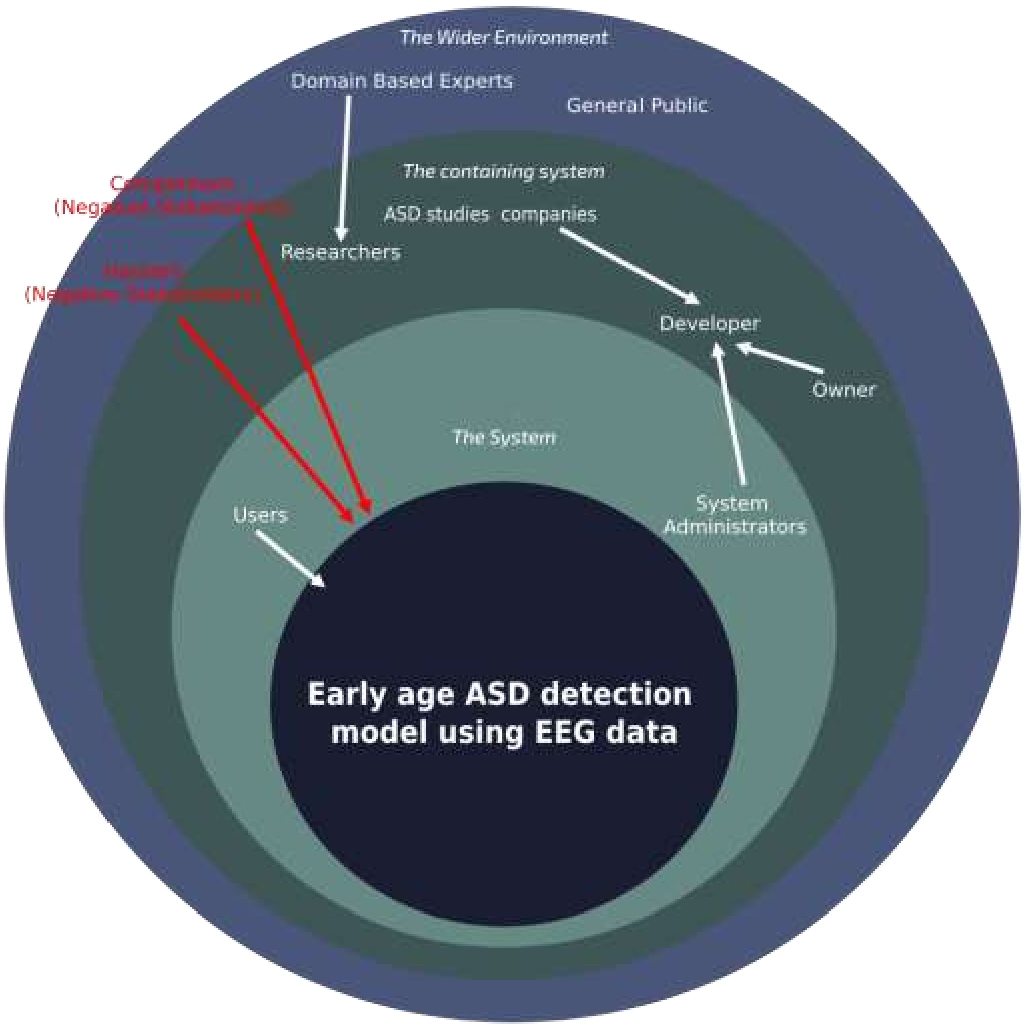


Figure 3.1- Onion Model

The Users and the System Administrators can be identified as direct interactors of this system. They are dealing with the product directly. This project is more towards a research based project. Therefore the second layer includes the researchers who are involved with the system by developing the technical background of the research along with the ASD studying companies who provide the necessary data for the research and the developers who used that data to develop the project. The wider environment consists of domain experts, the general public, other competitors who try to implement the same kind of solutions, and hackers who try to attack the system which can negatively affect the system.

3.3.2 Stakeholder Viewpoint

Stakeholder	Viewpoint
Functional Beneficiaries	
Users	Users will give the EEG data to the system and receives the result whether it is ASD positive or negative.
System Admin	System admins are responsible for providing accurate results with good efficiency. They are also responsible for monitoring the system and its performance of the system.
General Public	Able to access the application at any given time and have a good user experience without any interruptions while using the application.
Implementory	
Researcher	Other researchers who are trying to improve the accuracy of the system or trying to do enhancements to the project domain and the technologies used.
ASD studies companies	ASD studies companies that provide the necessary data collected for the project and try to find a cure for the ASD
Domain Experts	Experts who are sharing their knowledge in the project domain to improve the product and looking to improve the product domain using the results of the product
Developer	The developed application should give correct results to the end users.
Negative	
Hackers	Phishing for a possible breaching point to collapse the system.
Competitors	Will create similar products

Table 3.1-Stakeholder viewpoint table

3.3 Requirement Gathering Techniques Utilized

The process for gathering requirements is covered in this section. These techniques are used to determine which features the stakeholders who are connected to this project are expecting and which features are missing from the proposed solution or an existing product.

The techniques author has utilized while concentrating on the proposed solution, are to identify the main requirements of the proposed solution and drawbacks and missed parts in the existing solution, and the areas that need improvement. Some of the requirement-gathering techniques are as below.

Technique	**Justification**
Brainstorming	Brainstorming was conducted with the people and colleagues who were interested in this domain. This method was chosen because, when used for requirement collection, it is simple to discover the criteria that must exist from many angles. Not only that but those fresh concepts could not even be picked up in a different study.
Questionnaires	This is often referred to as social analysis. Analysts will be free to monitor the surroundings of the stakeholders here. When stakeholders are unable to articulate their requirements, they are appropriate. This was chosen for this research because it helps to identify the practical scenarios of what are the identification methods used currently for the identification of autism, and this helps to validate the findings which have been gathered in previous requirement-gathering techniques.
Interviews	Interviewing is a requirement gathering methodology that works well with both general users of the system and also on domain experts. Interviewing people results in learning more as it opens the door to a more discussion-based structure rather than just straightforward replies.
Literature review	Exploration of previous researches which are similar in background and areas of interest is helpful in understanding the drawbacks of those

	systems and techniques used to develop the system. This further helps in filling the existing research gaps in considered domains.

Table 3.2- Requirement gathering techniques

From the above-discussed techniques, most requirements are gathered by observing and analysing the existing research. Analysing the drawbacks and the findings of the technical components seen in similar research helps to enhance the proposed system while achieving its project aim and objectives. Additionally, it helps to identify which types of a methods and algorithms better suit to be utilized in each stage of the implementation process of the system proposed. Given that this is more of a research-based project, the best methodology to be carried out is decided by analysing the existing research through a literature review which is carried out before starting to design and implement the project.

3.4 Discussion of Results

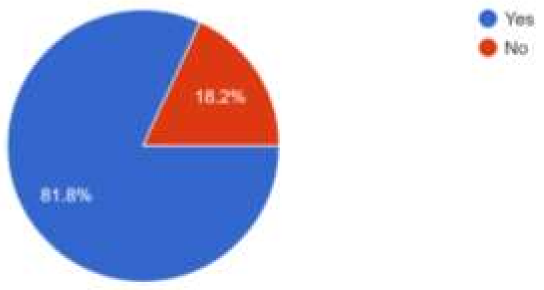
3.4.1 Findings from Literature Review

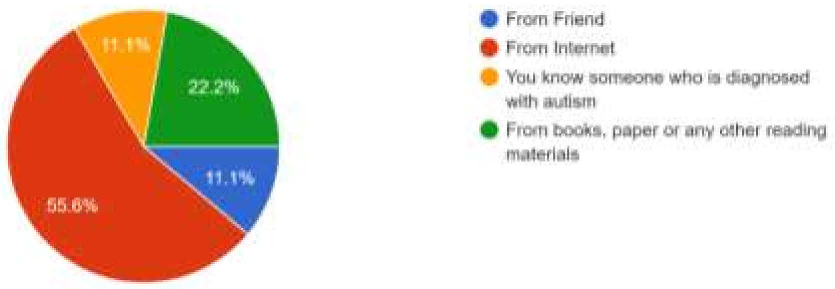
Findings	**Citation**
Classification of Autism Spectrum Disorder using EEG data	(Ali et al., 2020,2021) (Jamal et al., 2014) (Jiang et al., 2022)
Consider EEG data pre-processing techniques such as filtering, ICA, bad channel interpolation	(Djemal et al., 2017) (Thapaliya et al., 2018)
Unsupervised techniques such as deep learning for the detection of ASD classification using EEG data	(Ali et al., 2020) (Prieto et al., 2016)
Consider power spectrum analysis and use disbanding of frequency channels	(Chang et al., 2018) (Jayawardana et al., 2019) (Lai et al., 2018)

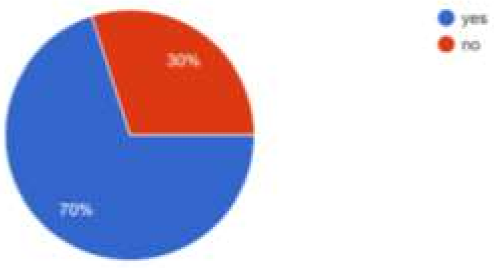
Table 3.3-Findings from Literature Review

3.4.2 Findings from Questionnaires

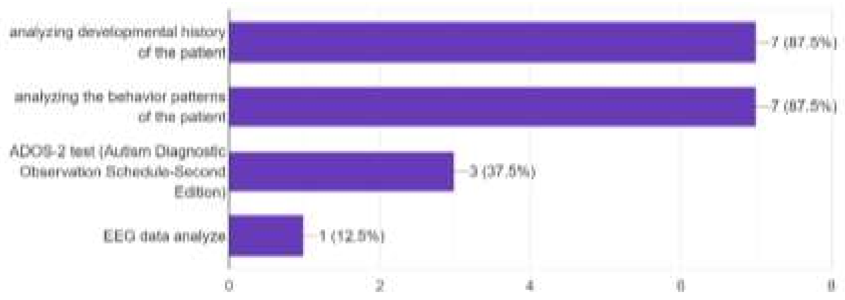
Question	**Do you know about autism spectrum disorder?**
Aim of Question	Analyzing the awareness of ASD and what people think about ASD

	
Observations	A majority of 81.8% know about what autism spectrum disorder is.

Question	**How do you get to know about autism spectrum disorder?**
Aim of Question	To get an idea about how much awareness people have about autism spectrum disorder 
Observations	A majority of 55.6% get to know about ASD from the internet and 22.2% get to know about it from reading materials and 11.1% know a person who was diagnosed with ASD.

Question	**Are you familiar with the techniques used to detect autism spectrum disorder?**
Aim of Question	To get an idea about how much awareness people have about the techniques used to detect ASD. 

Observations	70% of the people who answered the questionnaire know about the techniques that use in current society to detect ASD

Question	**What are the techniques you know about detecting autism?**
Aim of Question	To get an idea about how much awareness people have about the techniques currently used to detect ASD. 
Observations	Only one response got for the question about using EEG data analysis for ASD detection. Most people only know about analysing the behavioural patterns and development history of the patients.

3.5 Data Gathering Techniques Selected

This project needed a data set that contains EEG data of both children who are diagnosed with Autism and of healthy children (no prevalence of Autism). From the conducted literature review it has been found that they have used several EEG datasets, but there are only a few open EEG datasets available publicly.

From the conducted literature review it was found that there are few free EEG datasets available. According to Djemal et al. EEG based on the computer-aided diagnosis of autism spectrum disorder using Wavelet, Entropy and ANN used the dataset provided by King Abdulaziz University (KAU) Brain-Computer Interface (BCI) Group, Jeddah, Saudi Arabia. (ALI, N.A. et al., 2021) Also, there are a few websites that provide EEG datasets. One of them is SFARI, although SFARI is an effort that focuses on ASD, not all of the subjects in the VIP collection have ASD: they have a spectrum of precisely identified genetic disorders that are associated with ASD. Few open EEG datasets are available for public usage. There is also a dataset available from the National Database for Autism Research (NDAR)

Some previous researchers used EEG datasets collected by their universities. So contacting them can gather the EEG data because EEG data with participants who have autism spectrum disorder are confidential and also it is hard to get the data for early-age children. There is a data set collected by M. Jaime and S.Jayarathna for Evaluating EEG and Eye moments for autism spectrum disorder. It contains EEG and Infrared thermography (IRT) data of participants who have ASD spectrum and control subjects. The study includes 17 participants between the ages of 5 to 17 years old. For the moment it is decided to contact M. Jaime or S.Jayarathna to collect their dataset because it consists of early-age EEG data with ASD.(Jayawardana, Jaime and Jayarathna 2019). Figure 3.2 and 3:3 shows the data distribution and time series of the dataset.

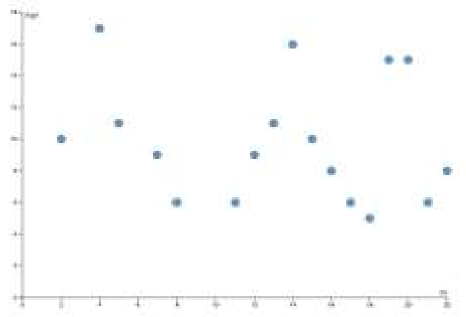


Figure 3.3-Participant ID vs age in dataset

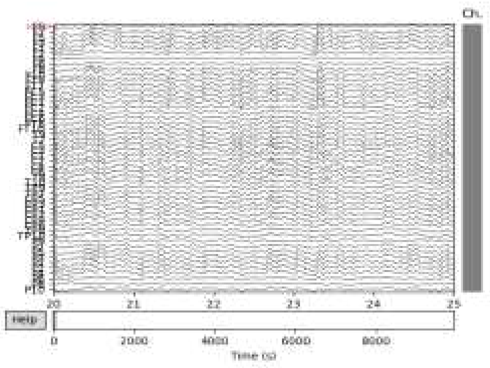


Figure 3.2-Electrode readings of dataset

3.6 Use case Diagrams

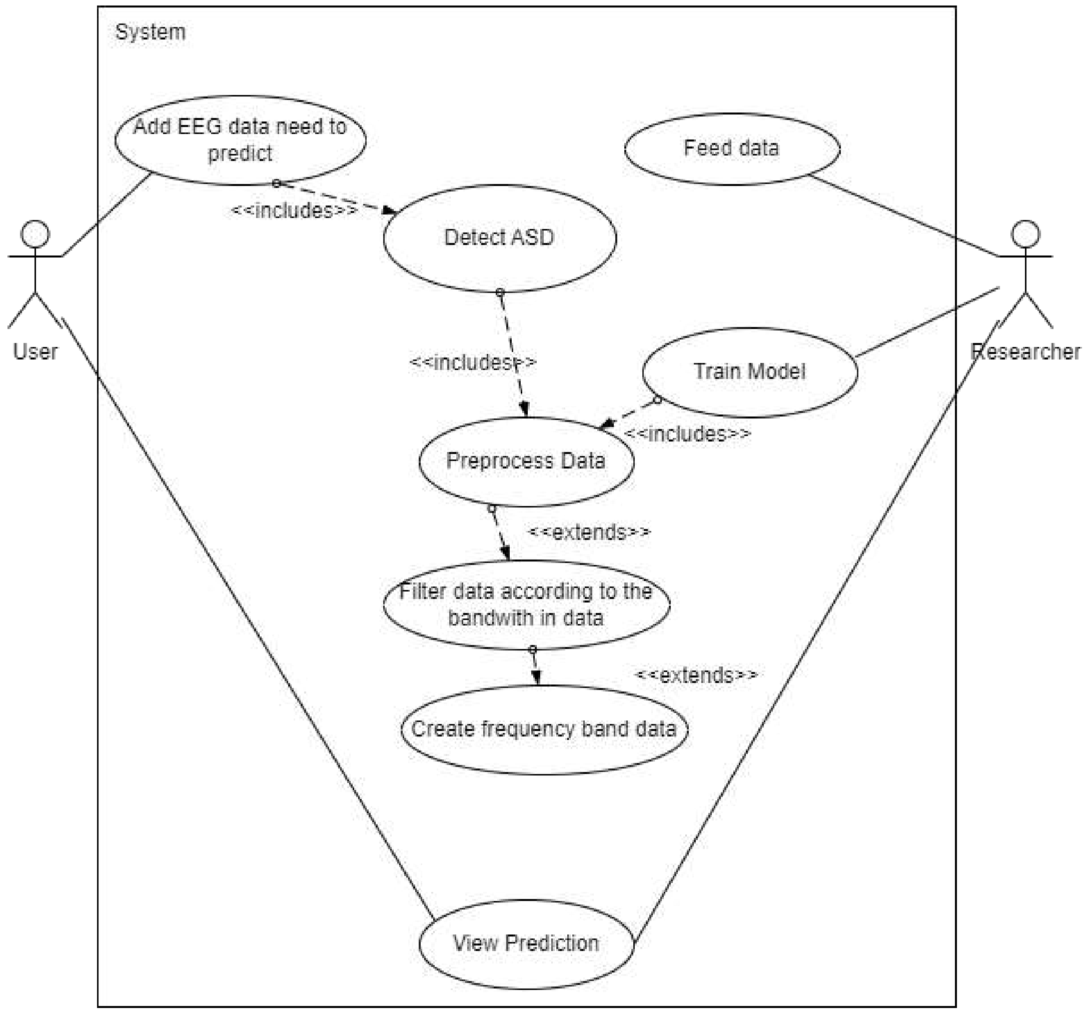


Figure 3.4-Use case diagram

The user is shown as the system's primary stakeholder in the use case diagram. However, based on the nature of the project, a researcher has also been taken into account in this diagram as a stakeholder. Stakeholders will mostly receive two different interfaces depending on their needs.

There are two main tasks this system facilitates its user to engage with. They are adding the EEG data and getting the prediction result of whether the EEG is showing autism spectrum disorder or not. When generating the prediction, the proposed system will clean the EEG data provided by the user, filter the dataset data according to the bandwidth and remove the noise. The frequency band dataset will be created using the pre-processed data which is done using wave decomposition. That frequency band dataset will give the prediction using the model. The researcher can add new data to the system and train the model according to different models. Also, the researcher can view the system's prediction just for research purposes.

3.7 Use case Descriptions

Use case ID	**UC - 03**
Use Case Name	**Train the model**
Priority	High
Actors	Researcher
Descriptions	Use the researcher-fed data to the system and train the model.
Pre- Conditions	The system should be already fed with the data by the researcher
Extended Use Cases	None
Included Use Cases	Pre-process Data
Triggering Event	The researcher selects the model type and starts training and testing the model
Main Flow	The Researcher trains the model
Alternate Flow	None
Exceptional Flow	None
Postconditions	None

Table 3.4-Use case description-Train model

Use case ID	**UC - 04**
Use Case Name	**Pre-process data**
Priority	High
Actors	System
Descriptions	Pre-process the data for training the model and EEG data

	added by the user
Pre- Conditions	The application should be ready to use
Extended Use Cases	Filter data according to the bandwidth of the EEG data
Included Use Cases	Train model Detect ASD
Triggering Event	User import the files to the system and submit
Main Flow	The data passed to detect ASD will get pre-processed
Alternate Flow	When the model gets trained to get the data fed by the researcher to the system and pre-processed them
Exceptional Flow	None
Postconditions	The system should not crash while pre-processing the data

Table 3.5-Use case description-Pre-process Data

Use case ID	**UC - 05**
Use Case Name	**Predict ASD positive or not using neural networks**
Priority	High
Actors	System
Descriptions	Detect ASD
Pre- Conditions	The application should be ready to use
Extended Use Cases	None
Included Use Cases	Pre-process data
Triggering Event	Importing and submitting the files to the system by the user
Main Flow	The system takes the EEG data and passes it to detect ASD,

	The added EEG data get pre-processed Filtered according to the bandwidth
Alternate Flow	If the added EEG data is in bad format invoke an error message
Exceptional Flow	User import non-EEG data
Postconditions	The system should not crash while detecting ASD from EEG data

Table 3.6-Use case description-Predict ASD Positive or Negative

The other use case descriptions can be found in Appendix B: B.1

3.8 Context Diagram – Level 1

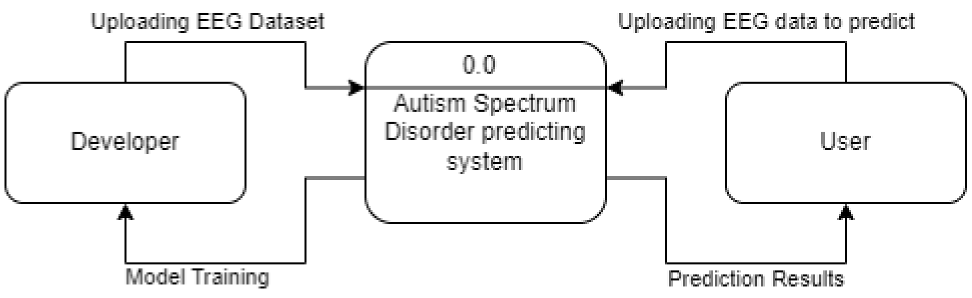

Uploading EEG Dataset
Uploading EEG data to predict
0.0
Autism Spectrum Disorder predicting system
Developer
User
Model Training
Prediction Results

Figure 3.5-Context Diagram

Figure 3:5 shows the context diagram of the proposed autism spectrum disorder classification on EEG data using a neural network system. The context diagram includes 3 main components. Developer, User, and the autism spectrum disorder detecting system. The user represents the individuals or any organization that will use the system to upload EEG data and get the prediction from the system. The developer represents the one who is responsible to upload the EEG data set to the system to create the trained models and maintain the system. The system represents the autism spectrum disorder prediction system which includes the hardware, software and other components that do the task.

3.9 Functional and Non-Functional Requirements

Both Functional and non-functional requirements of this system are identified in tables 3.6 and 3.7. All functional and non-functional requirements are evaluated in importance and mentioned with the appropriate use case. The priority levels can be identified as given below:

- Essential - Core functionalities of the system that must be implemented.
- Desirable - Functionalities which adds value to the system yet are not extremely essential to the proposed system.
- Luxury – Functionalities that are not necessarily required to the system yet these can be implemented in future.

3.9.1 Functional Requirements

ID	Functional Requirement	Priority	Use-Case Number
FR -01	Allows the user to submit EEG data to the system	Essential	UC - 01
FR -02	The system should be able to detect EEG is showing ASD	Essential	UC-04, UC -05
FR -03	Researchers should develop a model to predict ASD for EEG	Essential	UC -03
FR -04	The user and researcher should be able to see the prediction done by the system for the provided EEG data	Essential	UC -08
FR -05	The system should be able to pre-process data and remove the noisy EEG data using filtering	Essential	UC-04, UC -06
FR -06	Create Frequency data set from the pre-processed data	Essential	UC -07

Table 3.7-Functional Requirements

3.9.2 Non-functional Requirements

ID	Non-functional Requirement	Priority	Description
NFR -01	Accuracy	Essential	The system should be capable of providing

			accurate predictions to each provided EEG data and should be able to achieve higher accuracy than the similar research that are already existing
NFR -02	Performance	Essential	The system should respond more quickly and use fewer resources when it is running.
NFR -03	Usability	Desirable	The system should be easy to maintain and configure
NFR -04	Maintainability	Desirable	Future upgrades should be simple to implement in the system without requiring major changes.
NFR -05	Extensibility	Luxury	Future improvements should be simple to implement in the system without requiring major adjustments.

Table 3.8-Non-functional Requirements

3.10 Chapter Summary

In the above chapter the author analyses the stakeholders of this research using an onion model followed by explaining the requirement-gathering techniques. Then the data gathering is discussed where the author explains how the data set used in this research was collected. Next in this chapter author has included the use case diagrams and use case descriptions followed by the domain model. At last, in this chapter, the functional and non-functional requirements are discussed.

Chapter 04 - Project Management Methodology Selection

4.1 Chapter Overview

This chapter elaborates the project management methodology of the proposed solution, following a discussion of research technique. Next the methedologies for Software design and development will be discussed. Then the project management techniques that will work best for this project's development will be stated. At last, this chapter covers the risk mitigation plan along with the ethical and legal issues that need to be kept in mind during the project's development.

4.2 Research Methodology

Research methodology simply refers to the way by which a researcher plans a study. Two main categories of research methodologies can mainly be identified, and they are deductive and inductive. Based on an existing theory, a hypothesis is developed in the deductive research technique, which is followed by the creation and development of a research strategy to test the given hypothesis. The inductive approach is more distinctive because it will put out brand-new theories considering the results of the investigation.

This research is following deductive research methodology because this research already has an existing theory. Here the author tries to identify and improve early-age autism spectrum disorder detection using electroencephalogram data using neural networks

4.3 Software Design Methodology

Software design methodology offers a methodical and logical way to move forward with the design process as well as a set of rules for making decisions. The three primary types of software design approaches are the structured design approach, the object-oriented approach, and the functional-oriented approach.

The proposed solution is conceived into several orderly components using the divide and conquer technique in the structured design approach. This simplifies the problem-solving process by breaking the solution down into smaller parts. The solution's small modules are organized hierarchically to enable communication between them. Because of this organized design process, the arrangements are thought to have strong cohesiveness and low coupling.

The function-oriented design is made up of numerous smaller sub-systems called functions. These operations can carry out important system tasks. The system is viewed as the overall perspective of all operations. Some characteristics of structured design, which use the divide and conquer strategy, are carried over into function-oriented design. By dividing the entire system into smaller functions, this design strategy offers a means of abstraction by hiding the information and how they work. These functional modules can communicate with one another and use globally accessible data to share information. Because this approach system contains more components and non-modularized system design is quite hard to catch on to.

Instead of focusing on the functionalities that are present in the software system, object-oriented design revolves around the entities and their traits. This design strategy emphasizes entities and their traits. The engaged entities are crucial to the idea of a software solution. The object-oriented design approach follows mainly three characteristics inheritance, polymorphism, and encapsulation. In inheritance OOD allows similar classes to stack up hierarchically. In encapsulation, the attributes (data variables) and methods (operation on the data) are bundled together into each object. In polymorphism, OOD languages provide a mechanism where methods performing similar tasks but varying in arguments, can be assigned the same name. Also, the components of the application are called objects and they can stand alone or can depend on each other. The object-oriented approach is easy to understand and more convenient when performing testing on the system.

Based on the advantages such as ease of implementation, test and the nature of the project such as the creation of two different API endpoints initially to upload EEG data for the user and give predictions; the suggested system would employ the function-oriented design method after analysing the previously mentioned design techniques.

4.4 Software Development Methodology

It is essential to have a well-managed software development technique for a project to be successful. The project manager or the development team should select the software development technique. But in this case, the author, who is a software developer, will choose the best software development technique. There are just a few approaches for software development that can be covered here, but each has unique benefits and limitations. Several well-known software development approaches are listed below.

- **Agile development methodology** - When implementing new features, teams employ the agile development technique to reduce risk. Teams create software in iterations that include tiny increments of new functionality according to all agile development methodologies. The agile development methodology comes in a variety of forms, including scrum, extreme programming, Test driven development and feature-driven development. Agile software development's main advantage is that it enables iterative software releases. Iterative releases increase productivity by enabling teams to identify and correct flaws and set expectations early on. With regular incremental enhancements, they also enable consumers to enjoy the benefits of software earlier. But Agile development methodologies focus on real-time communication, thus new users frequently lack the necessary background information to get started. They are time- and labour-intensive because developers must finish each feature inside each iteration before asking users for approval.
- **Waterfall development methodology**- The waterfall method is frequently regarded as the oldest approach to software development. The requirements, design, implementation, verification, and maintenance phases of the waterfall technique each focus on a different objective. Before moving on to the following phase, each phase must be finished. In most cases, there is no procedure for going back and changing the project or direction. The waterfall development process is simple to comprehend and administer due to its linear character. The waterfall method works well for projects with precise goals and consistent criteria. The waterfall development method is often slow and costly due to its rigid structure and tight controls.
- **Prototyping development methodology** - The prototype model and the waterfall model are quite similar, although the prototype model is not linear. There is no direct relationship between the design and development phases; rather, they go back and forth in a cycle. As a result, the prototype will be constructed progressively. After each cycle, the prototype can be assessed to see whether it has met the criteria or if any requirements are still lacking. The user can also offer suggestions for improving the features of the built-in prototype.
- **Rapid application development methodolog**y - Component-based construction benefits from the rapid application development concept. This approach places less emphasis on

planning and more emphasis on the quick evolution of the process. RAD. is better suited for tasks with clear objectives. Additionally, this approach needs developers that are very educated in the chosen technological field. The most successful projects for rapid application development are those that have a well-defined business aim and a defined user group and are not computationally demanding.

After careful analysis of the software development methodologies mentioned above, the prototype method was chosen to be used in this project. This is selected since the author of this study is creating a product which revolves around a prototype. Additionally, based on the prototype build, additional requirements and flaws will be found and then can be included in the next build of the prototype to help it accomplish its stated goals. Also, since this is based on neural network models to achieve the non-functional requirements such as accuracy and performance prototype model is most suitable.

4.5 Project Management Methodology

Project management techniques are used to manage constraints like scope, money, and time to provide a high-quality outcome from the project. PRINCE2 is one of the project managements approaches that we might employ. A popular project management approach is called PRINCE2 (PRojects IN Controlled Environments). According to PRINCE2, there are several steps to take and data that needs to be gathered. Stakeholder interaction is incorporated into the PRINCE2 methodology at an early stage. One of the key advantages of PRINCE2 is that it can be improved continuously as a result of lessons learned, and this methodology also includes a defined approach to guarantee the project's success.

Six Sigma and Agile are two more project management approaches, none of which we selected for our project. The Six Sigma method is better suited for large-scale projects and calls for specialized training in the methodology. It might be difficult to determine a project's progress using the agile methodology. The PRINCE2 approach has been selected for this project as the project management methodology due to the drawbacks of the Six Sigma and Agile methodologies.

4.6 Risk Mitigation Plan

Risks are uncertain possibilities that could arise when working on a project. These hazards may affect the project in both favourable and unfavourable ways. As a result, whenever a risk arises during the project process, having a risk mitigation plan in place is crucial. A risk mitigation plan outlines both potential risks that could occur during the project as well as a strategy for reducing each risk that has been identified. The project's risk reduction strategy is shown in the table.

Risk 01	**Change of Requirements**		
Risk Level	High	**Frequency**	High
Description	It is like research projects to experience ongoing demand changes. Where the developed needs will be implemented will be decided for each section of the research. Even when the requirement alteration is anticipated, it must nevertheless change throughout the study.		
Mitigation Plan	● Prioritize critical requirements ● If a requirement change comes prioritize it and adds allocate time for it according to the required size ● Always prioritize the critical requirements according to the importance and the time that they need to be done ● Be prepared for adjustments in advance.		
Risk 2	Failing to update technical skills and domain knowledge		
Risk Level	High	**Frequency**	High
Description	As numerous studies are being done in the field, the project's chosen region changes from time to time. As a result, it can be challenging to stay up to date with the domain and related technological expertise as it is explored.		
Mitigation Plan	● Reading up on recent studies in the field, being informed, and maintaining your expertise ● staying up to date by reading about new technological breakthroughs. ● Maintain regular contact with domain experts to be informed of the most recent advancements in the industry.		

Risk 3	Lack of familiarity with cutting-edge resources and technology that could be employed with in the project		
Risk Level	Medium	**Frequency**	High
Description	There is a risk until the author is at ease using the new tools and technologies.		
Mitigation Plan	● To become comfortable with new tools and technologies, read about them and use them. ● Try to follow the documentation of the certain techniques that you want to learn		
Risk 04	Containing bugs in the system		
Risk Level	High	**Frequency**	High
Description	There may be issues in the final prototype that weren't found throughout the implementation and testing phases.		
Mitigation Plan	● Use recommended practices when designing the system. ● From the beginning of the project, create unit test cases for each minor component. (Follow a Test-driven development approach)		
Risk - 05	Not being able to achieve the expected accuracy		
Risk Level	High	**Frequency**	Medium
Description	The prototype's accuracy might not be sufficient enough as the expected accuracy		
Mitigation Plan	● Keep a selection of solutions on hand as a fallback. ● Early prototype implementation completion allows for the possibility of small alterations. ● Increase accuracy by conducting additional analyses, and indicating them in the evaluation.		

Risk - 06	Hardware Software Failure		
Risk Level	Medium	**Frequency**	Low
Description	Both unexpected hardware failures and data loss are possible.		
Mitigation Plan	● Maintain an additional machine with the same specifications and configurations ● Documentation of all configurations used during system installation.		
Risk -07	Time constraints in completing the project on time		
Risk Level	Medium	**Frequency**	Low
Description	The time that should be allotted to the project can be limited by unexpected illnesses or other pressing tasks such as migration to another country or a lot of workloads from the workplace		
Mitigation Plan	● Follow the schedule strictly. ● When planning the project, allot more time. ● Request for mitigation if they cannot complete the project on time		

Table 4.1-Risk Mitigation Plan

4.7 Legal and Ethical Considerations

This chapter focuses on the legal, ethical, and professional concerns that may arise throughout the development of the project.

4.7.1 Legal considerations

Project-related legal considerations are found in areas like data, tools and third-party libraries. Legal considerations will have an impact on the data because the research will use EEG data of patients. So, when collecting the data or requesting data from a professor who has done research in the same domain it needs to discuss all the details through email so in an event of legal issue can show the proof that you have requested the data set for this research and the professor had given the access to use that dataset. But the developer shouldn't save or improperly use the user's data when it comes to the application's users. As a result, the suggested system will not keep usernames or any other private information about users.

Additionally, when choosing tools and libraries for the development of the project, the author must make sure that all third-party libraries, IDEs, tools, frameworks, and languages adhere to open-source licensing.

4.7.2 Ethical Considerations

Ethics impose a set of standards on organizations and development projects, resulting in benefits and safety. Violations of ethics may have negative legal repercussions, and be disastrous for the organization or enterprise related this project and to the author. When domain experts provide feedback, the author must make sure that the person's full consent was obtained before the feedback was given. In addition, the author must ensure that none of the project-related documents contains any copied material. If utilized, the author must offer appropriate attribution or ensure that the information is correctly cited in the works.

4.7.3 Professional Considerations

Both the development and the management teams should adhere to professional standards throughout the project. This stands true for the author of this project. When finishing the project, the author must adhere to the BSC code of conduct. To abide by the Code of Conduct, the developers must be highly skilled, committed to the success of the project and also eager to learn. Many methods of research, design and development can be explored to perform professionally. After extensive research to identify the tools and libraries that are both suited for the project and make the lives of developers and management easier, implementations are carried out. These resources can be accessed through legal channels. All of the project's sensitive information should be stored safely without creating any entry points for data breaches from the perspective of data security and vulnerability. No one should have altered or changed any of the recorded results. False information should not be included in any project-related documents of any kind. And any professional documentation format should be used in those documents.

4.8 Chapter Summary

The project's possible methodologies were discussed in this chapter, and the best strategies were chosen from among them. This chapter justifies the project management technique, software development design methodologies, and research methodology choices in a small chapter. The risk mitigation plan is made in addition to that. Finally, it is highlighted how legal, ethical, and professional issues should be taken into account throughout the project.

Chapter 05 - Design

5.1 Chapter Overview

The design and architecture of the system will be covered in this chapter. The justification for the selected analytic and design techniques, as well as constraints and specific architectural aims, will next be discussed. The following section of this chapter will examine low-level and high-level architectural models using sequence diagrams and class diagrams.

5.2 Block Diagram

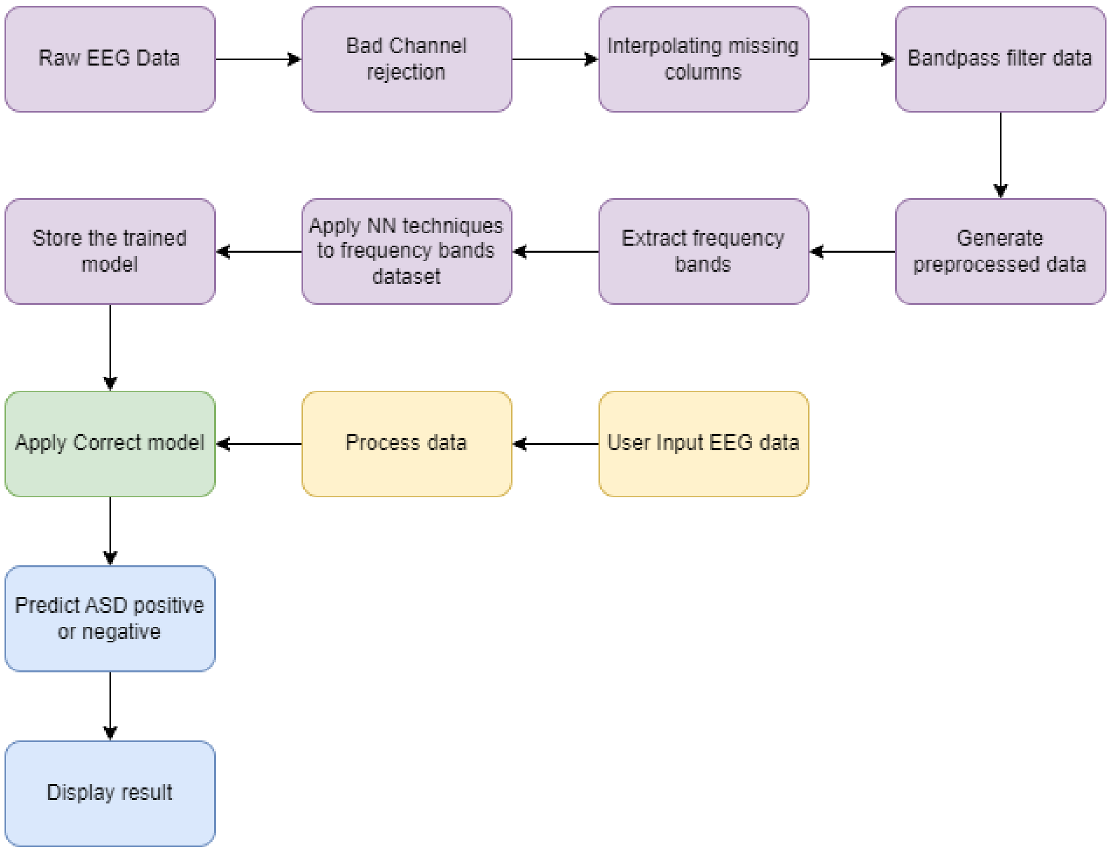


Figure 5.1-Block diagram of the system

As per the diagram above (i.e., figure 5.1) first get the raw EEG data for each participant and combine them into a single data frame then that data need to be pre-processed. Pre-processing EEG data is done in steps. First, need to remove the bad channels, in our dataset, there are some columns which contain the Not a number (NaN) data which happens due to the way electrode placement is done while collecting the EEG data. Then need to interpolate the missing columns for the EEG data. EEG data is noisy due to various reasons, so the author needs to collect only the EEG data in the correct frequency without the noisy ones and needs to send the data through a bandpass filter and filter those data. After pre-processing the data need to extract the frequency bands of the data and use those frequency band labelled data to train the model in different neural network techniques and generate a model which can predict when the user inputs EEG data whether that EEG data shows autism spectrum disorder or not. Finally, show the predicted output to the user who entered EEG data.

5.3 High-Level Architecture

High-level architecture refers to the fundamental layout of a system's high-level modules and components, as well as the relationships between them. The author here chose to adopt the three-tier architecture out of all the available design architectures. The explanation for not choosing the three-tier design over the other architecture types is given below.

- Microservices - The microservices architecture is effective because of the decoupling of each service and the independence that enables any of the services to be upgraded or replaced independently. However, this design is more frequently utilized in large projects that include numerous independent phases.
- Event-driven architecture - It might be challenging to maintain and test standalone and isolated modules in an event-driven architecture. Due to reuse and adaptability, this architecture is inappropriate.
- Tiered architecture - The main advantage of a three-tier architecture is that each tier may be built concurrently by a different development team and can be updated or scaled as necessary without affecting the other tiers because each tier runs on its infrastructure.
- Layered Architecture - The suggested solution works best with the layered for the data to be maintained well, made adaptably, and scaled to various levels in the design, layered architecture is required. Different parts of solutions are developed so that interoperations can be made with various implementation levels. This decision was made mostly because components in a tiered design are physically separated. Additionally, this project does not require the physical separation of its components. The logical separation of components is referred to as layering.

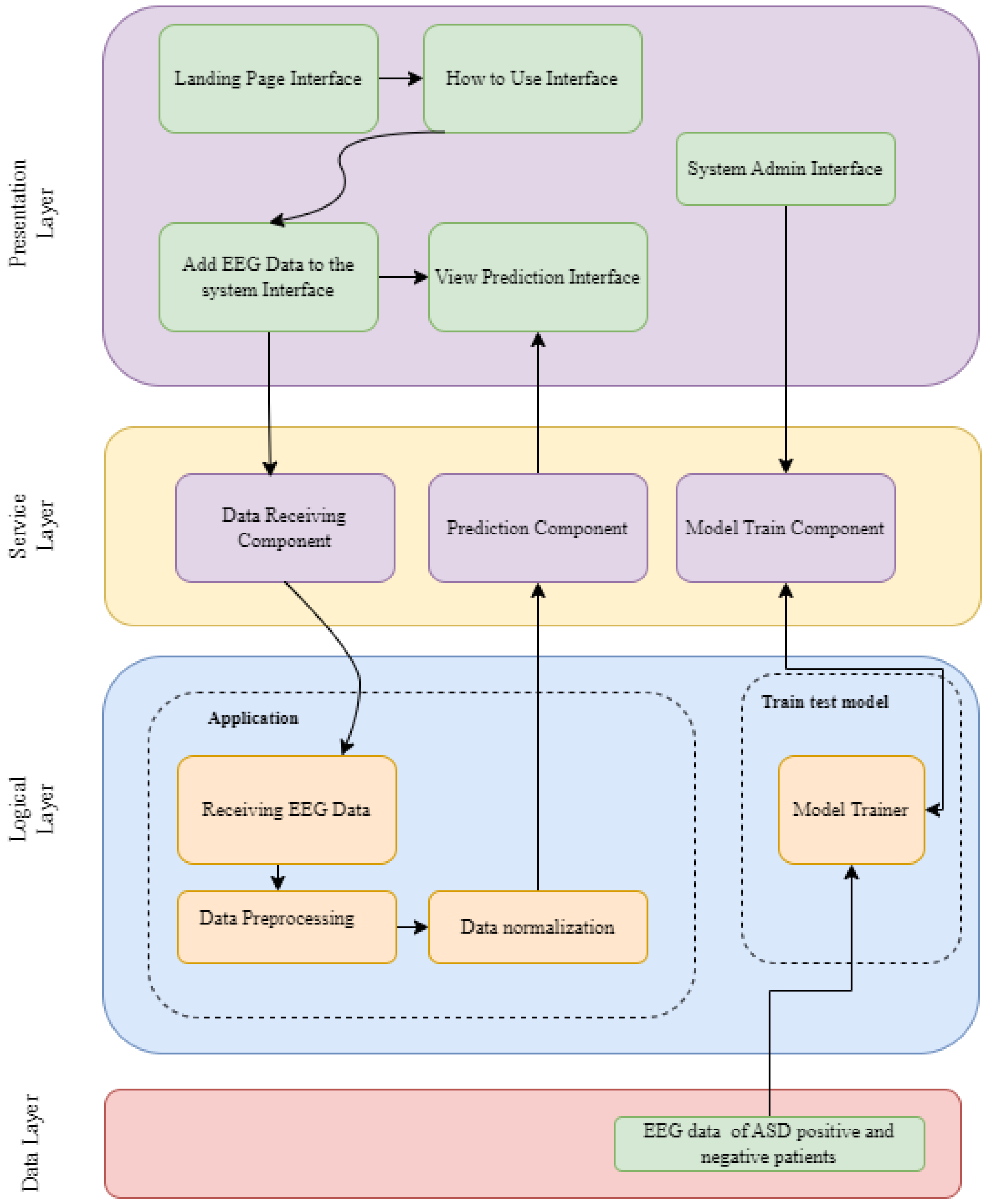


Figure 5.2-High level architecture of the system

Figure 5.2 will show the suggested system's layered architecture based on the previous justification. The presentation layer, the service layer, the logical layer and the data layer are the three components that make up the architecture of the suggested solution. In the sections that follow, the levels and components of the high-level architecture depicted in Figure x are briefly discussed.

5.3.1 Presentation Layer

This layer serves as the system interface that users interact with. General All user-oriented functionalities that control how users interact with the system are contained in the presentation layer of the high-level architecture.

- User interface - The user is directly engaged with the proposed system with this layer/ In this layer there are a few interfaces that the user can interact with among them two main interfaces are
    - Add EEG data - The user can use this interface to add the EEG data that he/she needs to get the prediction. This will directly communicate with the preceding layer.
    - View Prediction - The user can use this interface to see the prediction results of the EEG data that he inputs to the system.
- System Admin interface - This interface is for the system administrator or in this case the researcher to train the model if there is a dataset updated in the data layer

5.3.2 Service Layer

This layer works as an intermediate layer that communicates with the presentation layer and the business layer. This layer contains the business logic and gives the front the necessary features. This layer also makes it possible for the system to be connected to other systems. This layer consists of three main components.

- Data receiving component - Users will be able to upload the EEG data to the system that needs pre-processing and other tasks that need to be done to the raw EEG by servicing an API
- Prediction component - The API is used to users the predicted ASD positive or negative results.
- Model train component - This API is used by the system administrator to send what kind of model he needs to train and get the test results of the trained model

5.3.3 Logical Layer

All the system's essential features are contained in this layer. This layer controls data sources and interacts with the persistence layer. There are two modules in this layer.

- Application - This module maintains the core functionality of the application, taking EEG data. and pre-process the data and normalize and after sending it to the prediction component that will do the prediction
- Training module - This module will create a neural network model which can predict whether EEG data is showing autism spectrum disorder using EEG data and a labelled data set of the participants.

5.3.4 Data Layer

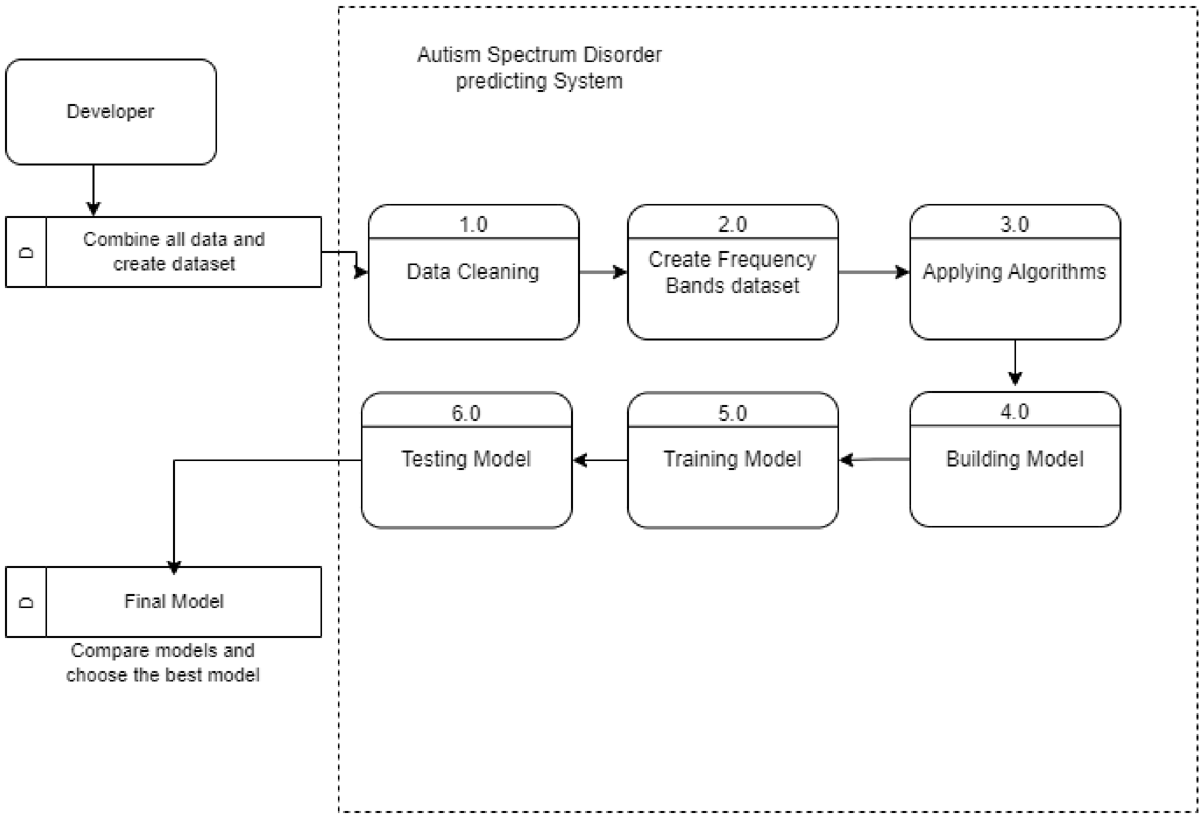
The system-related data sources are located in this layer. The main data source will be the EEG data set from data set collected by Dr Mark Jaime from Indiana University-Purdue University Columbus (IUPUC) which helps to train the model.

5.4 System Process Flowchart

The algorithm's flow and decision structures are depicted in the flowchart below. This flow chart explains most of the system because it is primarily procedural. Figure 3.5 shows the data flow diagram in level 1 which is a context diagram and below discussed is the level 2 diagram which is a container diagram.

5.4.1 Level 2 – Data Flow

The level-2 Data flow diagram in figure 5.3 will show the six containers inside the autism spectrum disorder predicting system and each component will do different tasks. The data cleaning component will clean the EEG data by removing noise in the data. Creating a frequency bands container will create the frequency bands data set from EEG data. Applying algorithms container will apply the algorithms for the frequency bands dataset. Building a model container will build the different neural network models. The training model container will handle the model training and the Testing model container will handle the testing of the trained model.



Autism Spectrum Disorder predicting System
Developer
D
Combine all data and create dataset
1.0
Data Cleaning
2.0
Create Frequency Bands dataset
3.0
Applying Algorithms
4.0
Building Model
5.0
Training Model
6.0
Testing Model
D
Final Model
Compare models and choose the best model

Figure 5.3-Data Flow-Level 2

5.5 Sequence Diagram

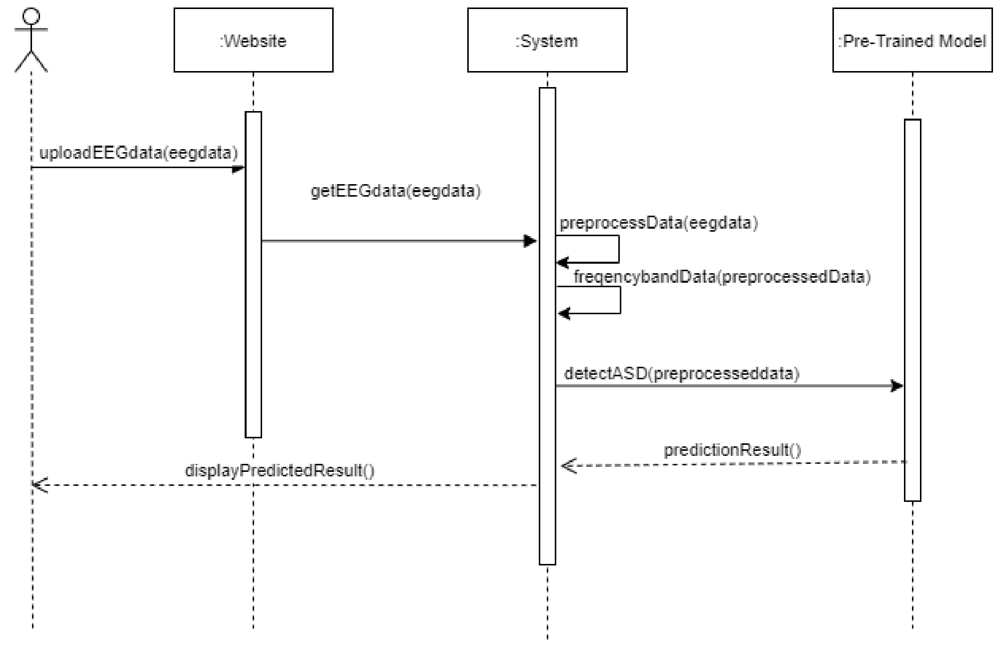


Figure 5.4-Sequence diagram

The above figure 5.5 shows the sequence diagram for the whole system, and it is showing the happy path of the application. As described in the sequence diagram the user can be observed as the only actor in the system. Users access the system as a web application to know about the corresponding EEG data showing autism spectrum disorder. The first user needs to upload EEG data to the system. Then the uploaded EEG data will get processed in the system in a way where it can be applied to the pre-trained model to get the prediction. Then the pre-trained model will give a prediction of whether the EEG data shows ASD or not. That result will be displayed back to the user.

5.6 Activity Diagram

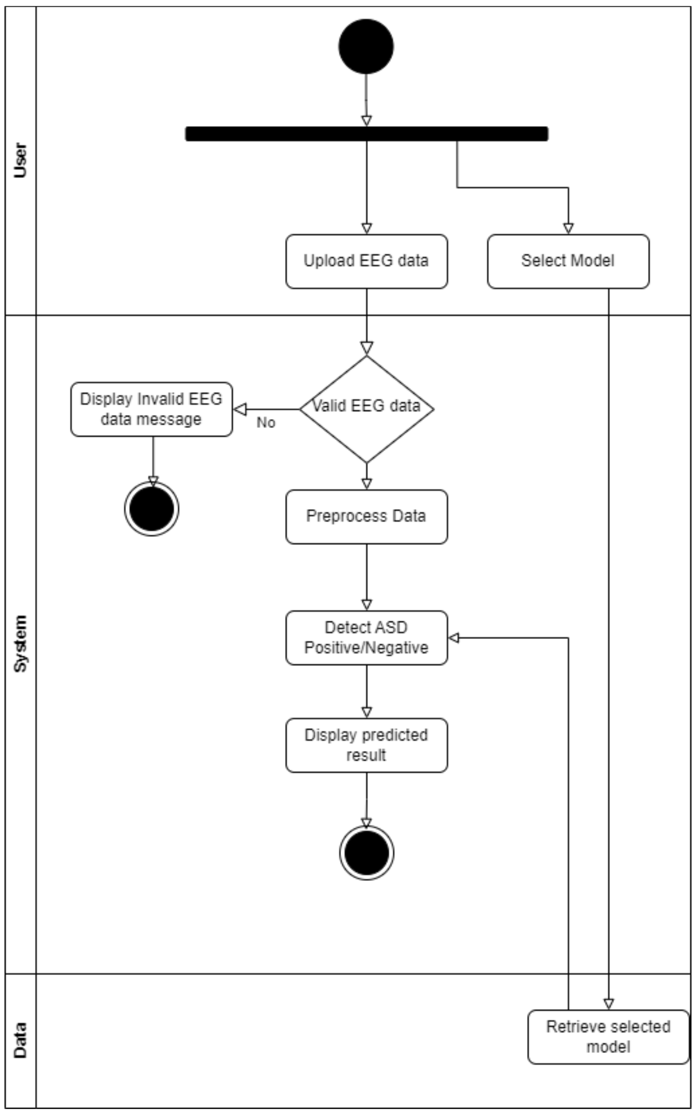


Figure 5.5-Activity Diagram

The above diagram (i.e., Figure 5.6) shows the algorithm of the system. First, the user will upload the EEG data files and then the system will check whether the EEG data files are valid or not. If they are invalid system will prompt a message saying the uploaded EEG data is invalid. If they are valid, they will be processed, then the user can select which model he/she will use to detect whether the EEG data shows autism spectrum disorder after model selection;

the model will get retrieved from the data and to the prediction and display the predicted output to the user.

5.7 UI Wireframe

The wireframes were developed to show how the model works. UI is a web application in which users can input EEG data and check whether it shows ASD-positive or ASD-negative please refer to APPENDIX C: C.1 for the UI wireframe.

5.8 Chapter Summary

This chapter covered the design of the system using several diagrams. First, the system's components and their interactions were depicted in the block diagram and high-level architectural diagram. Then a class diagram was used to display the design of each crucial module. A sequence diagram was then created to depict the sequence of system occurrences. Lastly, the activity diagram was developed to illustrate the functionalities of the system.

Chapter 06 - Implementation

6.1 Chapter Overview

This chapter will cover the libraries, frameworks, and IDEs that have been used to build the implementation of the project. Not only that but also the choice of the programming language along with the justifications. Furthermore, the here author will discuss the implementation with the screenshots and the code snippets and algorithms that have been used to test the model so far. In this chapter, the author shows how the design idea transforms into the code.

6.2 Technology Selection

6.2.1 Programming Language Selection

When deciding on a programming language for the development, the developer is required to recognize the main technologies that are going to be needed for implementing the system. For the system proposed here, neuroscience techniques will be mostly used. The table in Appendix D: D.1 compares the potentialities of different programming languages when using the neuroscience approaches. In the field of neuroscience, however, the most common languages you're likely to come across are R, MATLAB, and Python. The most suitable language for training the models and testing the models will be determined by comparing the languages mentioned and discussed in the table in Appendix D: D.1 Program Language Selection.

In comparison to Python, MATLAB and R are recognised as better suited for statistics and metrics purposes. Furthermore, the author only has a basic level of familiarity with those two languages. To implement a model using neural networks to classify the EEG data most specialists in neuroscience choose Python. This is due to the fact that python comes with a variety of packages and open-source libraries. Additionally,this author has previous experience in working with Python and thereby overcoming possible obstacles that come up during environment setup. Furthermore implementation will be easier with the strong python related community support. On top of that, the below mentioned advantages of the use of python has been reasons for previous researches to recommend its use. MEG and EEG data analysis with MNE-Python by Gramfort and Luessi proved that using the MNE python library consists of all algorithms and utility functions being built consistently and having clear interfaces, users can create M/EEG data analysis pipelines by writing Python scripts. Furthermore, MNE-Python is closely linked to the larger Python neuroimaging ecosystem, as well as key Python libraries for scientific computation (NumPy, SciPy), visualization (matplotlib and seaborn), and analysis (NumPy, SciPy). Based on all the above grounds, Python was chosen as the programming language in developing this prototype.

6.2.2 Selection of Libraries

In Appendix D: D.2 Libraries Selected for the Prototype the author is deciding the libraries that can use to read data, pre-process, train models, test, and visualization the EEG data set.

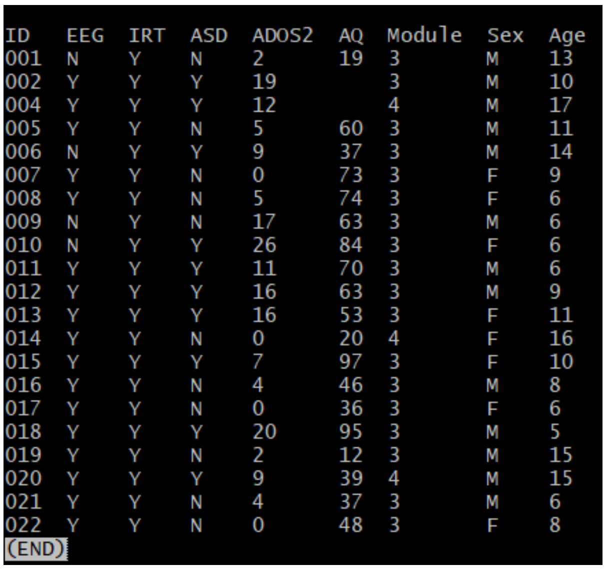
The mentioned libraries and their versions are specifically selected to work in the Python environment without causing any library mismatch issues or functionality not supporting the issue. Also, all the libraries used are stable and they have a strong support community. Apart from the above reasons these libraries are open-source and available for free use.

6.2.3 Data Selection

The author's research is early age autism spectrum disorder classification on electroencephalogram data using neural networks. There are several EEG datasets available online, but it is hard to find an EEG data set of early-age children because it comes with private and confidential data. So, the author found a data set collected by Dr Mark Jaime from Indiana University-Purdue University Columbus (IUPUC) author was able to contact Prof.Sampath Jayarathna from Old Dominion University Norfolk. Get the data set. This dataset consists of seventeen subjects.

6.3 Dataset Implementation

In this dataset following details can be found. Each participant has a unique {ID} and their recordings are grouped into folders by their {ID}. The EEG recordings are stored at eeg/{ID}. These { ID } s range from 1 - 22, yet some {ID}s are ignored due to errors in data collection.



```
ID   EEG  IRT  ASD  ADOS2  AQ  Module  Sex  Age
001  N    Y    N    2      19  3       M    13
002  Y    Y    Y    19         3       M    10
004  Y    Y    Y    12         4       M    17
005  Y    Y    N    5      60  3       M    11
006  N    Y    Y    9      37  3       M    14
007  Y    Y    N    0      73  3       F    9
008  Y    Y    N    5      74  3       F    6
009  N    Y    N    17     63  3       M    6
010  N    Y    Y    26     84  3       F    6
011  Y    Y    Y    11     70  3       M    6
012  Y    Y    Y    16     63  3       M    9
013  Y    Y    Y    16     53  3       F    11
014  Y    Y    N    0      20  4       F    16
015  Y    Y    Y    7      97  3       F    10
016  Y    Y    N    4      46  3       M    8
017  Y    Y    N    0      36  3       F    6
018  Y    Y    Y    20     95  3       M    5
019  Y    Y    N    2      12  3       M    15
020  Y    Y    Y    9      39  4       M    15
021  Y    Y    N    4      37  3       M    6
022  Y    Y    N    0      48  3       F    8
(END)
```

Figure 6.1-Participant data

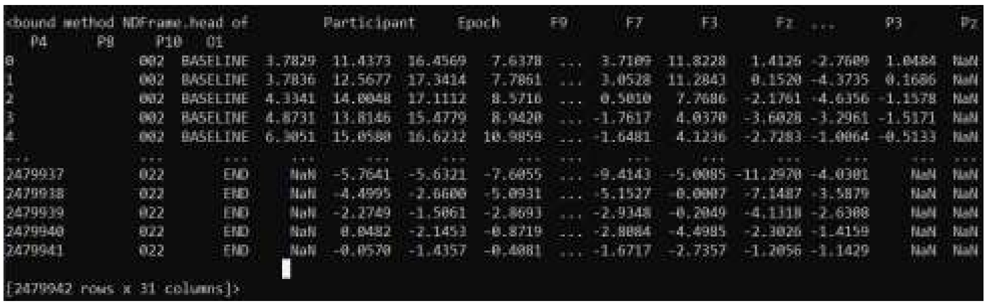
Each participant is described using the following fields as shown in figure 6.1

- ID - The participant's ID
- Diagnosis - The clinical ASD diagnosis of the participant
- Sex - Gender of participant
- Age - The age of the participant
- AQ - Autism quotient of the participant (a self-assessed measure of ASD traits)
- ADOS - Assessed ADOS-2 score of the participant
- Module - The ADOS-2 module used to assess the participant (differs by age)

The EEG recording of each participant are split across four files

- eeg/{ID}/{ID}_BASELINE.csv (Taken before the experiment to capture their resting state)
- eeg/{ID}/{ID}_START.csv (Taken during phase 1 of ADOS-2)
- eeg/{ID}/{ID}_MIDDLE.csv (Taken during phase 2 of ADOS-2)
- eeg/{ID}/{ID}_END.csv (Taken during phase 3 of ADOS-2)

After collecting every participant every recording into one data frame EEG data set consists of 2479942 rows as the image below. (i.e figure 6.2) They used a 32-channel LiveAmp wireless EEG system with active electrodes and a digital sampling rate of 250 Hz for EEG time series acquisition.



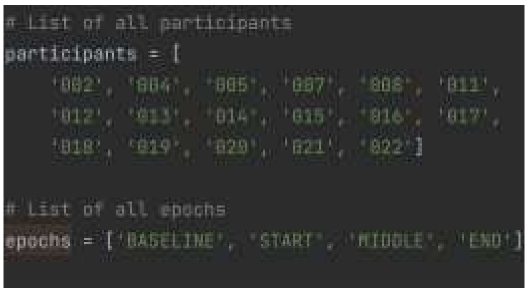
```
<bound method NDFrame.head of          Participant     Epoch       F9        F7        F3        Fz  ...       P3      Pz
   P4       P8      P10   O1
0             002  BASELINE  3.7829  11.4373  16.4569   7.6378  ...   3.7109  11.8228   1.4126 -2.7609  1.0484  NaN
1             002  BASELINE  3.7836  12.5677  17.3414   7.7861  ...   3.0528  11.2843   0.1520 -4.3735  0.1686  NaN
2             002  BASELINE  4.3341  14.0048  17.1112   8.5716  ...   0.5010   7.7686  -2.1761 -4.6356 -1.1578  NaN
3             002  BASELINE  4.8731  13.8146  15.4779   8.9420  ...  -1.7617   4.0370  -3.6028 -3.2961 -1.5171  NaN
4             002  BASELINE  6.3051  15.0580  16.6232  10.9859  ...  -1.6481   4.1236  -2.7283 -1.0064 -0.5133  NaN
...           ...       ...     ...      ...      ...      ...  ...      ...      ...      ...     ...     ...  ...
2479937       022       END     NaN  -5.7641  -5.6321  -7.6055  ...  -9.4143  -5.0085 -11.2970 -4.0301     NaN  NaN
2479938       022       END     NaN  -4.4995  -2.6600  -5.0931  ...  -5.1527  -0.0007  -7.1487 -3.5879     NaN  NaN
2479939       022       END     NaN  -2.2749  -1.5061  -2.8693  ...  -2.9348  -0.2049  -4.1318 -2.6308     NaN  NaN
2479940       022       END     NaN   0.0482  -2.1453  -0.8719  ...  -2.8084  -4.4985  -2.3026 -1.4159     NaN  NaN
2479941       022       END     NaN  -0.0570  -1.4357  -0.4081  ...  -1.6717  -2.7357  -1.2056 -1.1429     NaN  NaN

[2479942 rows x 31 columns]>
```

Figure 6.2-EEG data before cleaning

The EEG data have been collected from the electrode placements as shown in Appendix D: D.3 figure.

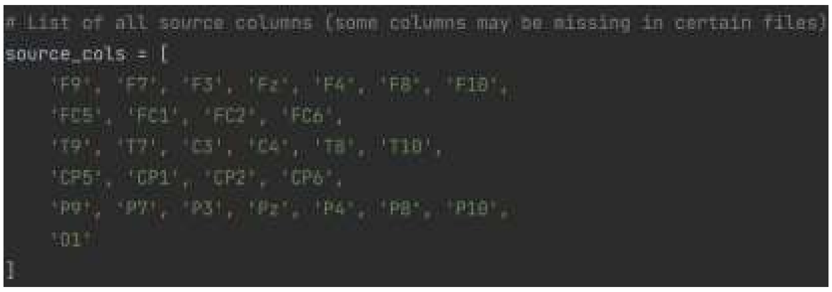
6.3.1 Read and Combine Data

In our dataset for each participant, there are 4 EEG data files, first, the author read all the files and combines them into one file.



```
# List of all participants
participants = [
    '002', '004', '005', '007', '008', '011',
    '012', '013', '014', '015', '016', '017',
    '018', '019', '020', '021', '022']

# List of all epochs
epochs = ['BASELINE', 'START', 'MIDDLE', 'END']
```

Figure 6.3-Information about participants and epochs



```
# List of all source columns (some columns may be missing in certain files)
source_cols = [
    'F9', 'F7', 'F3', 'Fz', 'F4', 'F8', 'F10',
    'FC5', 'FC1', 'FC2', 'FC6',
    'T9', 'T7', 'C3', 'C4', 'T8', 'T10',
    'CP5', 'CP1', 'CP2', 'CP6',
    'P9', 'P7', 'P3', 'Pz', 'P4', 'P8', 'P10',
    'O1'
]
```

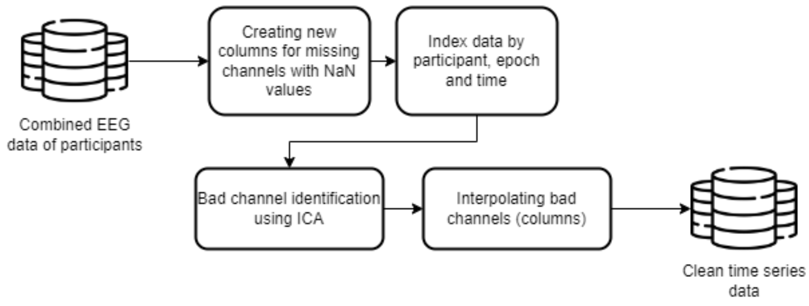
Figure 6.4-Source columns in the dataset

To add all the data to a single file. First, the author creates static variables in the "info.py" file that contains the details of figure 6.3 and figure 6.4 which are participant IDs, epochs, and source columns. Next import those details and the Pandas library to our data-reading python file.

Next to create a full data file; need a data frame to hold all the data so the data frame will make in a way which has the participant ID, epoch and then the source columns. Then read the data in each file and append them to the initially created data frame and finally save the data frame as a “. ftr” extension file, which can be read as a feather file. The coding snippet for the reading and combining all files are shown in Appendix D: D.7

6.3.2 Pre-process Data

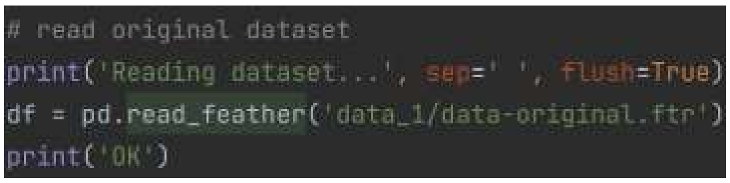
Raw EEG data contains a lot of noise as the author discussed in the possible noise-removing approaches section in the literature review. The author follows the following pipeline to pre-process the raw EEG data as shown in figure 6.5



Combined EEG data of participants
Creating new columns for missing channels with NaN values
Index data by participant, epoch and time
Bad channel identification using ICA
Interpolating bad channels (columns)
Clean time series data

Figure 6.5-EEG data pre-process pipeline

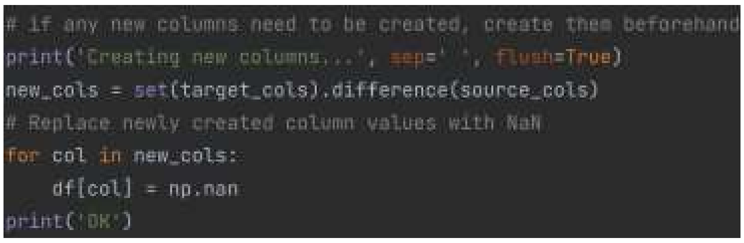
First, load the combined data of EEG participants from the feather file that have the original data saved in the combined data part as a Pandas data frame as shown in figure 6.6.



```
# read original dataset
print('Reading dataset...', sep=' ', flush=True)
df = pd.read_feather('data_1/data-original.ftr')
print('OK')
```

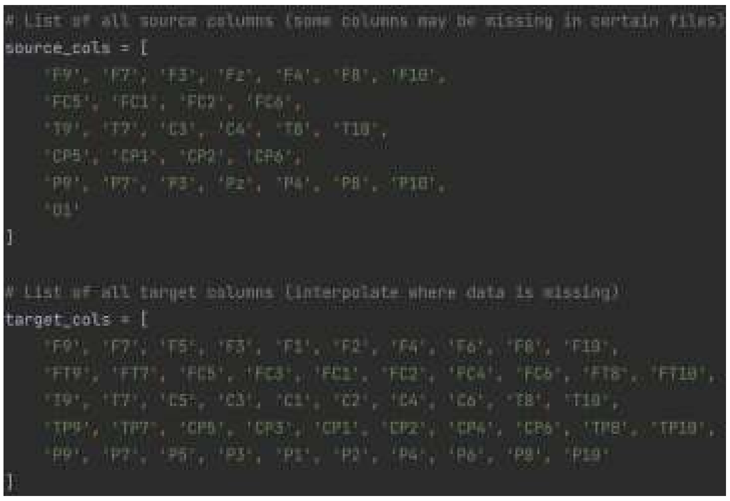
Figure 6.6-Load combined original EEG data feather file to the data frame

Then create the new columns if needed beforehand and fill those columns with NumPy NaN values, this is done because EEG data is saved as float32 type and NaN values in NumPy are working as float32 data even though they don’t contain numbers. When creating a new column, the author will check the difference between (i.e Figure 6.7) target_cols and source_cols (i.e Figure 6.8) which are two constant variables. This is done using the “mne” python package which uses standard 50-channel data processing.



```
# if any new columns need to be created, create them beforehand
print('Creating new columns...', sep=' ', flush=True)
new_cols = set(target_cols).difference(source_cols)
# Replace newly created column values with NaN
for col in new_cols:
    df[col] = np.nan
print('OK')
```

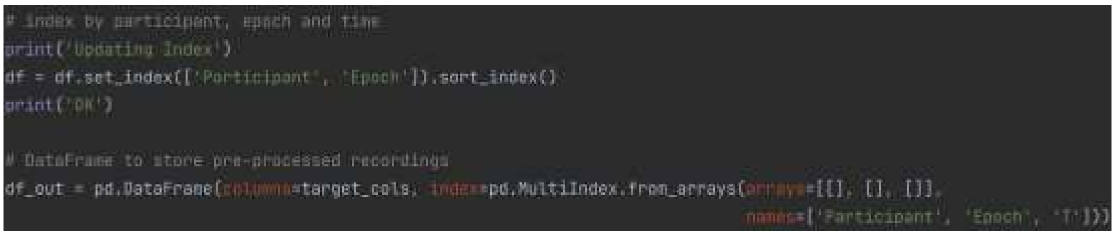
Figure 6.7-Create missing columns and fill them with NaN values



```
# List of all source columns (some columns may be missing in certain files)
source_cols = [
    'F9', 'F7', 'F3', 'Fz', 'F4', 'F8', 'F10',
    'FC5', 'FC1', 'FC2', 'FC6',
    'T9', 'T7', 'C3', 'C4', 'T8', 'T10',
    'CP5', 'CP1', 'CP2', 'CP6',
    'P9', 'P7', 'P3', 'Pz', 'P4', 'P8', 'P10',
    'O1'
]

# List of all target columns (interpolate where data is missing)
target_cols = [
    'F9', 'F7', 'F5', 'F3', 'F1', 'F2', 'F4', 'F6', 'F8', 'F10',
    'FT9', 'FT7', 'FC5', 'FC3', 'FC1', 'FC2', 'FC4', 'FC6', 'FT8', 'FT10',
    'T9', 'T7', 'C5', 'C3', 'C1', 'C2', 'C4', 'C6', 'T8', 'T10',
    'TP9', 'TP7', 'CP5', 'CP3', 'CP1', 'CP2', 'CP4', 'CP6', 'TP8', 'TP10',
    'P9', 'P7', 'P5', 'P3', 'P1', 'P2', 'P4', 'P6', 'P8', 'P10'
]
```

Figure 6.8- Lists of all the target and source columns.

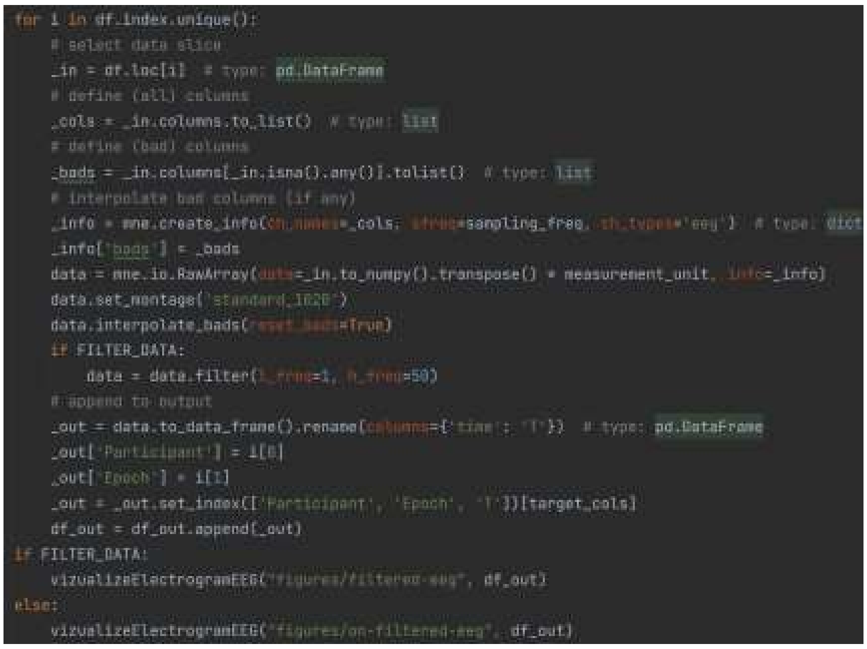
Then the data will be indexed by participant, column and epoch to update the indexing of the data frame before applying the pre-processing for bad column removal, independent component analysis and column interpolation. Also, create a new data frame with "target_cols" and the indexing is applied "participant, epoch and time" to store the pre-processed recordings (i.e., figure 6.9)



```
# index by participant, epoch and time
print('Updating Index')
df = df.set_index(['Participant', 'Epoch']).sort_index()
print('OK')

# DataFrame to store pre-processed recordings
df_out = pd.DataFrame(columns=target_cols, index=pd.MultiIndex.from_arrays(arrays=[[], [], []],
                                                                            names=['Participant', 'Epoch', 'T']))
```

Figure 6.9-Indexing data and creating a new data frame to hold the pre-processed data

For interpolating the missing columns, the author will take the unique slice from the data and use it to pre-process as shown in below Figure 6.10;



```
for i in df.index.unique():
    # select data slice
    _in = df.loc[i]  # type: pd.DataFrame
    # define (all) columns
    _cols = _in.columns.to_list()  # type: list
    # define (bad) columns
    _bads = _in.columns[_in.isna().any()].tolist()  # type: list
    # interpolate bad columns (if any)
    _info = mne.create_info(ch_names=_cols, sfreq=sampling_freq, ch_types='eeg')  # type: dict
    _info['bads'] = _bads
    data = mne.io.RawArray(data=_in.to_numpy().transpose() * measurement_unit, info=_info)
    data.set_montage('standard_1020')
    data.interpolate_bads(reset_bads=True)
    if FILTER_DATA:
        data = data.filter(l_freq=1, h_freq=50)
    # append to output
    _out = data.to_data_frame().rename(columns={'time': 'T'})  # type: pd.DataFrame
    _out['Participant'] = i[0]
    _out['Epoch'] = i[1]
    _out = _out.set_index(['Participant', 'Epoch', 'T'])[target_cols]
    df_out = df_out.append(_out)
if FILTER_DATA:
    vizualizeElectrogramEEG("figures/filtered-eeg", df_out)
else:
    vizualizeElectrogramEEG("figures/on-filtered-eeg", df_out)
```

Figure 6.10-Interpolating missing columns and removing bad columns

So, what happens in the above code snippet is the author will select the unique recording set and apply the data pre-processing techniques to it. The bad columns are identified as the NaN value columns in the selected data slice and then created information that can apply that particular slice to the "mne" package, so the channel names are the column names, the sampling frequency is the digital sampling rate of the dataset which is 250Hz and the channel types are passed as EEG because the "mne" package is capable of other time series data handling as well such as fMRI, EOG, etc. Then the bad column data is also added to the mne info. Then the data is added to the mne package using the mne.io.rawArray() function. Then the EEG montage is set to standard_1020 montage of mne, therefore the author created the missing columns for the EEG data.

Then if filtering is applied the EEG data will get filtered with a bandpass filter of 1Hz -50Hz; this will remove the unnecessary noise in the data as the above-mentioned steps. Then the pre-processed slice will append to the data frame that was created to hold the pre-processed data. In Figure 6.11 and Figure 6.12 can see the difference between the bandpass filter effects.

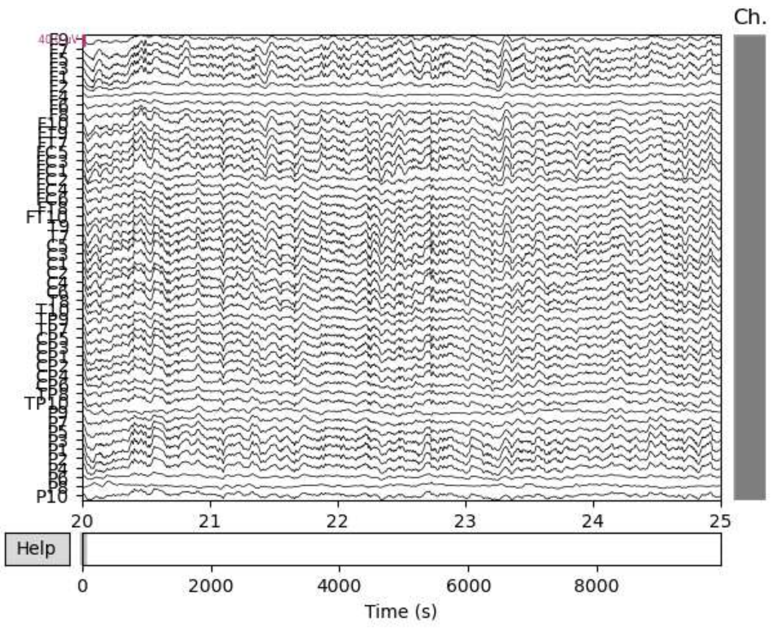


Figure 6.11-Unfiltered EEG data

While selecting the bad columns Independent Component Analysis (ICA) needs to be done. Figure 6.13 shows the code snippet for the ICA using the python mne package. After plotting each channel can identify which channels can be rejected and need interpolation.

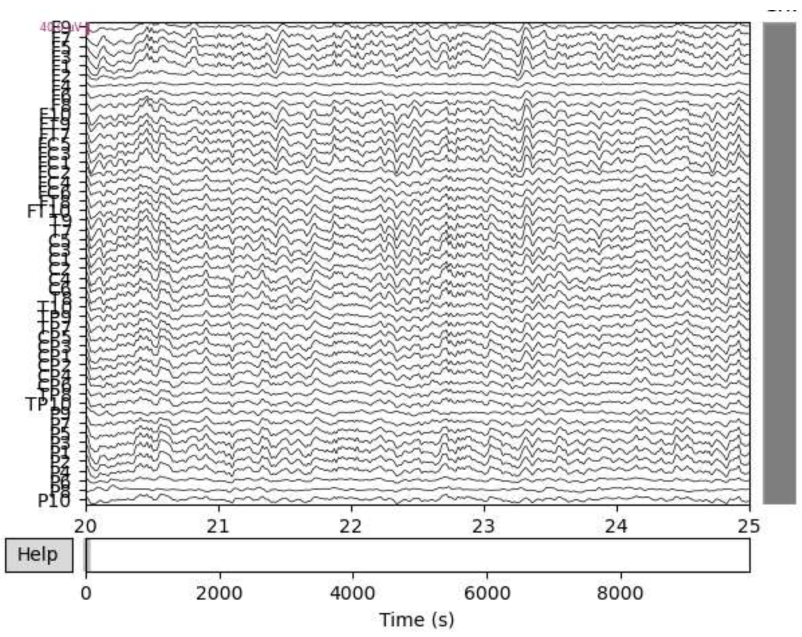
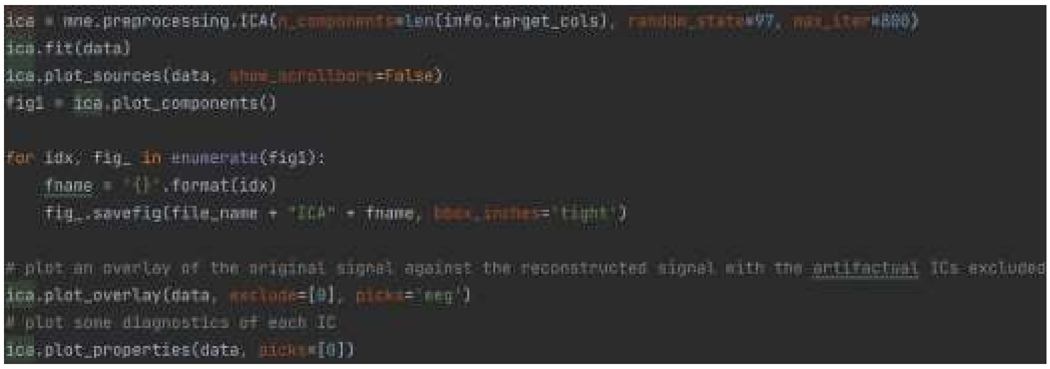


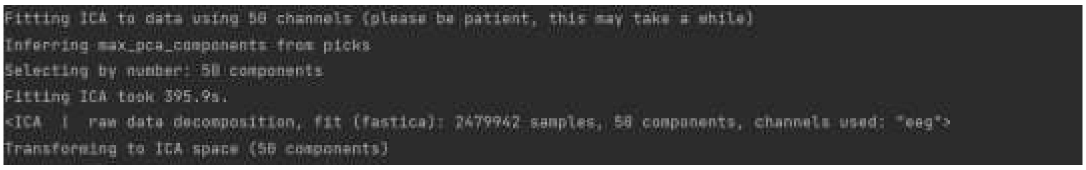
Figure 6.12-Filtered EEG data



```
ica = mne.preprocessing.ICA(n_components=len(info.target_cols), random_state=97, max_iter=800)
ica.fit(data)
ica.plot_sources(data, show_scrollbars=False)
fig1 = ica.plot_components()

for idx, fig_ in enumerate(fig1):
    fname = '{}'.format(idx)
    fig_.savefig(file_name + "ICA" + fname, bbox_inches='tight')

# plot an overlay of the original signal against the reconstructed signal with the artifactual ICs excluded
ica.plot_overlay(data, exclude=[0], picks='eeg')
# plot some diagnostics of each IC
ica.plot_properties(data, picks=[0])
```

Figure 6.13-Independent Component analysis

When doing ICA; can visualise and see which independent components (IC) need to be rejected or not. Normally IC can repair Electrooculography (EOG) and Electrocardiography (ECG) artefacts. In this project, the author will apply ICA to 50 channels as shown in figure 6.13 in there the n_components mean the number of ICs going to check.



```
Fitting ICA to data using 50 channels (please be patient, this may take a while)
Inferring max_pca_components from picks
Selecting by number: 50 components
Fitting ICA took 395.9s.
<ICA  |  raw data decomposition, fit (fastica): 2479942 samples, 50 components, channels used: "eeg">
Transforming to ICA space (50 components)
```

Figure 6.14-ICA time taken, no of components analysed

Applying ICA takes a few seconds. (i.e figure 6.14) Then we can generate the figure 6.15 graph and analyse the components independently. In the graph, pretty clearly see that the first component (ICA000) captures the EOG signal quite well, and the second component (ICA001) looks a lot like a heartbeat which is basically like, ECG So the filtering and interpolating column didn't affect anything in our dataset.

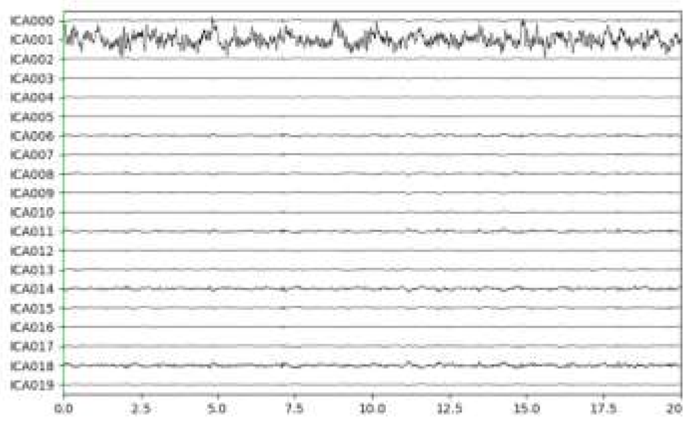


Figure 6.15-Time series of ICA

Additionally, by using plot components, the MNE can display the scalp field distribution of each component. Based on the values in the ICA mixing matrix, these are interpolated. There are more ways to observe our EOG and ECG artefacts even though it is very evident from the plots above that ICs are doing so. Using plot overlay can first plot an overlay of the original signal and the reconstructed signal without the artefactual ICs as in figure 6.19.

Also can check if there is anything wrong happen with the ICs while doing ICA using IC properties plots as in Appendix D: D.4.Then can check if the eye blinks (EOG) are removed by comparing the before and after pre-processing data in Appendix D: D.5

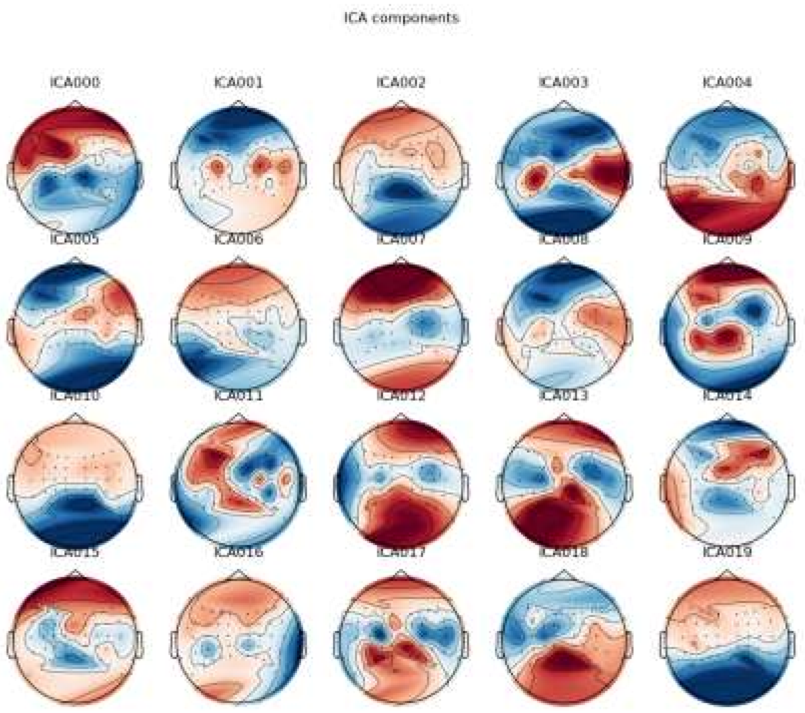
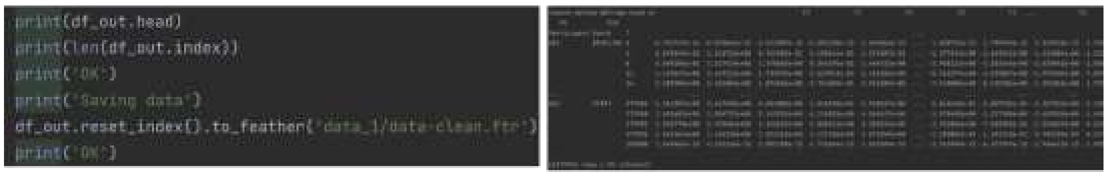


Figure 6.16-Scalp field distribution of each component

After doing the pre-processing author saves the pre-processed data to a feather file for further usage. (i.e., figure 6.17)



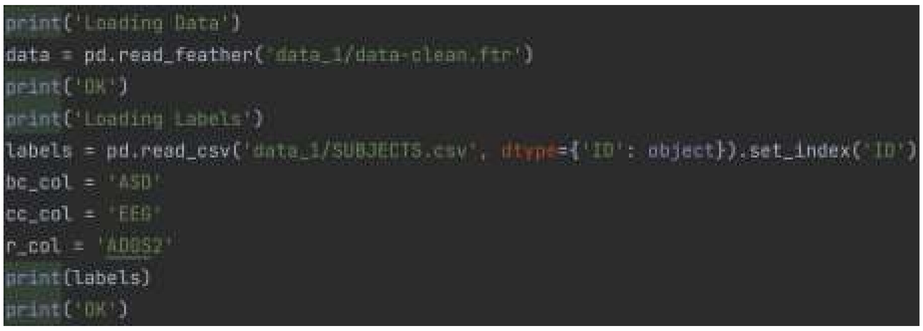
```
print(df_out.head)
print(len(df_out.index))
print('OK')
print('Saving data')
df_out.reset_index().to_feather('data_1/data-clean.ftr')
print('OK')
```

[illegible]

Figure 6.17-Preprocessed data save and pre-processed data containing data frame

6.4 Implementation of Frequency Bands Dataset

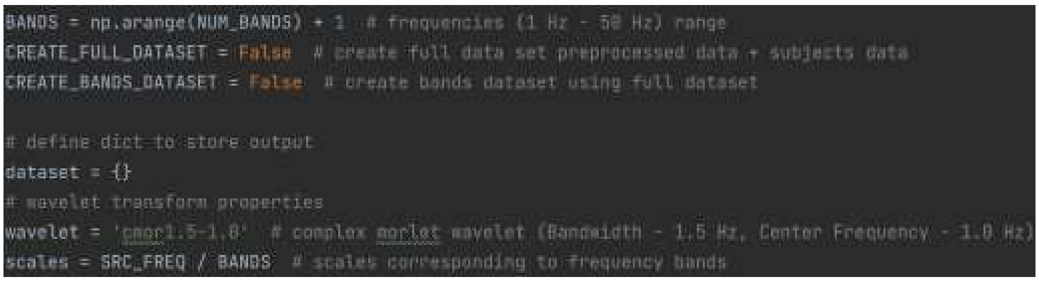
In this section, the author describes how he generated the frequency bands dataset using the pre-processed data. First, load the pre-processed data and the subject details CSV file which has the details of the subjects which have column 'ASD',' EEG' and 'ADOS2' values as shown in the figure 6.18 code snippet.



```
print('Loading Data')
data = pd.read_feather('data_1/data-clean.ftr')
print('OK')
print('Loading Labels')
labels = pd.read_csv('data_1/SUBJECTS.csv', dtype={'ID': object}).set_index('ID')
bc_col = 'ASD'
cc_col = 'EEG'
r_col = 'ADOS2'
print(labels)
print('OK')
```

Figure 6.18-load pre-processed EEG data and Subject data

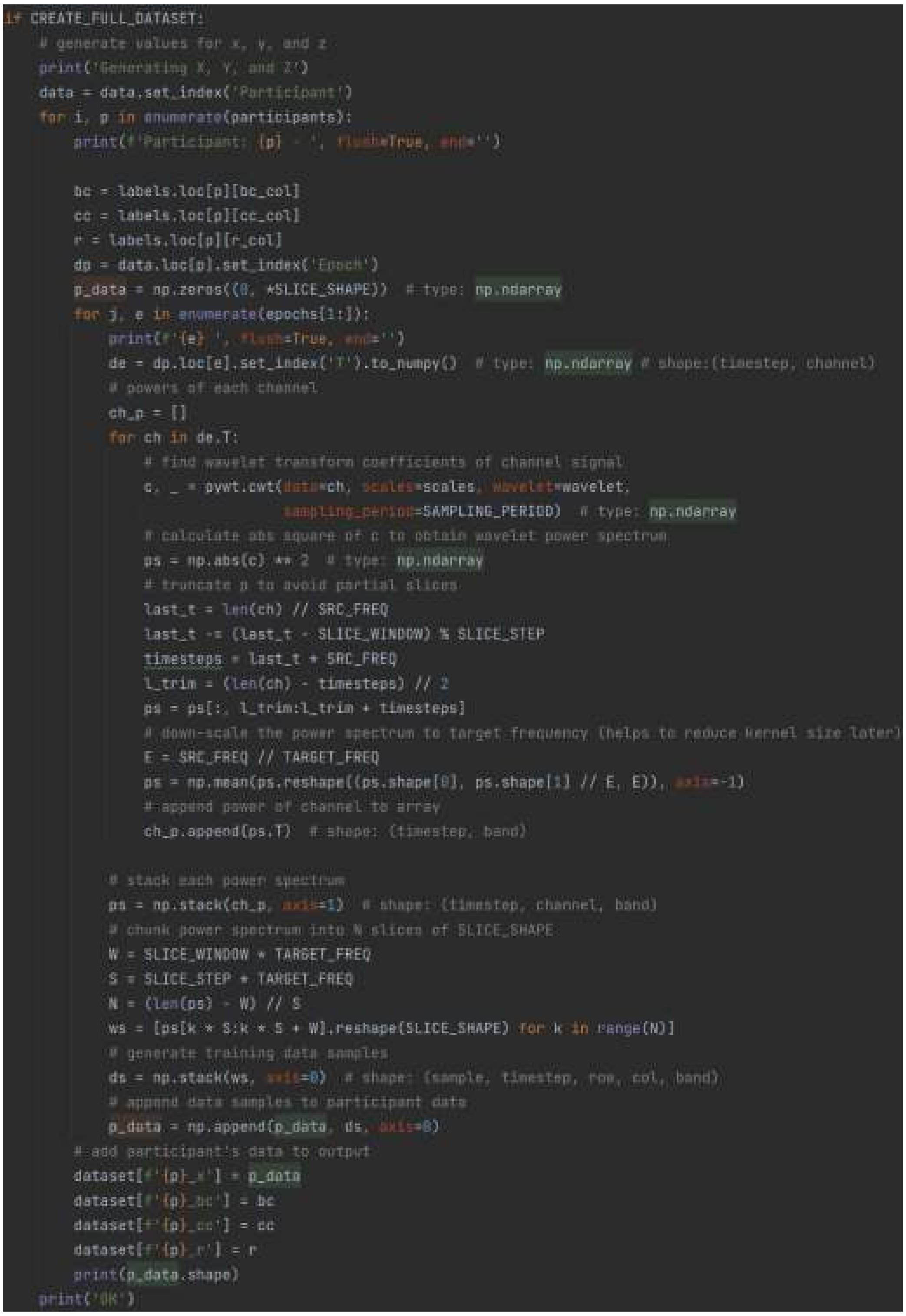
Next, the author defines some global variables that need to be while creating the frequency bands dataset. (i.e., figure 6.19)



```
BANDS = np.arange(NUM_BANDS) + 1  # frequencies (1 Hz - 50 Hz) range
CREATE_FULL_DATASET = False  # create full data set preprocessed data + subjects data
CREATE_BANDS_DATASET = False  # create bands dataset using full dataset

# define dict to store output
dataset = {}
# wavelet transform properties
wavelet = 'cmor1.5-1.0'  # complex morlet wavelet (Bandwidth - 1.5 Hz, Center Frequency - 1.0 Hz)
scales = SRC_FREQ / BANDS  # scales corresponding to frequency bands
```

Figure 6.19-Global variables

In the above variables, for this task, a Complex Morlet Wavelet with a centre frequency of 1Hz and a bandwidth of 1.5Hz was used (cmor1.5-1.0) which is in the "pywavelet" package. The author creates the full dataset which is a combination of pre-processed data and the data from the subjects.csv file. In the CREATE_FULL_DATASET section as shown in figure 6.20, the author will generate the values for x, y and z. X for EEG data, Y for ASD is true or not and Z is for ADOS2 score. So, to create the full dataset with pre-processed data and subjects' data; set the index of pre-processed data frame index to the participant column and loop it.



```python
if CREATE_FULL_DATASET:
    # generate values for x, y, and z
    print('Generating X, Y, and Z')
    data = data.set_index('Participant')
    for i, p in enumerate(participants):
        print(f'Participant: {p} - ', flush=True, end='')

        bc = labels.loc[p][bc_col]
        cc = labels.loc[p][cc_col]
        r = labels.loc[p][r_col]
        dp = data.loc[p].set_index('Epoch')
        p_data = np.zeros((0, *SLICE_SHAPE))  # type: np.ndarray
        for j, e in enumerate(epochs[1:]):
            print(f'{e} ', flush=True, end='')
            de = dp.loc[e].set_index('T').to_numpy()  # type: np.ndarray # shape:(timestep, channel)
            # powers of each channel
            ch_p = []
            for ch in de.T:
                # find wavelet transform coefficients of channel signal
                c, _ = pywt.cwt(data=ch, scales=scales, wavelet=wavelet,
                                sampling_period=SAMPLING_PERIOD)  # type: np.ndarray
                # calculate abs square of c to obtain wavelet power spectrum
                ps = np.abs(c) ** 2  # type: np.ndarray
                # truncate p to avoid partial slices
                last_t = len(ch) // SRC_FREQ
                last_t -= (last_t - SLICE_WINDOW) % SLICE_STEP
                timesteps = last_t * SRC_FREQ
                l_trim = (len(ch) - timesteps) // 2
                ps = ps[:, l_trim:l_trim + timesteps]
                # down-scale the power spectrum to target frequency (helps to reduce kernel size later)
                E = SRC_FREQ // TARGET_FREQ
                ps = np.mean(ps.reshape((ps.shape[0], ps.shape[1] // E, E)), axis=-1)
                # append power of channel to array
                ch_p.append(ps.T)  # shape: (timestep, band)

            # stack each power spectrum
            ps = np.stack(ch_p, axis=1)  # shape: (timestep, channel, band)
            # chunk power spectrum into N slices of SLICE_SHAPE
            W = SLICE_WINDOW * TARGET_FREQ
            S = SLICE_STEP * TARGET_FREQ
            N = (len(ps) - W) // S
            ws = [ps[k * S:k * S + W].reshape(SLICE_SHAPE) for k in range(N)]
            # generate training data samples
            ds = np.stack(ws, axis=0)  # shape: (sample, timestep, row, col, band)
            # append data samples to participant data
            p_data = np.append(p_data, ds, axis=0)
        # add participant's data to output
        dataset[f'{p}_x'] = p_data
        dataset[f'{p}_bc'] = bc
        dataset[f'{p}_cc'] = cc
        dataset[f'{p}_r'] = r
        print(p_data.shape)
    print('OK')
```

Figure 6.20-Create a full dataset

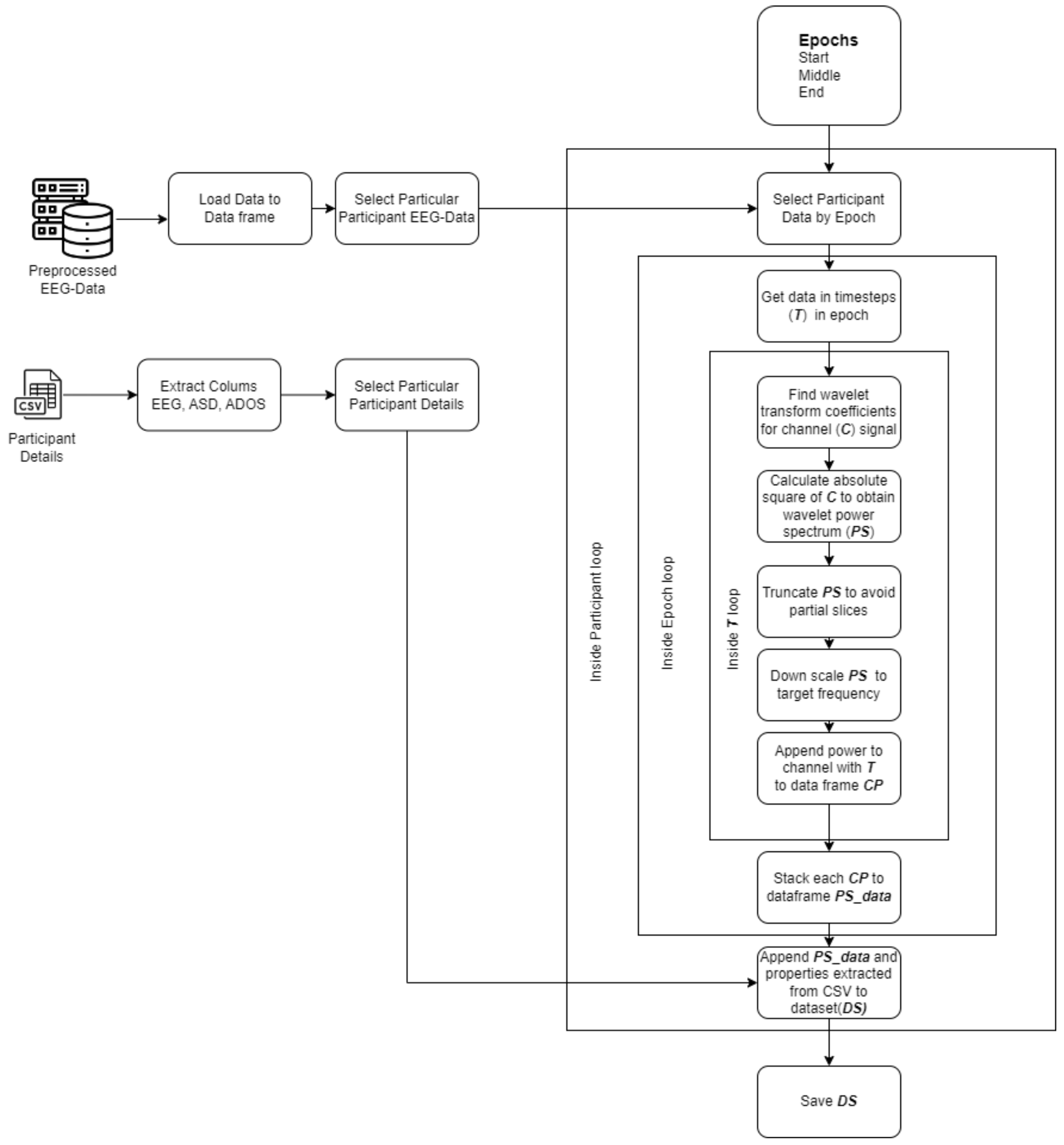
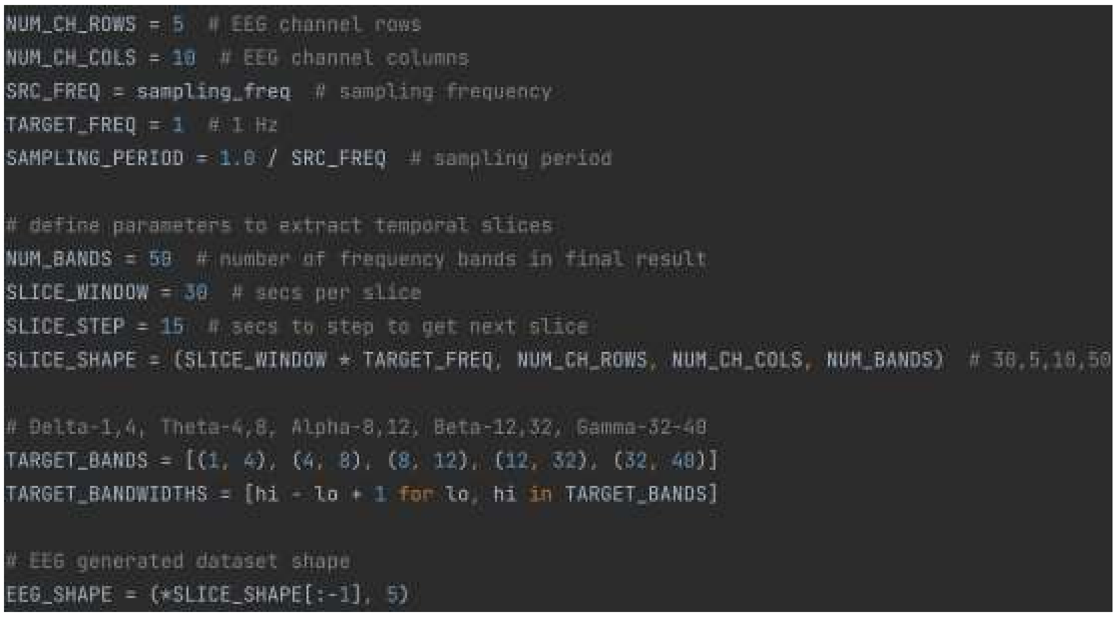

Epochs
Start
Middle
End
Preprocessed EEG-Data
Load Data to Data frame
Select Particular Participant EEG-Data
Select Participant Data by Epoch
Get data in timesteps (T) in epoch
Participant Details
Extract Colums EEG, ASD, ADOS
Select Particular Participant Details
Find wavelet transform coefficients for channel (C) signal
Calculate absolute square of C to obtain wavelet power spectrum (PS)
Truncate PS to avoid partial slices
Down scale PS to target frequency
Append power to channel with T to data frame CP
Inside Participant loop
Inside Epoch loop
Inside T loop
Stack each CP to dataframe PS_data
Append PS_data and properties extracted from CSV to dataset(DS)
Save DS

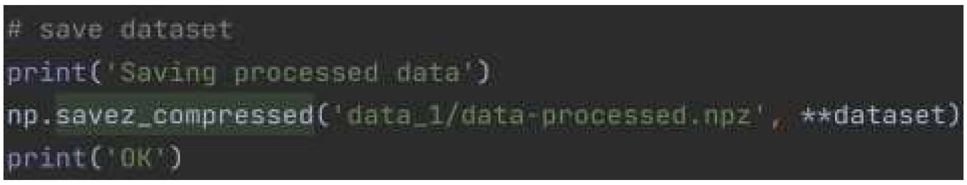
Figure 6.21-Create full dataset logic

The figure 6.20 code snippet logic is shown in figure 6.21. While looping the participant data in pre-processed EEG data set, get the respective participant data from the subject.csv data frame and assign them to variables Then loop the epochs without the BASELINE epoch, which are START, MIDDLE and END for a particular participant. Access the epoch rows and convert them to a NumPy array. After that loop the selected values according to the time. While looping the values find the wavelet transform coefficient of the channel signal using the pywavelets package. Then calculate the absolute square value of coefficients to obtain the wavelet power spectrum. Then downscale the power spectrum values to match the target frequency. Then append the power of the channel to an array. To do these use the global variables in figure 6.22. After looping all the time, stack each power spectrum and then chunk the power spectrum into slices. Then reshape it to sample, timestep, row, column and band which can use as training data. Finally, save the full data as NumPy compressed file as shown in the figure 6.23 code snippet.



```
NUM_CH_ROWS = 5  # EEG channel rows
NUM_CH_COLS = 10  # EEG channel columns
SRC_FREQ = sampling_freq  # sampling frequency
TARGET_FREQ = 1  # 1 Hz
SAMPLING_PERIOD = 1.0 / SRC_FREQ  # sampling period

# define parameters to extract temporal slices
NUM_BANDS = 50  # number of frequency bands in final result
SLICE_WINDOW = 30  # secs per slice
SLICE_STEP = 15  # secs to step to get next slice
SLICE_SHAPE = (SLICE_WINDOW * TARGET_FREQ, NUM_CH_ROWS, NUM_CH_COLS, NUM_BANDS)  # 30,5,10,50

# Delta-1,4, Theta-4,8, Alpha-8,12, Beta-12,32, Gamma-32-40
TARGET_BANDS = [(1, 4), (4, 8), (8, 12), (12, 32), (32, 40)]
TARGET_BANDWIDTHS = [hi - lo + 1 for lo, hi in TARGET_BANDS]

# EEG generated dataset shape
EEG_SHAPE = (*SLICE_SHAPE[:-1], 5)
```

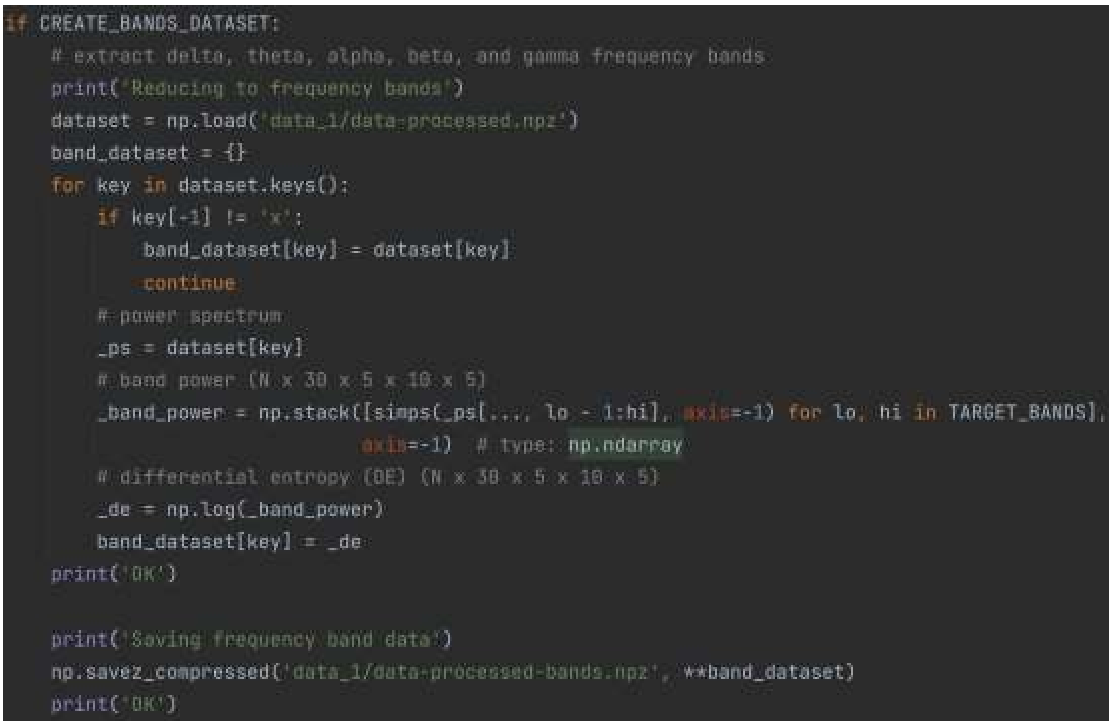
Figure 6.22-Constant variables used in generating a full dataset



```
# save dataset
print('Saving processed data')
np.savez_compressed('data_1/data-processed.npz', **dataset)
print('OK')
```

Figure 6.23-Save full data set

Then to create the frequency bands dataset that which author is going to use for creating neural networks the author loads the full data set pre-processed in the above step and loads it to a NumPy array. Then extract the delta, theta, alpha, beta, and gamma frequency bands. (i.e., Figure 6.24) When the author tries to check the power spectrum of the participants who are positive and negative ASD can see the clear differences between the power spectrums in Appendix D: D.6.

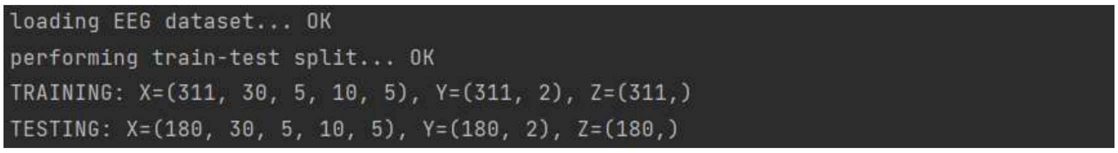


```python
if CREATE_BANDS_DATASET:
    # extract delta, theta, alpha, beta, and gamma frequency bands
    print('Reducing to frequency bands')
    dataset = np.load('data_1/data-processed.npz')
    band_dataset = {}
    for key in dataset.keys():
        if key[-1] != 'x':
            band_dataset[key] = dataset[key]
            continue
        # power spectrum
        _ps = dataset[key]
        # band power (N x 30 x 5 x 10 x 5)
        _band_power = np.stack([simps(_ps[..., lo - 1:hi], axis=-1) for lo, hi in TARGET_BANDS],
                               axis=-1)  # type: np.ndarray
        # differential entropy (DE) (N x 30 x 5 x 10 x 5)
        _de = np.log(_band_power)
        band_dataset[key] = _de
    print('OK')

    print('Saving frequency band data')
    np.savez_compressed('data_1/data-processed-bands.npz', **band_dataset)
    print('OK')
```

Figure 6.24-Create frequency bands dataset

6.5 Implementation of Neural Network Models

As discussed in the literature review, the author is using several neural network techniques to create several models and tries to find the optimum model and the techniques. The data set will be split to train and test using random fractions and create train and test using the X, Y and Z values created while creating the frequency bands dataset. Training data is 70% and Testing data is 30% of the dataset. The coding snippet for the train test data split is in Appendix D: D.8 After splitting the dataset it will split like figure 6.25.

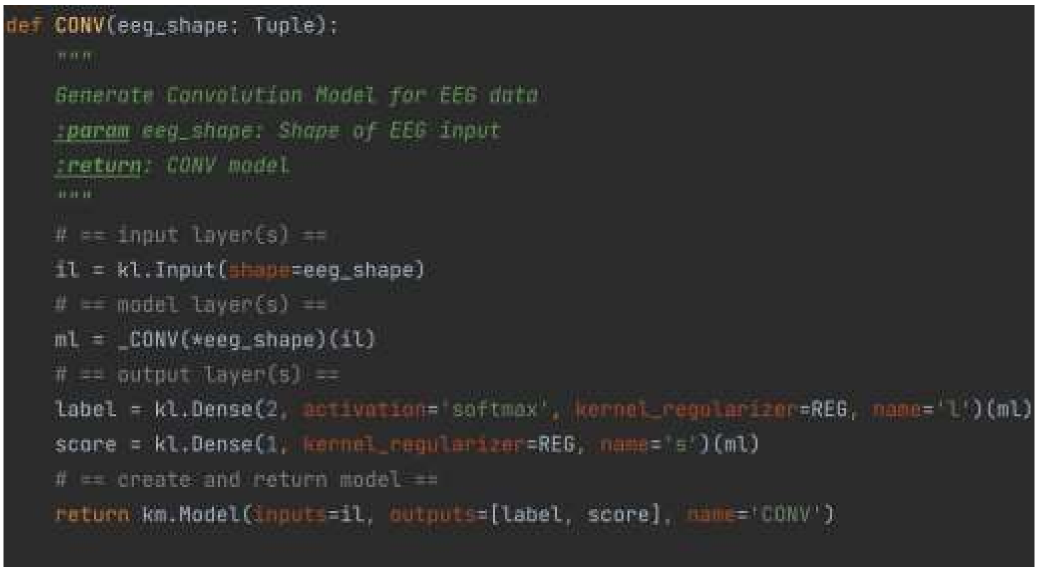


```
loading EEG dataset... OK
performing train-test split... OK
TRAINING: X=(311, 30, 5, 10, 5), Y=(311, 2), Z=(311,)
TESTING: X=(180, 30, 5, 10, 5), Y=(180, 2), Z=(180,)
```

Figure 6.25-Train-Test data split

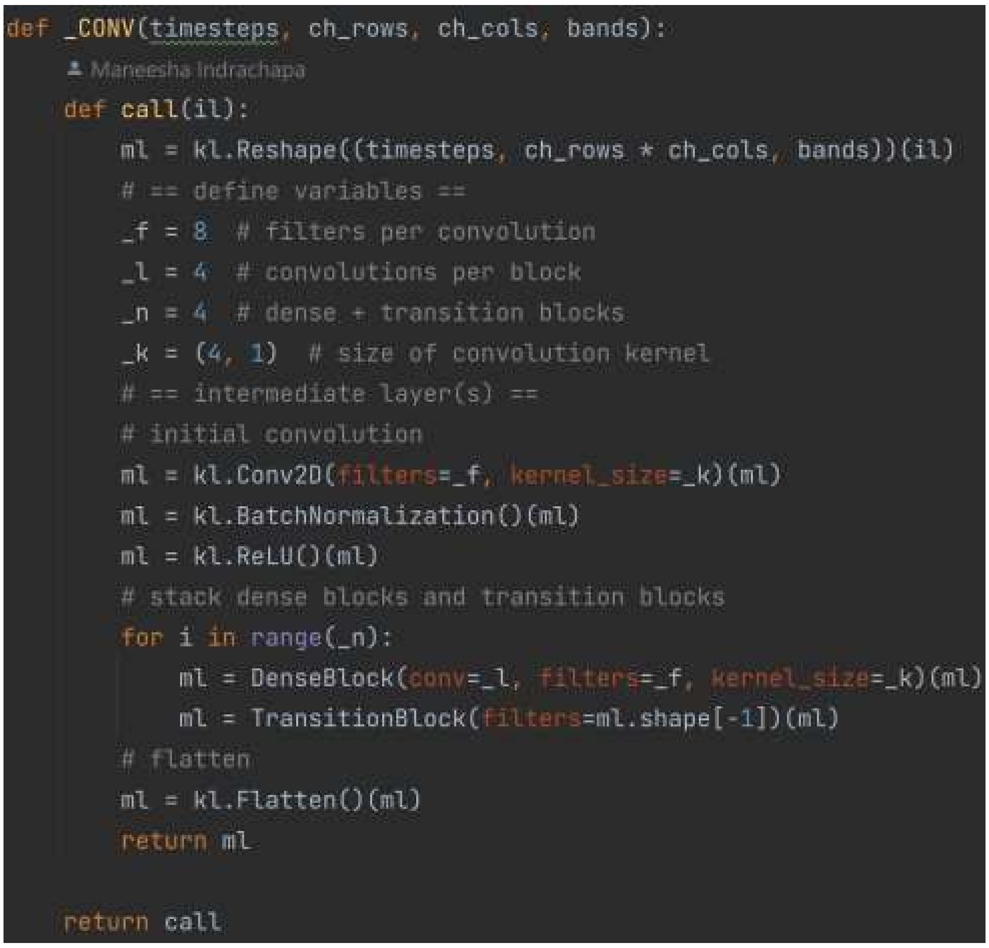
Next, the author creates global variables for losses and model-specific configurations which need to pass for the model generations. The coding snippet for model-specific global configurations can be found in Appendix D: D.9

6.5.1 Implementation of Convolutional Model



```
def CONV(eeg_shape: Tuple):
    """
    Generate Convolution Model for EEG data
    :param eeg_shape: Shape of EEG input
    :return: CONV model
    """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _CONV(*eeg_shape)(il)
    # == output layer(s) ==
    label = kl.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(ml)
    score = kl.Dense(1, kernel_regularizer=REG, name='s')(ml)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='CONV')
```

Figure 6.26-Convolutional model function with input and output layers

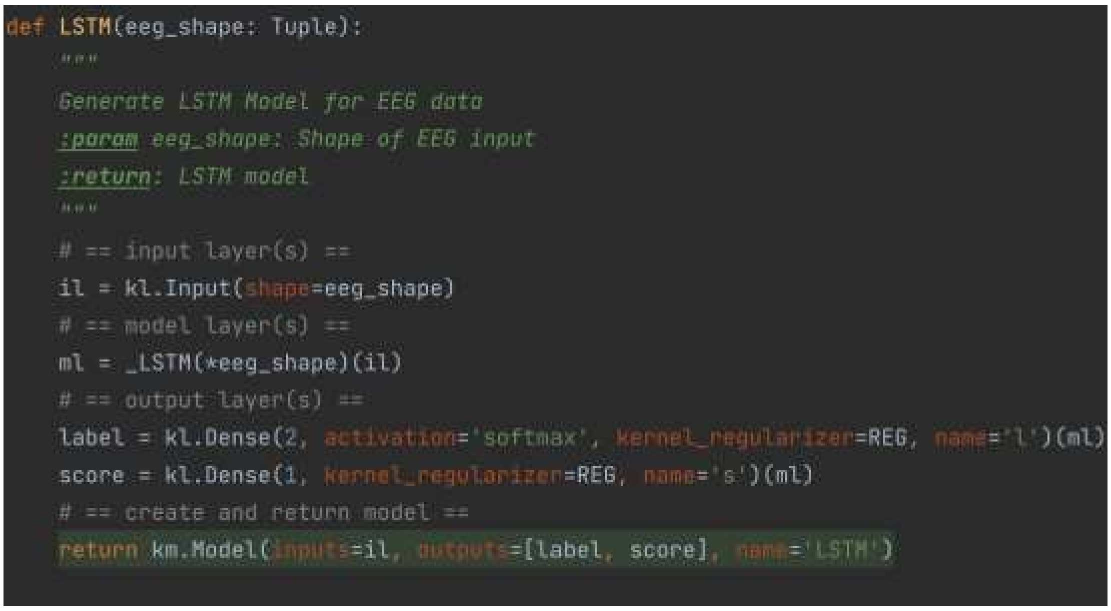


```
def _CONV(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa
    def call(il):
        ml = kl.Reshape((timesteps, ch_rows * ch_cols, bands))(il)
        # == define variables ==
        _f = 8  # filters per convolution
        _l = 4  # convolutions per block
        _n = 4  # dense + transition blocks
        _k = (4, 1)  # size of convolution kernel
        # == intermediate layer(s) ==
        # initial convolution
        ml = kl.Conv2D(filters=_f, kernel_size=_k)(ml)
        ml = kl.BatchNormalization()(ml)
        ml = kl.ReLU()(ml)
        # stack dense blocks and transition blocks
        for i in range(_n):
            ml = DenseBlock(conv=_l, filters=_f, kernel_size=_k)(ml)
            ml = TransitionBlock(filters=ml.shape[-1])(ml)
        # flatten
        ml = kl.Flatten()(ml)
        return ml

    return call
```

Figure 6.27-Convolutional model function with hidden layers

As shown in the figure, the CNN model was implemented using the TensorFlow library and the input layer passed the EEG_Shape and then passed to the _CONV function which contains the hidden layers. (i.e., Figure 6.26) As shown in figure 6.27 inside the hidden layers, firstly author reshaped the layer which need to pass and defined the variables that need to pass to the Conv2D layer. After that, a Batch Normalization was done. Then ReLu activation function was used to finish the initial convolution. After that Dense and transition blocks were used to optimize the layers and finally flattened layer was used. The model summary can be seen in Appendix D: D.10.

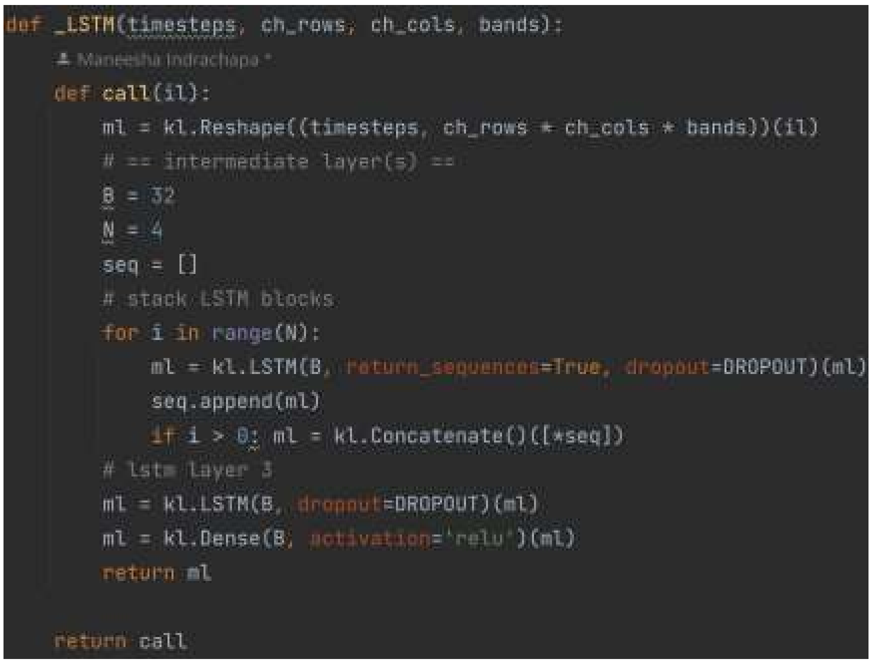
6.5.2 Implementation of the LSTM Model



```
def LSTM(eeg_shape: Tuple):
    """
    Generate LSTM Model for EEG data
    :param eeg_shape: Shape of EEG input
    :return: LSTM model
    """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _LSTM(*eeg_shape)(il)
    # == output layer(s) ==
    label = kl.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(ml)
    score = kl.Dense(1, kernel_regularizer=REG, name='s')(ml)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='LSTM')
```

Figure 6.28-LSTM model function with input and output layers

As shown in the figure, for the implementation of the LSTM model the author used an input layer which contains EEG_shape which is predefined in the global variables and then passed to the _LSTM function which contains the hidden layers. (i.e., Figure 6.28) In the hidden layers, the author reshaped the shape using reshape layer and defined the variables that need to pass to the LSTM layer. After passing to the LSTM layer the author did the concatenation of those layers. Then sent it again through an LSTM layer and finally through a Dense layer. (i.e., Figure 6.29) The model summary can be seen in Appendix D: D.11.

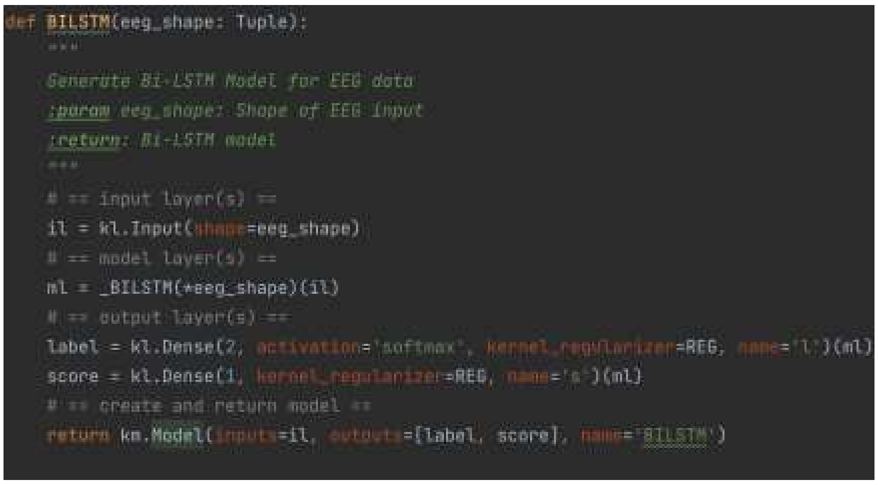


```python
def _LSTM(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa *
    def call(il):
        ml = kl.Reshape((timesteps, ch_rows * ch_cols * bands))(il)
        # == intermediate layer(s) ==
        B = 32
        N = 4
        seq = []
        # stack LSTM blocks
        for i in range(N):
            ml = kl.LSTM(B, return_sequences=True, dropout=DROPOUT)(ml)
            seq.append(ml)
            if i > 0: ml = kl.Concatenate()([*seq])
        # lstm layer 3
        ml = kl.LSTM(B, dropout=DROPOUT)(ml)
        ml = kl.Dense(B, activation='relu')(ml)
        return ml

    return call
```

Figure 6.29-LSTM model function with hidden layers

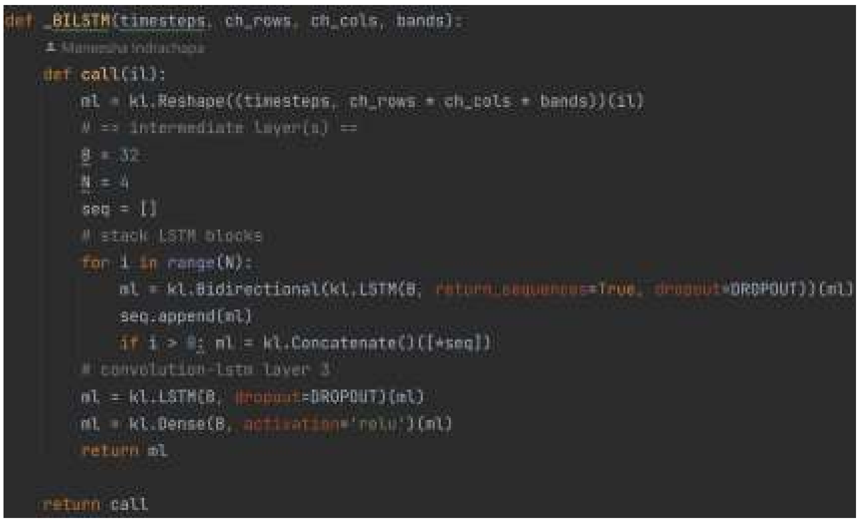
6.5.3 Implementation of Bi-LSTM Model

As shown in the figure, the Bi- LSTM model was implemented using the TensorFlow library and for the input, the layer passed the EEG_shape and then passed to the _BILSTM function which contains the hidden layers. (i.e., Figure 6.30) In the hidden layers, firstly author reshaped the shape that is passed and defined the variables that need to pass to the LSTM layer inside the Bidirectional layer. The author used Bidirectional and LSTM blocks and after doing the LSTM layer function did the concatenation of those layers after that send it again through an LSTM layer and finally through a Dense layer. (i.e., Figure 6.31) The model summary can be seen in Appendix D: D.12.



```python
def BILSTM(eeg_shape: Tuple):
    """
    Generate Bi-LSTM Model for EEG data
    :param eeg_shape: Shape of EEG input
    :return: Bi-LSTM model
    """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _BILSTM(*eeg_shape)(il)
    # == output layer(s) ==
    label = kl.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(ml)
    score = kl.Dense(1, kernel_regularizer=REG, name='s')(ml)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='BILSTM')
```

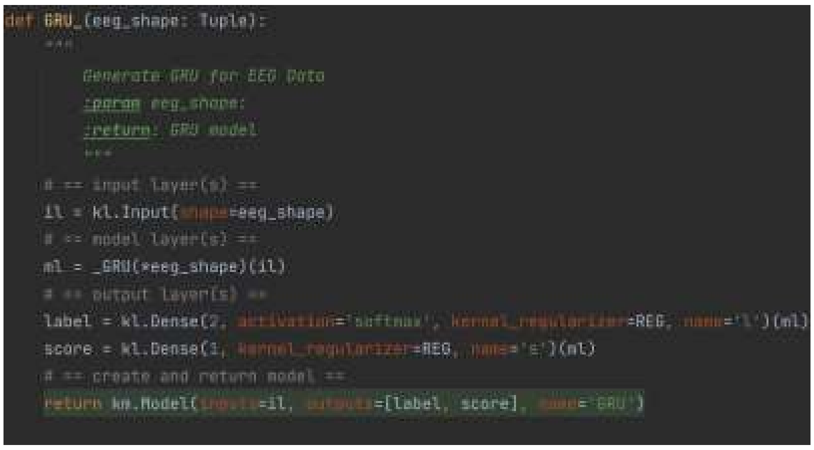
Figure 6.30-Bi-LSTM model function with input and output layers



```
def _BILSTM(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa
    def call(il):
        ml = kl.Reshape((timesteps, ch_rows * ch_cols * bands))(il)
        # == intermediate layer(s) ==
        B = 32
        N = 4
        seq = []
        # stack LSTM blocks
        for i in range(N):
            ml = kl.Bidirectional(kl.LSTM(B, return_sequences=True, dropout=DROPOUT))(ml)
            seq.append(ml)
            if i > 0: ml = kl.Concatenate()([*seq])
        # convolution-lstm layer 3
        ml = kl.LSTM(B, dropout=DROPOUT)(ml)
        ml = kl.Dense(B, activation='relu')(ml)
        return ml

    return call
```

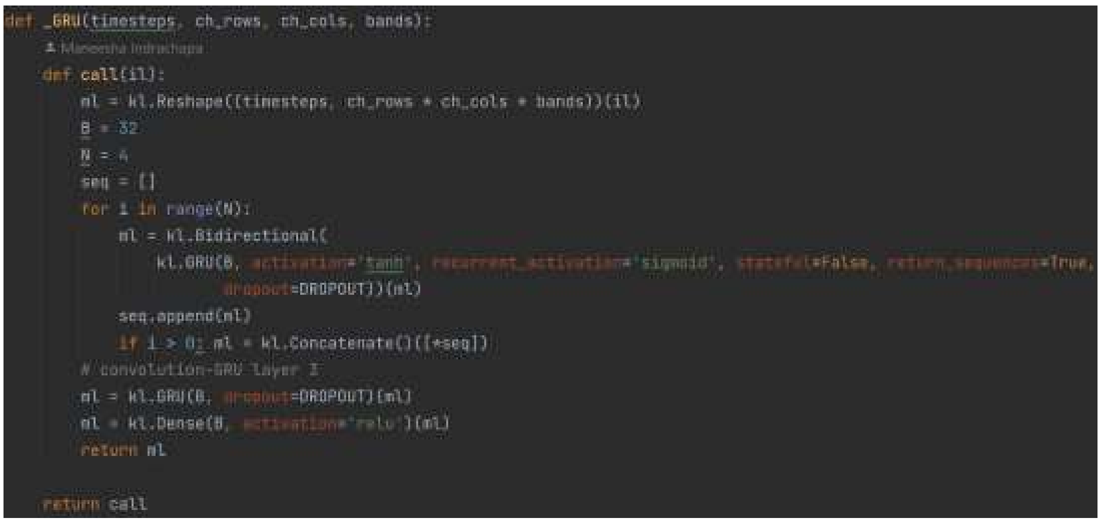
Figure 6.31-Bi-LSTM model function with hidden layers

6.5.4 Implementation of GRU Model



```
def GRU_(eeg_shape: Tuple):
    """
        Generate GRU for EEG Data
        :param eeg_shape:
        :return: GRU model
        """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _GRU(*eeg_shape)(il)
    # == output layer(s) ==
    label = kl.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(ml)
    score = kl.Dense(1, kernel_regularizer=REG, name='s')(ml)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='GRU')
```

Figure 6.32-GRU model function with input and output layers

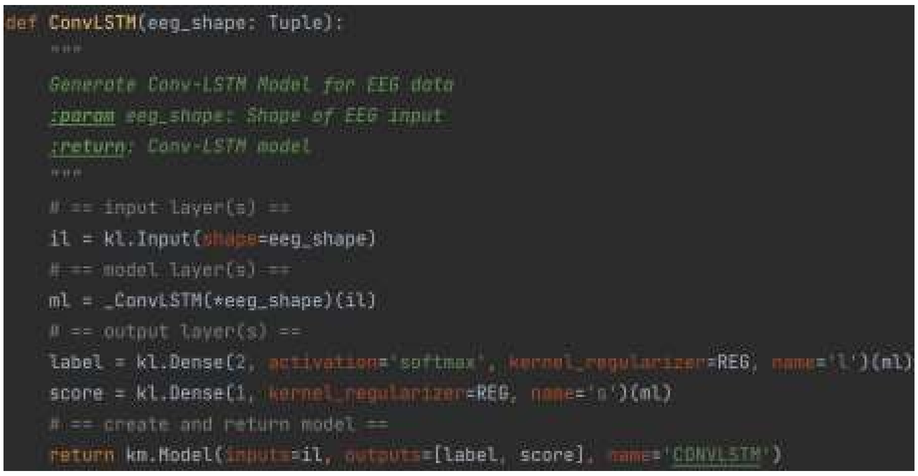


```
def _GRU(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa
    def call(il):
        ml = kl.Reshape((timesteps, ch_rows * ch_cols * bands))(il)
        B = 32
        N = 4
        seq = []
        for i in range(N):
            ml = kl.Bidirectional(
                kl.GRU(B, activation='tanh', recurrent_activation='sigmoid', stateful=False, return_sequences=True,
                       dropout=DROPOUT))(ml)
            seq.append(ml)
            if i > 0: ml = kl.Concatenate()([*seq])
        # convolution-GRU layer I
        ml = kl.GRU(B, dropout=DROPOUT)(ml)
        ml = kl.Dense(B, activation='relu')(ml)
        return ml

    return call
```

Figure 6.33-GRU model function with hidden layers

As shown in the figure, the GRU model was implemented. The input layer passed the EEG_shape and then passed to the _GRU function which contains the hidden layers. (i.e., Figure 6.32) In the hidden layers, firstly author passed through a reshape-layer. Then defined the variables and passed through a GRU layer inside the Bidirectional layer. The author concatenates the used Bidirectional and GRU blocks after that and again sent through a GRU layer and finally through a Dense layer. (i.e., Figure 6.33) The model summary can be seen in Appendix D: D.13.

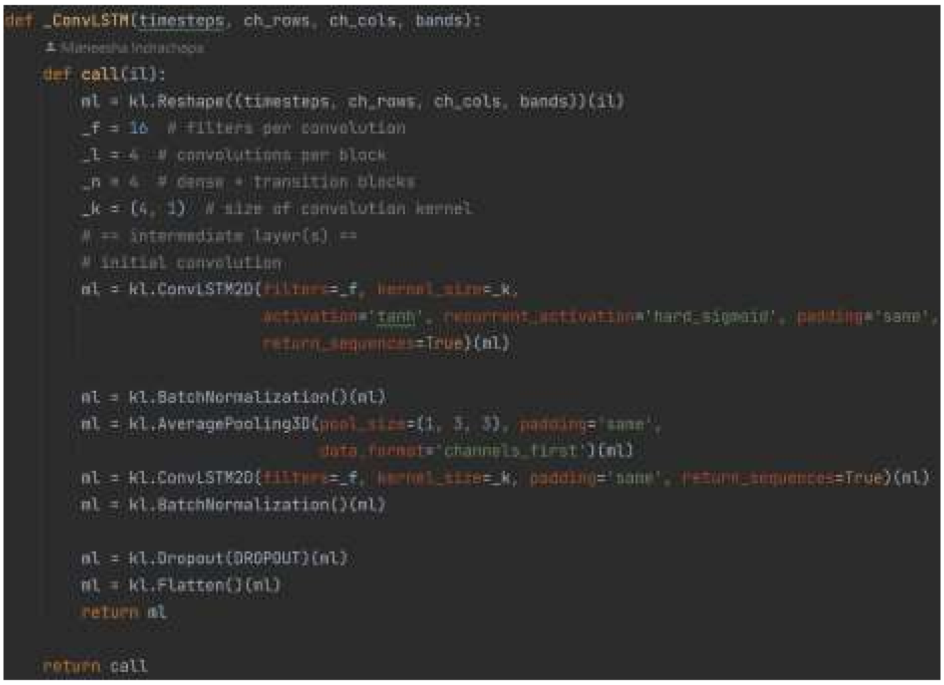
6.5.5 Implementation of ConvLSTM Model



```
def ConvLSTM(eeg_shape: Tuple):
    """
    Generate Conv-LSTM Model for EEG data
    :param eeg_shape: Shape of EEG input
    :return: Conv-LSTM model
    """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _ConvLSTM(*eeg_shape)(il)
    # == output layer(s) ==
    label = kl.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(ml)
    score = kl.Dense(1, kernel_regularizer=REG, name='s')(ml)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='CONVLSTM')
```

Figure 6.34-ConvLSTM model function with input and output layers

As shown in the figure, the ConvLSTM model was implemented using the TensorFlow Keras library. The input layer input shape was the EEG_shape shown in Figure 6.34 and then passed to the _CONVLSTM function which contains the hidden layers. In the hidden layers, firstly author reshaped the shape that is passed and defined the variables that need to pass to the ConvLSTM2D layer. After that, a Batch Normalization was done. Then it gets sent through the AveragePooling2D layer. Then again sent through the ConvLSTM2D layer and Batch normalization was done. Finally, use the 0.2 dropout layer to convert 20% of weights to zero to stop overfitting the dataset and used a flattened layer. (i.e., Figure 6.35) The model summary can be seen in Appendix D: D.14.



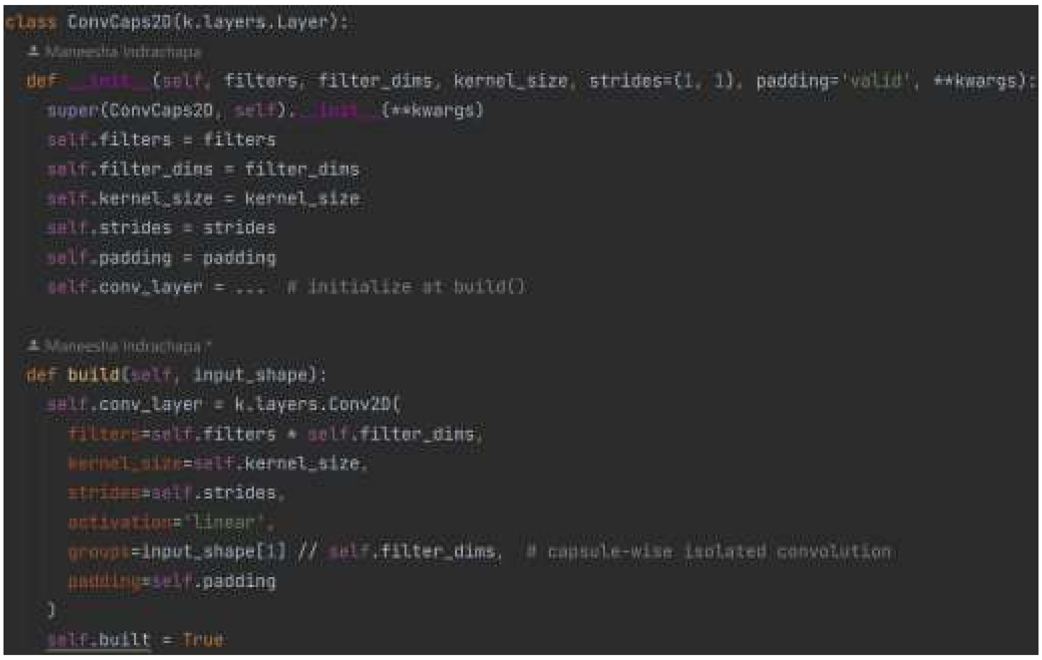
```
def _ConvLSTM(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa
    def call(il):
        ml = kl.Reshape((timesteps, ch_rows, ch_cols, bands))(il)
        _f = 16  # filters per convolution
        _l = 4  # convolutions per block
        _n = 4  # dense + transition blocks
        _k = (4, 1)  # size of convolution kernel
        # == intermediate layer(s) ==
        # initial convolution
        ml = kl.ConvLSTM2D(filters=_f, kernel_size=_k,
                           activation='tanh', recurrent_activation='hard_sigmoid', padding='same',
                           return_sequences=True)(ml)

        ml = kl.BatchNormalization()(ml)
        ml = kl.AveragePooling3D(pool_size=(1, 3, 3), padding='same',
                                 data_format='channels_first')(ml)
        ml = kl.ConvLSTM2D(filters=_f, kernel_size=_k, padding='same', return_sequences=True)(ml)
        ml = kl.BatchNormalization()(ml)

        ml = kl.Dropout(DROPOUT)(ml)
        ml = kl.Flatten()(ml)
        return ml

    return call
```

Figure 6.35-ConvLSTM model function with hidden layers

6.5.6 Implementation of CapsNet Model

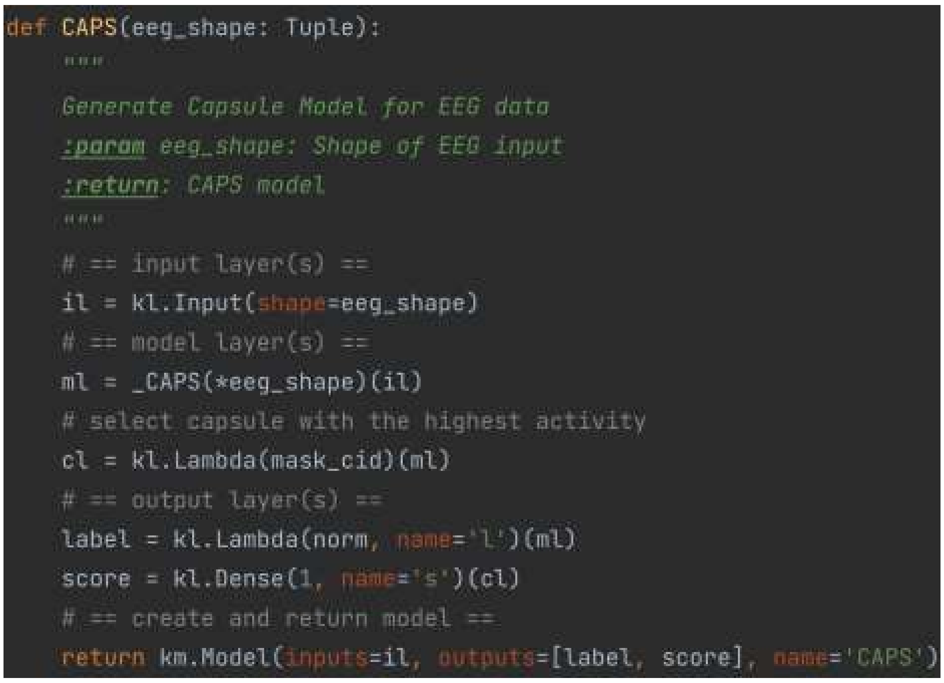
The CapsNet was implemented by the author to create the CapsNet model following the original paper presented for the capsule network implementation by Sabour, Frosst and Hinton 2017. (i.e., Figure 6.36)

As shown in the figure, the CapsNet model was implemented using the TensorFlow Keras library and the input layer shape was EEG_shape which is EEG_SHAPE = (*SLICE_SHAPE[:-1], 5) shown in Figure 6.37 and then passed to the _CAPS function which contains the hidden layers. In the hidden layers, firstly author reshaped the shape that is passed and defined the variables that need to pass to the Conv2D layer. After that, a Batch Normalization was done. Then it gets sent through the ReLu activation function and several dense, transition blocks. Then it will convert to the capsule domain using ConvCaps2D layer and squash which computes the probability of vector 0-1. (i.e., Figure 6.38) The model summary can be seen in Appendix D.15.



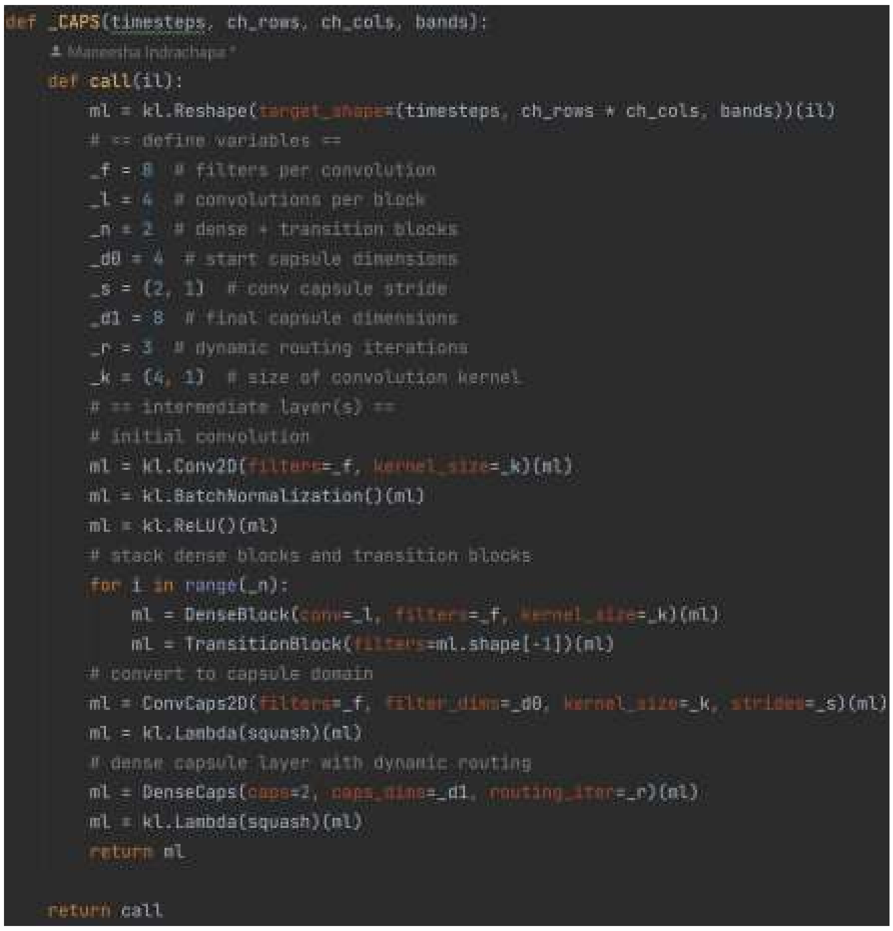
```python
class ConvCaps2D(k.layers.Layer):
  Maneesha Indrachapa
  def __init__(self, filters, filter_dims, kernel_size, strides=(1, 1), padding='valid', **kwargs):
    super(ConvCaps2D, self).__init__(**kwargs)
    self.filters = filters
    self.filter_dims = filter_dims
    self.kernel_size = kernel_size
    self.strides = strides
    self.padding = padding
    self.conv_layer = ...  # initialize at build()

  Maneesha Indrachapa *
  def build(self, input_shape):
    self.conv_layer = k.layers.Conv2D(
      filters=self.filters * self.filter_dims,
      kernel_size=self.kernel_size,
      strides=self.strides,
      activation='linear',
      groups=input_shape[1] // self.filter_dims,  # capsule-wise isolated convolution
      padding=self.padding
    )
    self.built = True
```

Figure 6.36-ConvCaps2D implementation



```python
def CAPS(eeg_shape: Tuple):
    """
    Generate Capsule Model for EEG data
    :param eeg_shape: Shape of EEG input
    :return: CAPS model
    """
    # == input layer(s) ==
    il = kl.Input(shape=eeg_shape)
    # == model layer(s) ==
    ml = _CAPS(*eeg_shape)(il)
    # select capsule with the highest activity
    cl = kl.Lambda(mask_cid)(ml)
    # == output layer(s) ==
    label = kl.Lambda(norm, name='l')(ml)
    score = kl.Dense(1, name='s')(cl)
    # == create and return model ==
    return km.Model(inputs=il, outputs=[label, score], name='CAPS')
```

Figure 6.37-CapsNet model function with input and output layers



```python
def _CAPS(timesteps, ch_rows, ch_cols, bands):
    Maneesha Indrachapa *
    def call(il):
        ml = kl.Reshape(target_shape=(timesteps, ch_rows * ch_cols, bands))(il)
        # == define variables ==
        _f = 8  # filters per convolution
        _l = 4  # convolutions per block
        _n = 2  # dense + transition blocks
        _d0 = 4  # start capsule dimensions
        _s = (2, 1)  # conv capsule stride
        _d1 = 8  # final capsule dimensions
        _r = 3  # dynamic routing iterations
        _k = (4, 1)  # size of convolution kernel
        # == intermediate layer(s) ==
        # initial convolution
        ml = kl.Conv2D(filters=_f, kernel_size=_k)(ml)
        ml = kl.BatchNormalization()(ml)
        ml = kl.ReLU()(ml)
        # stack dense blocks and transition blocks
        for i in range(_n):
            ml = DenseBlock(conv=_l, filters=_f, kernel_size=_k)(ml)
            ml = TransitionBlock(filters=ml.shape[-1])(ml)
        # convert to capsule domain
        ml = ConvCaps2D(filters=_f, filter_dims=_d0, kernel_size=_k, strides=_s)(ml)
        ml = kl.Lambda(squash)(ml)
        # dense capsule layer with dynamic routing
        ml = DenseCaps(caps=2, caps_dims=_d1, routing_iter=_r)(ml)
        ml = kl.Lambda(squash)(ml)
        return ml

    return call
```

Figure 6.38-CapsNet model function with hidden layers

6.6 Implementation of Web application

The author used the below figure 6.39 architecture to create the prototype. The author developed a front-end application using Angular and a backend using the Flask python framework. To run the prediction function flask will run the python script created for the prediction and that script will select the pre-trained model from the storage.

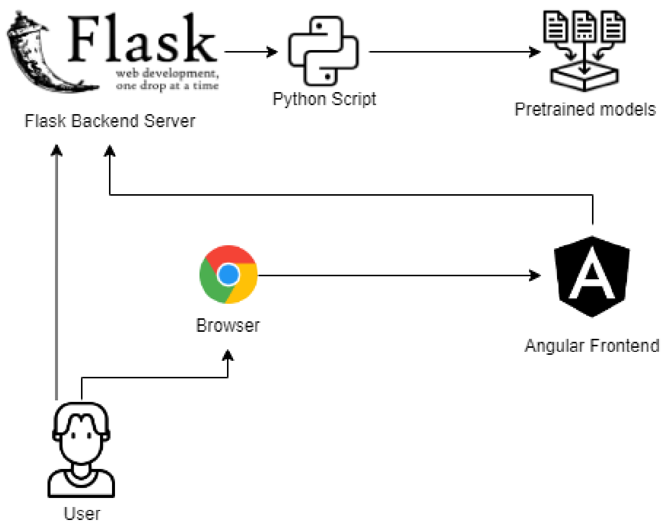
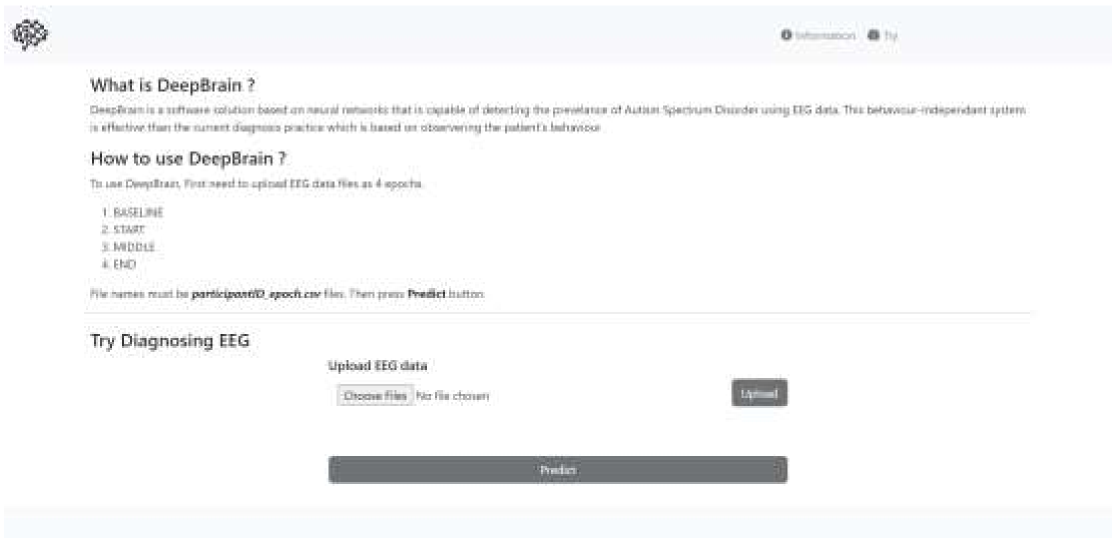


Figure 6.39-Web application architecture



What is DeepBrain ?

DeepBrain is a software solution based on neural networks that is capable of detecting the prevelance of Autism Spectrum Disorder using EEG data. This behaviour-independant system is effective than the current diagnosis practice which is based on observering the patient's behaviour.

How to use DeepBrain ?

To use DeepBrain, First need to upload EEG data files as 4 epochs.

1. BASELINE
2. START
3. MIDDLE
4. END

File names must be ***participantID_epoch.csv*** files. Then press **Predict** button.

Try Diagnosing EEG

Upload EEG data

Figure 6.40-Web application home page

The Web application home page consists of summarized information about the project and how to use it. (i.e., Figure 6.40) Users can upload the EEG data using the upload function (Appendix D: D.17) and it is developed to upload multiple files. After that user can click the “predict” button which triggers the prediction endpoint (Appendix D: D.16) and can see the predicted output in several seconds as shown in figure 6.41 below.

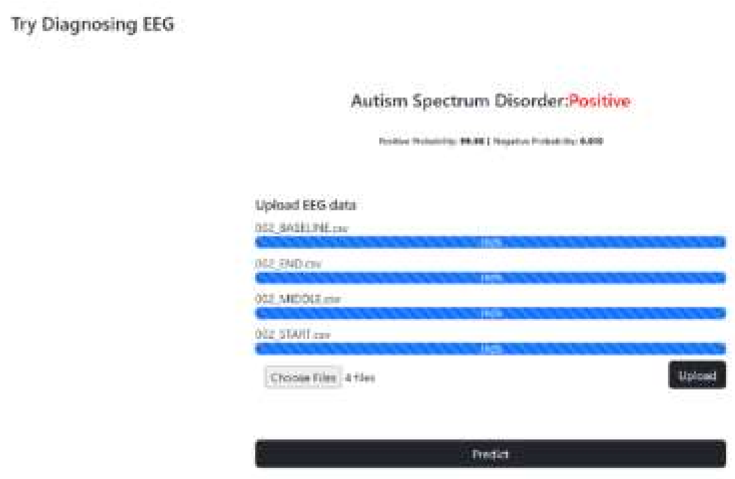


Figure 6.41-Web application prediction

6.8 Chapter Summary

The implementation of the system's primary functionalities was covered in this chapter. The author describes the reasonings behind his choice of Python programming language, libraries selection and data selection under the technology selection headline. Then the author describes dataset pre-processing pipeline he implemented which included Noise filtering, column interpolation and independent component analysis. Then describes how the pre-processed data was used in creating a frequency bands data set step by step. Then the user describes the 6 neural network models that were implemented along with code snippets and model summaries. Finally, the author describes web application endpoints and user interface development.

Chapter 07 - Testing

7.1 Chapter Overview

This will focus on the prototype implementation's testing phase. Testing goals, technique options, and other subjects will all be covered in this chapter. Then, this will be evaluated to ensure effective implementation of the functional requirements and ensure that the output expected is satisfactory.

7.2 Functional Requirement Testing

Functional requirement testing is the process of evaluating an application to see if it satisfies the functional demands placed on the system. The author has specified two required and two desired functional needs, following the Functional Requirement Table. The two necessary functional prerequisites are as follows.

1. The application should be able to predict ASD-positive or ASD-negative when an EEG data set is given by the user
2. The application should only use EEG data provided by the user and generate the necessary frequency bands dataset for predictions.

Different test types can be used to do functional testing. Unit testing, integration testing, and system testing are some of the frequently used functional test types.

7.2.1 Unit Testing

Unit testing is the process of determining whether a single module or component is operating as intended. Additionally, unit tests can be carried out while the application is still being developed.

7.2.1.1 Testing Scenarios

The following test cases will primarily be tested via unit tests.

- Using the assigned algorithm, the model is constructed appropriately.
- The model predicts values
- The system receives EEG data system will generate the necessary frequency bands data set to make predictions.
- Python script integrations are not necessary for the operation of web components.

7.2.1.2 Testing Method

For the unit testing work, Whitebox and Blackbox Testing methodologies will be used. As a result, Blackbox testing will be performed by entering test values into the program. Line-by-line code values will be tested utilizing debug methods during white-box testing.

7.2.1.3 Testing Tools

For this project unit testing, two primary tools were employed which are IntelliJ PyCharm IDE and VsCode. PyCharm will be used primarily for algorithm-based testing, and its debug mode was the primary testing tool for debugging Python-based algorithms. For testing web-based components, VsCode was employed.

The unit tests performed for the created system to validate the functional requirements are listed below.

7.2.1.4 - Unit Test Cases

Test Scenario ID	TS01
Test Priority	P1
Unit	Trained Model
Functional Requirement	FR-02, FR-03, FR-04
Overall Status	100%
Test Description	The application should be able to predict whether EEG data is showing autism spectrum disorder or not.

Test Case ID	Test case description	Expected Result	Actual Result	Status
T1	testing the predictive	Expected to see the prediction of	All the algorithms are predicting	Passed

	abilities of the algorithms. Start the Python program that contains the forecasting algorithms.	each algorithm	values for test data	100%
T2	analyzing the models to see if they were created using the allocated algorithms	The model summary should contain all the layers and details of the parameters	The model summary shows all the parameters and the details of the layers	Passed 100%
T3	Testing the accuracy of the models	Model accuracy should be in the range of 70-100%	Model accuracies are in the range of 70-100%	Passed 100%

Table 7.1-Unit testing for trained models

Test Scenario ID		TS02		
Test Priority		P1		
Unit		EEG data		
Functional Requirement		FR-01, FR-05, FR-06		
Overall Status		100%		
Test Description		The user should be able to upload EEG data, EEG data need to get pre-processed and creates frequency bands data set.		
Test Case ID	Test case description	Expected Result	Actual Result	Status

T4	The user should be able to upload EEG data files to the system	EEG data files are uploaded to a specific location in the system without any errors	EEG data is uploaded to the pointed directory without any issue, if there is an issue it will notice by the user	Passed 100%
T5	EEG data get pre-processed (Noise removal, ICA and column interpolation)	EEG data should get pre-processed and saved pre-processed EEG data in another file	EEG data get pre-processed and saved as a pre-processed data file in the secure location	Passed 100%
T6	Creating Bands data set for the uploaded EEG data	Pre-processed data should be converted to frequency bands data set that can be used in the pre-trained model	Pre-processed data is used and created frequency bands data set using it and apply it to the pre-trained model	Passed 100%

Table 7.2-Unit testing for EEG data

The results of the unit tests used to test the functional requirements above have been shown. The unit test only checks if the functionality is working at the unit level. To check all the flow is working their functional requirements need to be tested using integration testing as well.

7.2.2 Integration Testing

Integration tests are performed to ensure that the combined software modules and units are operating collectively according to plan. The created application will go through these steps. Integration tests to determine whether the previously tested units are functional following the integration.

7.2.2.1 Testing Method

To test the integrated tests, the white box testing approach is used.

7.2.2.2 Testing Tools

Google chrome debug tools, PyCharm and VSCode are used as the testing tools for integration testing.

7.2.2.3 Integration Test Cases

Test Scenario ID	TS03
Test Priority	P1
Integration Unit	Web-based prediction module
Units integrated	● Web-based form with multiple file upload unit ● Backend file save unit ● EEG data preprocess unit ● Noise filtering unit ● Frequency bands data generate unit ● Prediction unit ● Pre-trained models
Overall Status	100%
Test Description	Testing the integrated modules for forecasting functionality

Test Case ID	Test case description	Expected Result	Actual Result	Status
T7	Test if multiple	Files uploaded to a secured	When the user uploads multiple files using	Passed

	files uploading unit works	location in the backend	multiple files upload form files saved in the secured location in the backend	100%
T8	Test data pre-processing works	Pre-processed data need to be generated and saved in the backend	The pre-processed data file is generated and saved in the backend with the participant ID included	Passed 100%
T9	Creating Bands data set for the pre-processed data	Create a frequency bands data file that can be used in the model prediction from the pre-processed data	Pre-processed data is used and created frequency bands data set and saved in the backend secured location	Passed 100%
T10	The model prediction is shown in the front-end clearly	Model prediction for the uploaded data is shown in the user interface	The model prediction is shown in the user interface as “positive” or “negative”	Passed 100%

Table 7.3-Integration tests

The integrated tests mentioned above demonstrate that the units are satisfactorily integrated.

7.2.3 System Testing

System testing examines how well modules have been fully integrated into a new system. Therefore, the finished new application will be tested using system tests.

7.2.3.1 Testing Method

Black-box testing will be used to test the system functionalities.

7.2.3.2 Testing Tool

PyCharm and VSCode will be used to test the new system.

7.2.3.3 System Test Cases.

Test Scenario ID		TS04		
Test Priority		P1		
Functions		Main user functions		
Overall Status		Passed - 80%		
Test Description		The application should be able to predict whether EEG data is showing autism spectrum disorder or not.		
Test Case ID	Test case description	Expected Result	Actual Result	Status
T11	The user can see the landing page.	When a user comes to the website, can see the landing page with “about the system” and “how to use”	Users can see the landing page with “about the system” and” how to use”	Passed 100%
T12	The users can upload multiple EEG data files	Users can upload multiple files and can see the indicator with the upload status	Users can upload multiple files and can see the upload status	Passed 100%
T13	The predict button is disabled until the	Until the user uploads files, the predict button is disabled	Until the user uploads files, the predict button is disabled	Passed 100%

	user uploads files			
T14	Users can see the predicted result	After the EEG data upload user click the predict button user can see whether the EEG data shows ASD positive or not	Users can see ASD positive or negative in UI after uploading EEG data and clicking the predict button	Passed 100%
T15	The user can only upload .csv files	If a user uploads a file which uses another extension the error message needs to popup	The error message doesn't pop up	Passed 0%

Table 7.4-System test cases

According to the system test cases mentioned above, 80% of the test cases for all the main components passed. The pass rate for both the obligatory prerequisites, FR-01, FR-02, FR-03 and FR-04 is 100%. The error message can be completed but it needs to be checked if the system is going to use raw EEG data files as well. The results of the tests mentioned above can be regarded as satisfactory in terms of the implementation of the functional requirements.

7.3 Non-Functional Requirements Testing

Following the collection of requirements, Section 3.8.2 of Chapter 03 - Requirement Analysis had five non-functional needs. In this section, the accuracy, performance, usability, and compatibility of the system will be evaluated based on the specified non-functional needs. The implementation and results of the chosen non-functional requirement testing are described in detail in the following sections.

7.3.1 Accuracy Testing

An indicator of the model's performance across all classes is accuracy. When all classes are equally important, it is helpful. The number of accurate forecasts divided by the total number of predictions is used to compute it. The author tested accuracy for all neural network models as shown in table 7:4; To get the accuracies the author used 30% as testing data as shown in figure 6:27.

I. Precision

The accuracy is calculated as the ratio of Positive samples that were correctly classified to all samples that were classified as Positive (either correctly or incorrectly). The precision measures how well the model categorizes a sample as positive.

II. Recall

The recall is determined as the proportion of Positive samples that were correctly identified as Positive to all Positive samples. The recall gauges how well the model can identify Positive samples. The more positive samples that are identified, the larger the recall.

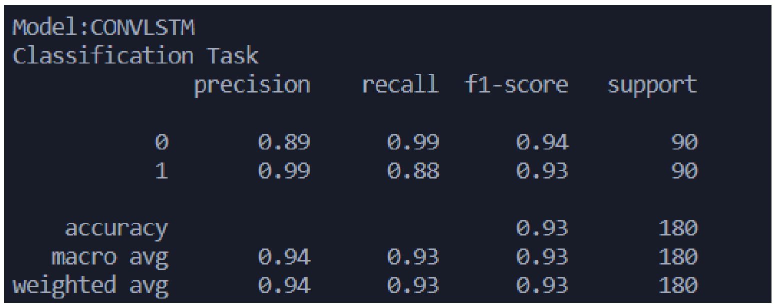
III. F1- Score

The accuracy of a model on a dataset is measured by the F1-score. It's applied to assess binary classification methods that label examples as "positive" or "negative." The harmonic mean of the model's precision and recall is known as the F-score, which is a method of combining the model's precision and recall.

Also in figure 7.1, the author shows the two highest accuracies obtained models which are ConvLSTM and figure 7:2 CapsNet with precision, recall and F1 score. In Appendix E: E.1 can find the other models' performances. Test results comparison and evaluation are done under the Evaluation of Test Results section in the Evaluation chapter.

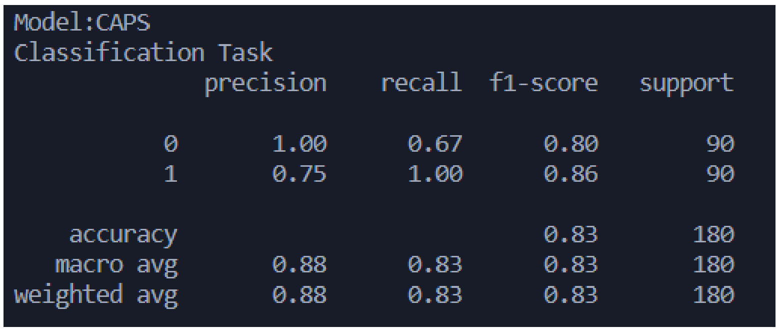
Model Name	Accuracy
Conv	74%
LSTM	76%
Bi-LSTM	72%
GRU	78%
CapsNet	83%
ConvLSTM	93%

Table 7.5-Accuracies of trained models



```
Model:CONVLSTM
Classification Task
              precision    recall  f1-score   support

           0       0.89      0.99      0.94        90
           1       0.99      0.88      0.93        90

    accuracy                           0.93       180
   macro avg       0.94      0.93      0.93       180
weighted avg       0.94      0.93      0.93       180
```

Figure 7.1-ConvLSTM model accuracy



```
Model:CAPS
Classification Task
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        90
           1       0.75      1.00      0.86        90

    accuracy                           0.83       180
   macro avg       0.88      0.83      0.83       180
weighted avg       0.88      0.83      0.83       180
```

Figure 7.2-CapsNet model accuracy

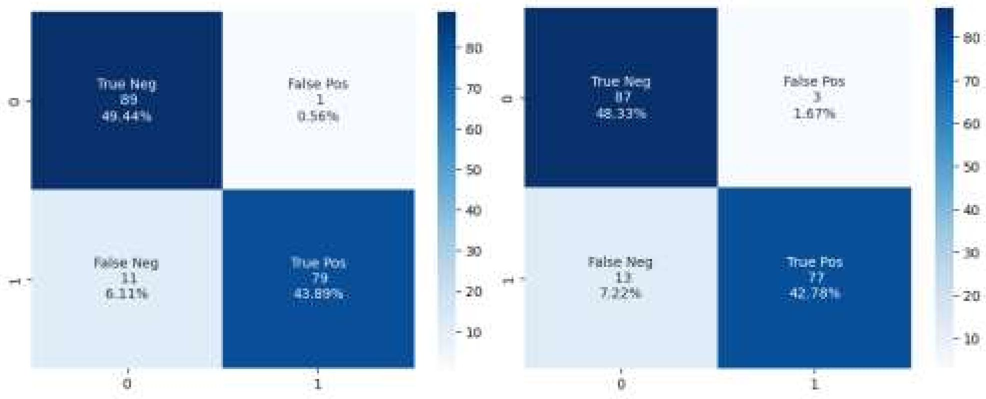


Figure 7.3-Confusion matrix of ConvLSTM(left) and CapsNet(right)

7.3.2 Performance Testing

7.3.2.1 Trained Model Performance

The below table shows the time to execute the predicted model without pre-processing the EEG data.

Model Name	Time (s)
Conv	3.7122766971588135
LSTM	4.1479198932647705
Bi-LSTM	7.035894870758057
GRU	5.2730913162231445

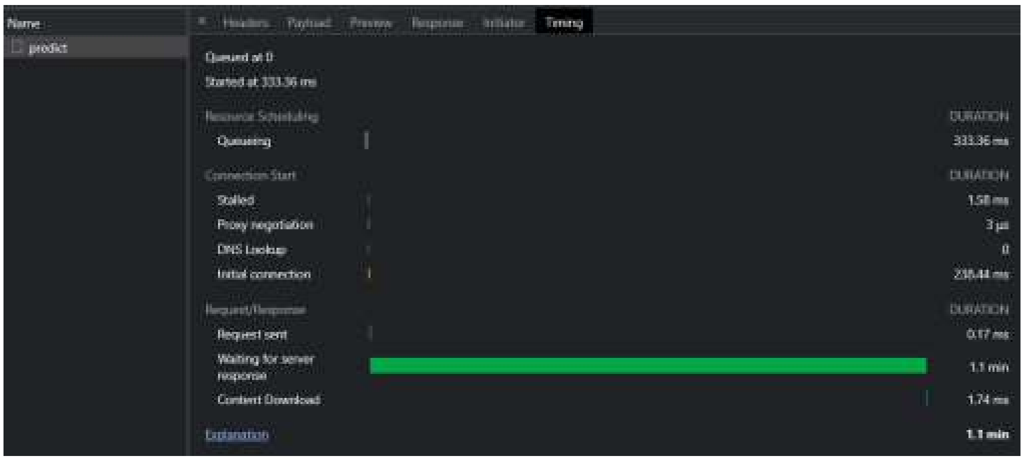
CapsNet	3.7594263553619385
ConvLSTM	1.9570257663726807

Table 7.6-Time taken to execute a prediction

ConvLSTM, Conv and CapsNet show performance better than the other models.

7.3.2.2 Implemented System Performance

The predicted API call time was used to evaluate the developed application's performance. The image below shows it takes 61 seconds for the API to be able to send the prediction. This takes this much time because of the EEG data pre-processing. This demonstrates how efficiently the system operates.



```
Queued at 0
Started at 333.36 ms

Resource Scheduling                 DURATION
  Queueing                          333.36 ms
Connection Start                    DURATION
  Stalled                           1.58 ms
  Proxy negotiation                 3 µs
  DNS Lookup                        0
  Initial connection                238.44 ms
Request/Response                    DURATION
  Request sent                      0.17 ms
  Waiting for server response       1.1 min
  Content Download                  1.74 ms
                                    1.1 min
```

Table 7.7-API performance

The environment details that the author carried out performance testing are in Appendix E: E:2.

7.3.3 Usability Testing

Usability testing assures that the system application is easy for customers to use. To assure usability, the system was introduced to 5 regular users ranging in age from 16 to 50, as well as to 6 students who are involved in related research in the big data analysis field. Normal users were pleased with the system's simplicity because it allows them to ask a question and immediately receive a response. Researchers were interested in the system because they wanted to know which measurements had led to inaccurate results. Overall, the usability of the system received positive comments from every user.

7.3.4 Maintainability and Extensibility

Any system that is built should be planned so that future upgrades to the system may be properly managed. It is simple to improve each module because the system was developed in modules. For instance, the prediction model and upload files are intended as two distinct components that may be changed and reused on any other pages.

7.4 Chapter Summary

This chapter was mostly concerned with testing the prototype in line with the testing goals. To ensure that the functional and non-functional components were implemented correctly, test cases were created. For the functional testing, Unit testing, Integration testing and System testing were carried out. And for the testing of non-functional requirements Accuracy testing, performance testing and usability testing were carried out. From the accuracy testing, two out of the six models, namely CapsNet and ConvLSTM, gave 83% and 93% accuracies respectively. Minor issues were found during testing which were fixed.

Chapter 08 - Evaluation

8.1 Chapter Overview

Based on the discussion from the previous chapters, we will critically examine each stage of the project in this chapter and assess the prototype, findings, designs, and outcomes. The following regions will receive the majority of our analysis' attention.

- Quantitative Evaluation
- Achievements of the Aim, Research Questions, Research Objectives and Operational Objectives
- Prototype evaluation
- Qualitative Evaluation
- Critical Self-Evaluation

8.2 Quantitative Evaluation

8.2.1 Defining the Benchmark / Control Study

There have been similar studies have been done as shown in Table 2:1 of the literature review. There have been neural network models implemented using CNN, LSTM, Bi-LSTM and CNNG. But there is no Neural network model implemented using CapsNet and ConvLSTM.

Also, Jayawardana, Jaime and Jayarathna 2019 were able to use this particular dataset and had to be able to implement a CNN model which has an accuracy of 95%. Ali et al. 2020 were able to obtain 80% using CNN. Also, Ali et al.2020 were able to obtain 99.6% accuracy using the Bi-LSTM model. The benchmark was set to obtain a more accurate model using other neural network model approaches and using EEG data. Table 8.1 shows a comparison between the proposed system and the other systems that exist.

	Proposed System	Jayawardana, Jaime and Jayarathna 2019	Ali et al. 2021	Ali et al. 2020
Dataset	**EEG data from 17 Participants Between 5 to 17 years of age**	EEG data from 17 Participants. Between 5 to	EEG data from SFARI involving 57 individuals	20 individuals dataset provided by King Abdulaziz

		17 years of age	(but 4 cases were excluded)	University (KAU) Brain-Computer Interface (BCI) Group, Jeddah, Saudi Arabia
Proposed Model	**ConvLSTM**	Random forest	Bidirectional LSTM	CNN
Model Accuracies	**93%**	97%	99.60%	80%

Table 8.1-Table of Benchmarking

8.2.2 Non-Functional Requirement Benchmark

The non-functional test case, NFR-1, is described in the test chapter. As shown in Table 7:4 the author was able to develop several models and among them, the best models were ConvLSTM and CapsNet which have an accuracy of 93% and 83% respectively. The accuracies obtained for this research are promising and compared to the CNN method proposed by Ali et al. 2020 the neural network model author proposed obtained more accuracy. But the random forest classifier proposed by Jayawardana, Jaime and Jayarathna 2019 has 4% more accuracy than the model proposed by the author.

8.2.3 Evaluation of Test Results

All of the tests carried out to create the model were described in the testing chapter. The development of an accurate model was one of the goals of this study. According to the benchmarking the author did, the two proposed models have good accuracy, precision, recall, and f1 scores. This illustrates how the tested final model has closed the necessary gap.

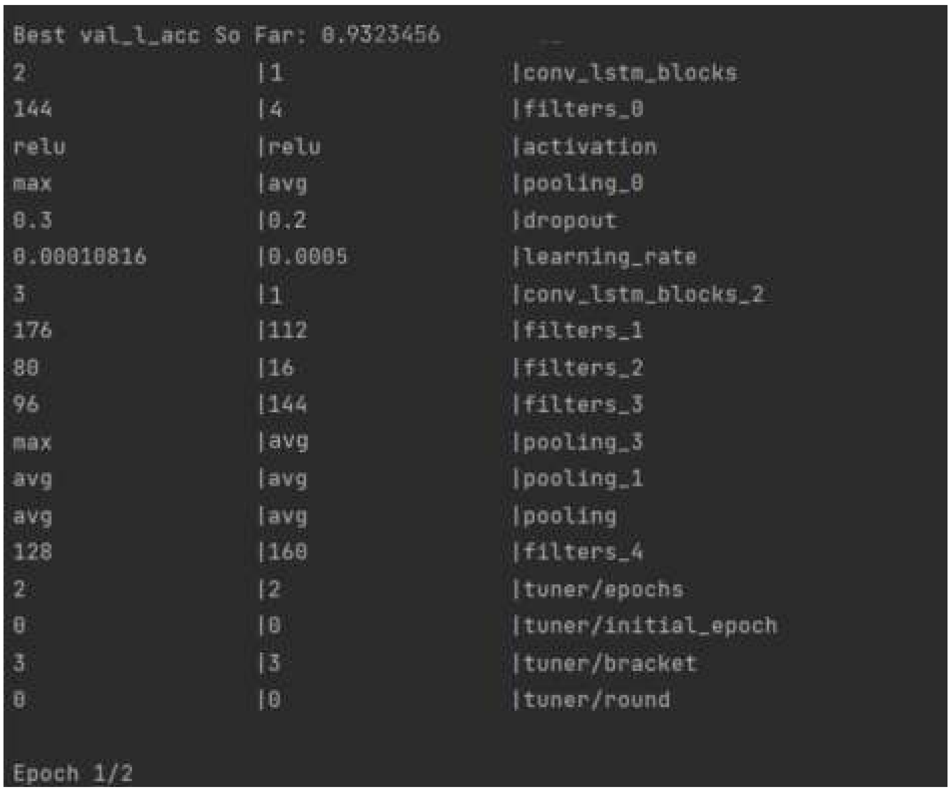
However, the accuracy can be increased by expanding the dataset and doing more experiments. Expanding the data set was not possible because the dataset the author used was the whole dataset he obtained. So the author tried out six neural network models and try to find the best accuracy, precision, recall and F1-score model by trying out several combinations. One way is to find out the optimum neural network layers and other parameters is brute-forcing, and the other way is to do hyperparameter tuning using the "Keras tuner"

8.2.3.1 Hyperparameter Tune by Brute-forcing

Hyperparameters are values that must be initialized for the network. The network cannot learn these values during training. Examples of hyperparameters in convolutional neural networks include kernel size, number of layers, activation function, loss function, optimizer, batch size, number of training epochs, etc. The author carried out a brute-force method for hyperparameter tuning for neural networks to find out at what time the accuracy tends to drop. He tried out only the number of layers for this method. To find the limit where the no of layers is dropping the accuracy is marked and used for the hyperparameter tuning using the "Keras Tuner". The number of layers brute forced for ConvLSTM is in Appendix F: F:1

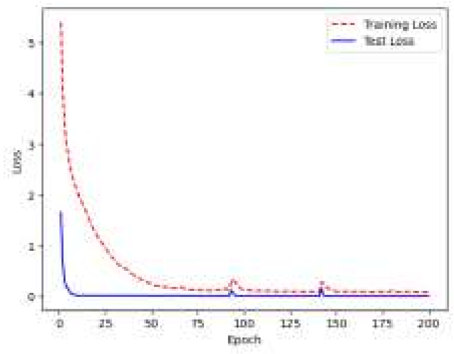
8.2.3.2 Hyperparameter Tuning using Keras Tuner

The author tried out hyperparameter tuning for all six models using different combinations. The best accuracy and F1 score were obtained from the ConvLSTM model which uses two ConvLSTM layers and one dense layer. Also, different learning rates are tested but the optimum rate found was 0.0005. Figure 8.1 shows the optimum hyperparameter tuning for ConvLSTM. Appendix F, F:2 shows the source code for hyperparameter tuning for ConvLSTM.



```
Best val_l_acc So Far: 0.9323456
2                 |1                  |conv_lstm_blocks
144               |4                  |filters_0
relu              |relu               |activation
max               |avg                |pooling_0
0.3               |0.2                |dropout
0.00010816        |0.0005             |learning_rate
3                 |1                  |conv_lstm_blocks_2
176               |112                |filters_1
80                |16                 |filters_2
96                |144                |filters_3
max               |avg                |pooling_3
avg               |avg                |pooling_1
avg               |avg                |pooling
128               |160                |filters_4
2                 |2                  |tuner/epochs
0                 |0                  |tuner/initial_epoch
3                 |3                  |tuner/bracket
0                 |0                  |tuner/round


Epoch 1/2
```

Figure 8.1-ConvLSTM model Optimum Hyperparameters



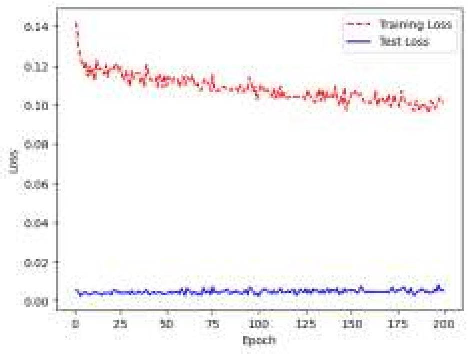
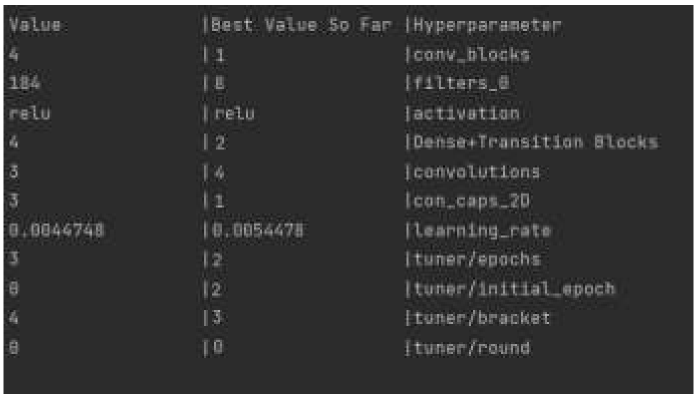


Figure 8.2-Train Test Loss ConvLSTM　　*Figure 8.3-Train-Test Loss CapsNet*

The best accuracy and F1 score were obtained from the CapsNet model which uses one Convolution layer and one dense caps layer. Also, different learning rates are tested but the optimum rate found was 0.0005. Figure 8.2 shows the optimum hyperparameter tuning for CapsNet. Appendix F, F:3 shows the source code for hyperparameter tuning for CapsNet.



```
Value               |Best Value So Far |Hyperparameter
4                   |1                 |conv_blocks
184                 |8                 |filters_0
relu                |relu              |activation
4                   |2                 |Dense+Transition Blocks
3                   |4                 |convolutions
3                   |1                 |con_caps_2D
0.0044748           |0.0054478         |learning_rate
3                   |2                 |tuner/epochs
0                   |2                 |tuner/initial_epoch
4                   |3                 |tuner/bracket
0                   |0                 |tuner/round
```

Figure 8.4-CapsNetmodel Optimum Hyperparameters

8.2.3.3 Model Accuracies after Parameter Tuning

The below figure shows the accuracies obtained after parameter tuning for neural networks.

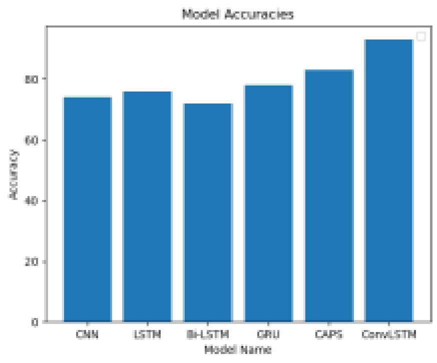


Figure 8.5-Best accuracy comparison

As can observe ConvLSTM and CapsNet models have better accuracies than other models. Therefore, ConvLSTM and CapsNet are more suitable for the dataset with accurate predictions.

8.2.3.4 Summary of Test Results

The author tried our numerous hyperparameter combinations such as layers, epochs, optimizers, activation functions, kernel size and learning rates because the control of a deep learning model's behaviour requires hyperparameter adjustment. Our predicted model parameters will give fewer results if our hyperparameters aren't properly tuned to minimize the loss function. This indicates that our model has more flaws.

After tuning the hyperparameters the author was able to get accuracies of 93% and 83% for ConvLSTM and CapsNet models. While conducting the hyperparameter tuning author was make sure not to overfit the model to the particular dataset.

8.3 Achievement of Aim

The aim of the research was to "Design, develop, test and evaluate behaviour-independent mechanisms to diagnose the prevalence of autism spectrum disorder in children at an early age"

At the end of this research project author was able to successfully implement a working prototype for an autism spectrum disorder predicting system using EEG data. For this author implemented several neural network-based models and was able to achieve 93% accuracy for the best model and also author was able to develop neural network models using ConvLSTM and CapsNet which is a new finding in detecting ASD using neural networks on EEG data.

8.4 Answer the Research Questions

RQ01 - How to classify autism spectrum disorder prevalence using neural networks based on EEG data at early ages?

This was a new concept for the author but after doing research the author was able to get a clear understanding of EEG data and neural networks. The hardest part was collecting a good dataset and pre-processing the EEG data. The author was able to find a data set and was able to create a pre-processing pipeline. Also, after doing some research, the author was able to use neural network models on the pre-processed EEG data for predictions.

RQ02 - How to optimize current classification techniques provided for the detection of autism spectrum disorder using EEG using neural networks?

There have been several studies done to classify ASD using EEG data and the author was able to use the data set and create two new neural network models which have accuracies of 83% and 93% to detect ASD in early ages using EEG data. Also, the author was able to improve the accuracies of other neural network models as well.

8.5 Achievement of Research Objectives

The achievement of the research objectives can be highlighted below.

Research Objective	Status
To identify the most suitable dataset for this classification.	Achieved
Clean and remove the noise in EEG data	Achieved
To identify the best approaches to implement the system by analyzing existing technologies used in previous studies.	Achieved
To implement a classifier using neural networks to classify ASD using EEG data, which produces more accurate results with a better performance.	Achieved
To evaluate the implemented system.	Achieved

Table 8.2-Achievement of Research Objectives

8.6 Achievement of Operational Objectives

The achievement of operational objectives can be highlighted below.

Operational Objective	Status
Prepare Project Proposal	Completed
Conduct Literature Review	Completed
Requirement Gathering	Completed
Design the system	Completed

Prototype Development	Completed
Testing and evaluation of the prototype	Completed
Complete the project and submit deliverables	Completed

Table 8.3-Status of Operational Objectives

8.7 Prototype Quality Evaluation

The prototype evaluation was done below criteria after presenting the system to 10 users who have experience in full stack development

- Graphical user interface (GUI)
- User experience
- Comments and improvements recommended
- Security
- Accessibility
- Availability

8.7.1 Prototype User Experience

The user experience evaluation was done for the prototype by using a simple questionnaire from the 10 users and their answers are summarized below. In Appendix F: F.4 can find the responses.

Question	Answer Summary
What is your first impression when you access the website?	Excellent - 2 /Good - 6 /Ok - 2 /Bad - 0
Did it load quickly?	Yes - 9 / No - 1
Did you try to upload EEG data?	Yes -10 / No -0
Did you try to make predictions using EEG data?	Yes -10 / No -0
Do you like the application?	Yes-10 / No- 0
Do you think it is user-friendly?	Excellent - 8 / Good - 1 / Ok -1/ Bad-0
What are the difficulties faced when using the application?	Takes time to do the prediction

Table 8.4-User Experience-Questions and results

8.7.2 User Interface Friendliness

Eight out of ten participants in the aforementioned sample questionnaire agreed that the program is user-friendly. Additionally, it appears from the submitted answers that the program is either Very Easy to Use or Not Bad. As a result, can conclude that using the user interface is not too challenging. Additionally, all 10 have successfully predicted a test. This demonstrates that everyone has successfully used the application.

8.7.3 Evaluation on GUI

According to Question 1: What is your first impression when you access the website? Got the responses as two people responded "Excellent," Six people said "Good," and two people stated "Ok." With 8 Good remarks, it is obvious that everyone thought the website was at least "good," which may mean that the site's GUI is good for the end users.

8.7.4 Practicality Issues

The only feedback got for the application is that it takes considerable time to do the prediction. This is happening because the system takes some time to pre-process the data and do the prediction. This can improve by using a subsystem which has more performance than the server and pass the data to that server to do the pre-processing task.

8.7.5 Security

Basic security precautions were given to the designed application. The EEG data upload is done securely but the application doesn't have login functionality. So that functionality can be developed in the next iterations.

8.7.6 Accessibility

Accessibility refers to how simple it is to use this software. The application was created as a web app, allowing users to access it from any portable device with an internet connection, such as a laptop, mobile phone, or tablet. This is a desired but non-functional condition.

8.7.7 Availability

The application was developed using Angular as the front end and Flask as the back end. Angular is one of the most popular frontend frameworks and Flask is a stable backend development python framework. The flask server can deploy in an AWS EC2 instance without issue and can attach to a load balancer. This will enable good availability.

8.8 Qualitative Evaluation

For the qualitative evaluation, the system was introduced to several domain experts and users. Each of them has given their evaluation. This evaluation was accomplished by analyzing the research concept, technical approach, evaluation criteria and usability. University students, the general public and researchers who conduct similar big data analytics have been considered as users. Appendix F.F.5 shows the ratings received for each evaluation.

8.8.1 Project Concept Evaluation

Feedback 01	This is a good topic because there are a lot of autistic people but only, they are recognized when they act differently. But using this can detect autism at an early age. Senior Software Engineer, WSO2 MSc BDA Postgraduate
Feedback 02	Applying neural networks to detect autism using EEG data is a new state art technique and if it can use for early age children it will be a great chance to detect autism Senior Software Engineer, CodeGen International
Feedback 03	This is great research for the advancement of the biomedical field Software Engineer, WSO2
Review	It is clear from concept evaluations that the project implemented is very beneficial to both general users and researchers to enable future analytics.

Table 8.5-Evaluation of project concept

8.8.2 Evaluation of Technical Approach

Feedback 01	Neural network models can identify hidden patterns in data and have provided to be a reliable model to predict certain outputs. thereby this approach is good

	Senior Software Engineer, WSO2 MSc BDA Postgraduate
Feedback 02	Applying neural networks to detect autism using EEG data is a new state art technique and if it can use for early age children it will be a great chance to detect autism Senior Software Engineer, CodeGen International
Feedback 03	As a big data project, this candidate is to understand and efficiently preprocess EEG data, which is time series data, and this can apply to other areas as well. Software Engineer, WSO2
Review	Using neural networks and EEG data to detect autism at an early age was selected as a good approach by most of the evaluators and they also mentioned this can be applied to other areas as well.

Table 8.6-Evaluation of Technical approach

8.8.3 Evaluation of Evaluation Criteria

Feedback 01	Since the model accuracies have shown to be significantly high, this can be considered reliable Senior Data Analyst - MAS Holdings Pvt Ltd MSc BDA Postgraduate
Feedback 02	Using the pre-processed data, he was able to successfully create models which have accuracies over 80% is remarkable Senior Software Engineer, CodeGen International
Feedback 03	They are good and if the accuracies if they were like 100% that means the models are overfitting. But here he was able to obtain accuracies over 80% for CapsNet and ConvLSTM seems reliable.

	Software Engineer, WSO2
Review	Users are satisfied with the model accuracy and also, they are satisfied that the author was able to develop models which have accuracies over 80%

Table 8.7-Evaluation of Evaluation criteria

8.8.3 Evaluation of Usability

Feedback 01	It's a minimal task the user has to do to view an output which is what is expected. Thereby the usability is user-friendly. Senior Data Analyst - MAS Holdings Pvt Ltd MSc BDA Postgraduate
Feedback 02	The system is very easy to use. it has all the documentation and how to use it, but the only drawback I see is it takes considerable time to give the results. Senior Software Engineer, CodeGen International
Feedback 03	The user interface seems simple and easy to use. Software Engineer, WSO2
Review	Users are satisfied with the usability of the system and the only drawback they see is taking considerable time to show the results. It is also mentioned in the prototype evaluation how to overcome this is mentioned in the prototype evaluation section

Table 8.8-Evaluation of Usability

8.9 Critical Self-Evaluation

The critical self-evaluation that follows focuses on and evaluates the whole research from the viewpoint of the author.

Criteria	Self-Evaluation by Author
Selected research topic	There have been several studies have been conducted based on EEG data to diagnose Autism using multiple classification techniques such as Random Forest, KNN, Decision Tree, etc., but not many studies have been conducted to detect Autism using Electroencephalogram data (EEG) using neural networks, in early stages of children. Therefore, the author thinks it is crucial to find a solution using neural networks.
The novelty of the research area	The author was able to successfully develop a ConvLSTM neural network model which has 93% accuracy and was able to implement a neural network model following the Capsule layer implementation which has 83% accuracy. There have been several neural network models developed for this research area, but no ConvLSTM or CapsNet models developed. The author believes these models are performing well and enough for the research with the time limitations.
Scope and the depth of the research	This study uses EEG data and neural networks to identify autism spectrum disorder. The author was able to successfully develop a pipeline to pre-process and remove noise in EEG data and tried out more than 100 models by changing hyperparameters for neural networks for the 6 neural network models and hyperparameter tuning using the Keras tuner. The scope and depth of the research are sufficient for the given time and resource constraints
Development approach undertaken	The author developed 2 models which have accuracies of 83% and 93% using CapsNet and ConvLSTM which can successfully predict ASD the optimum hyperparameters are chosen by following hyperparameter tuning as well. None of the researchers has created ConvLSTM or CapsNet models for autism spectrum disorder classification using EEG data.
Contribution	This is explained in Contribution to new knowledge in the

	Conclusion chapter
Evaluation approach	The author uses evaluation metrics such as accuracy, F1 score, precision and recall discussed in the testing chapter. While developing the models he tested them with most of the possible approaches. This is discussed in the evaluation of test results section. The author believes that the evaluation of the proposed model and findings were presented in the best way possible.
The user experience of the prototype	The author created a prototype using Angular and Flask which supports almost all browsers and developed it in an easy way for maintainability. Even though it is a prototype author believes that he was able to implement it user-friendly and easy-to-use application.
Limitations of the research and future enhancements	More about this is discussed in the Limitations and future enhancement chapter in the Conclusion Chapter.

Table 8.9-Critical Self-Evaluation

8.10 Ethical Review

The software used to implement the prototype is all open source. To use the data set got permission from Prof.Sampath Jayarathna. Also, when getting the feedback from users none of the identities was taken. Before receiving input, the author made sure to fully clarify the process for soliciting opinions from subject-matter experts. No plagiarized content was utilized, and the author makes care to provide credit to the appropriate people by citing any resources that were used. this is explained and discussed in legal and ethical considerations under the project management methodology selection chapter.

8.11 Chapter Summary

This chapter discusses in detail the method of evaluation and the strategy that was used. The quantitative evaluation defined the benchmark and in which line of the benchmark the proposed model lies. Also discussed the results he obtained for the trained models and what are the steps carried out to find optimal hyperparameters. Then the achievements of the aim, research questions, research objectives and operational objectives were discussed thoroughly.

Next, the author carried out prototype evaluation and qualitative evaluation. Finally, the author’s self-evaluation regarding the research and ethical review is done.

Chapter 09 - Conclusion

9.1 Chapter Overview

This chapter concludes the project report with a thorough analysis of the challenges the author encountered in achieving the objectives he set out to do as well as an evaluation of the knowledge base contributions he made. Then the author examines the information he learned from the modules as well; along with the expertise, the author had to self-learn. Finally, discuss the system's drawbacks and areas for future development.

9.2 Contribution to New Knowledge

9.2.1 Domain-Related Contribution

While conducting the literature review it was identified that the detection of autism spectrum disorder on EEG data using machine learning techniques is a mature research area most of the researchers used machine learning techniques such as Random Forest, Naive Bayes, etc. But using neural networks to detect ASD using EEG data; is a new research area and very less researchers have researched this. Therefore, implementing a system which can detect autism spectrum disorder using neural networks on EEG data is a new implementation which helps to identify autism spectrum disorder without behaviour-independent characteristics.

9.2.2 Technical Contribution

There was very little research which has used EEG data and tried to detect ASD at early ages using neural networks and the main problem the researchers faced was pre-processing EEG data and finding a good neural network model that can do predictions accurately. But here author was able to develop 2 new neural network models; ConvLSTM and CapsNet for detecting ASD using EEG data which never try out by any research with accuracies of 93% and 83% respectively. Other than the mentioned two models; the researcher used CNN, LSTM, Bi-LSTM and GRU models and tried to optimize them by hyperparameter tuning without making it overfit to the dataset. Therefore, this can be considered as a significant contribution to the selected research area. The future researchers who will be examining the system, will be able to use the pre-processing techniques used by the author and the implementation followed by the author to create new models and experiment new solutions to similar or related areas of research. This will positively contribute to enhancement of future solutions by overcoming any drawbacks identified in the future.

The author believes he made enough domain and technology contributions within a limited period.

9.3 Challenges Faced

9.3.1 Finding the dataset

EEG data was challenging to find because of confidential reasons. Some organizations which have data sets for EEG required to go through a lot of processes and requested data sets were not given to the other regions in the world. So, author send emails to the researchers who have done previous research using EEG data sets and were able to obtain a data set from Prof.Sampath Jayarathna.

9.3.2 Lack of knowledge in pre-processing EEG data

EEG data are different from the other types of data that the author used during his previous developments and had to read and know about how to pre-process EEG data. Removing noises and independent component analysis are some of the issues that the author faces during pre-processing of EEG data.

9.3.2 Time limitations

Compared to other researchers, the author chose to finish the project in two semesters of part-time study. Additionally, the author was able to allocate a few hours for research daily. As a result, it was challenging to gather data and interact with subject matter experts because the study required a thorough literature review. But it managed to finish the job in the allowed time.

9.4 Usage of taught modules

The majority of the MSc Big Data Analytics program modules that were taught were beneficial for finishing the research project. The modules that are listed below had the most impact on the author for succeeding in this research.

- **CMM - 703 - Data Analytics**

This module helped with getting an understanding on how data distribution works, dataset analytics, and learning to manage the data efficiently.

- **CMM - 708 - Research Methods**

  Learning this module and completing the required deliverables will equip you with the expertise and knowledge to evaluate a research project's many components critically and to do so with an effective justification. It also demonstrated the proper methodology for conducting topic research. This research assignment was greatly helped by the example project proposals that were completed as a class.

9.5 New Knowledge Gained from Taught Modules

- Through a review of the literature, the author was able to learn more about the chosen domain area and further his understanding of the goal and objectives.
- Able to learn neural networks and how to use them
- Able to learn how to work with time-series data such as EEG and how to pre-process them
- Time management, problem-solving, critical thinking, and writing of formal documentation also improved.

9.6 Limitations and Future Enhancements

9.6.1 Limitations

The proposed system has limitations below defined.

- The proposed system is mainly focused on EEG data, and it uses EEG data on .csv files, not the raw EEG data.
- The proposed system is focused on EEG data captured while doing ADOS-2 assessment by participants only.
- The author was able to find a data set of 17 participants as here author used a deep learning approach if there is a bigger dataset more performance could be acquired.
- The accuracies of models ConvLSTM and CapsNet are 93% and 83% of models perform accurately because F1 score, precision and recall values are precise.
- The proposed prototype takes a considerable amount of time to do the prediction because EEG data pre-processing needs a higher-performance system.

9.6.2 Future Enhancements

The author proposes future enhancements which can do to the proposed system

- Using a high participant dataset can improve the accuracy of the model and do more accurate predictions.
- Improve the pre-process pipeline of EEG data not only for .csv files but for raw EEG data files as well.
- Improve prototype performance using parallel functions and using a high-performance system.
- Enhance the web application to a MATLAB extension which can use with EEGLab as well

9.7 Chapter Summary

The project's concluding ideas were covered in this chapter. In the domain contributions section, the author highlights the prototype he developed to detect ASD using EEG data with neural networks. In the technical contributions section, he highlights the development of 2 new neural network models; ConvLSTM and CapsNet for detecting ASD using EEG data which were never tried out before by any research with accuracies of 93% and 83% respectively. The author recognises the dataset collection as one of his main challenges faced during this project and discusses other challenges he faced as well. Then he describes the usage of taught modules and new knowledge gained from taught modules. The project's limitations and room for improvement were then examined

References

ALI, N.A. et al., 2021. LSTM-based Electroencephalogram Classification on Autism Spectrum Disorder. International Journal of Integrated Engineering, 13(6), pp. 321–329.

ALI, N.A. et al., 2020. Autism spectrum disorder classification on electroencephalogram signal using deep learning algorithm. IAES International Journal of Artificial Intelligence, 9(1), pp. 91–99.

AMERICAN PSYCHIATRIC ASSOCIATION, 2013. Diagnostic and Statistical Manual of Mental Disorders. American Psychiatric Association.

CHANG, C.-Y. et al., 2018. Evaluation of artifact subspace reconstruction for automatic EEG artifact removal. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 1242–1245.

CHO, K. et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. ArXiv Preprint ArXiv:1406.1078.

CHUNG, J. et al., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. ArXiv Preprint ArXiv:1412.3555.

COHEN, M.X., 2019. A better way to define and describe Morlet wavelets for time-frequency analysis. NeuroImage, 199, pp. 81–86.

COMON, P., 1994. Independent component analysis, a new concept? Signal Processing, 36(3), pp. 287–314.

COOK, J. et al., 2021. Camouflaging in autism: A systematic review. Clinical Psychology Review, 89, p. 102080.

DENEKE, E., 2022. How to reduce noise in EEG recordings [11 solutions]. [online]. Mentalab. Available from: https://mentalab.com/reduce-noise-in-eeg-recordings/ [Accessed 20 December 2022].

DJEMAL, R. et al., 2017. EEG-based computer aided diagnosis of autism spectrum disorder using wavelet, entropy, and ANN. BioMed Research International, 2017.

DOLAN, W., n.d. Using the autism diagnostic observation schedule (ADOS) to discriminate between children with autism and children with language impairments without autism.

GAIKWAD, K.M. and CHAVAN, M.S., 2014. Removal of high frequency noise from ECG signal using digital IIR butterworth filter. In: 2014 IEEE Global Conference on Wireless Computing & Networking (GCWCN). pp. 121–124.

HAPUTHANTHRI, D. et al., 2019. An EEG based channel optimized classification approach for autism spectrum disorder. In: 2019 Moratuwa Engineering Research Conference (MERCon). pp. 123–128.

HOCHREITER, S. and SCHMIDHUBER, J., 1997. Long short-term memory. Neural Computation, 9(8), pp. 1735–1780.

JAMAL, W. et al., 2014. Classification of autism spectrum disorder using supervised learning of brain connectivity measures extracted from synchrostates. Journal of Neural Engineering, 11(4), p. 46019.

JAYAWARDANA, Y., JAIME, M. and JAYARATHNA, S., 2019. Analysis of temporal relationships between ASD and brain activity through EEG and machine learning. In: 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). pp. 151–158.

JIANG, W. et al., 2022. CNNG: A Convolutional Neural Networks with Gated Recurrent Units for ASD Classification. Frontiers in Aging Neuroscience, p. 723.

JIAO, Z., LI, H. and FAN, Y., 2020. Improving diagnosis of autism spectrum disorder and disentangling its heterogeneous functional connectivity patterns using capsule networks. In: 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI). pp. 1331–1334.

JOLLY, B.L.K. et al., 2019. Universal eeg encoder for learning diverse intelligent tasks. In: 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM). pp. 213–218.

KAMEL, M.I. et al., 2012. EEG based autism diagnosis using regularized Fisher Linear Discriminant Analysis. International Journal of Image, Graphics and Signal Processing, 4(3), p. 35.

KANG, J. et al., 2020. The identification of children with autism spectrum disorder by SVM approach on EEG and eye-tracking data. Computers in Biology and Medicine, 120, p. 103722.

KLINTWALL, L. and EIKESETH, S., 2014. Early and Intensive Behavioral Intervention (EIBI) in Autism. I VB Patel, VR Preedy & CR Martin (eds.), Comprehensive Guide to Autism (s. 117–138).

KUMAR, J.S. and BHUVANESWARI, P., 2012. Analysis of Electroencephalography (EEG) signals and its categorization–a study. Procedia Engineering, 38, pp. 2525–2536.

LAI, C.Q. et al., 2018. Artifacts and noise removal for electroencephalogram (EEG): A literature review. In: 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). pp. 326–332.

LIU, Y. et al., 2020. Multi-channel EEG-based emotion recognition via a multi-level features guided capsule network. Computers in Biology and Medicine, 123, p. 103927.

MATSON, J.L. and KOZLOWSKI, A.M., 2011. The increasing prevalence of autism spectrum disorders. Research in Autism Spectrum Disorders, 5(1), pp. 418–425.

MILLER, D., REES, J. and PEARSON, A., 2021. “Masking is life”: Experiences of masking in autistic and nonautistic adults. Autism in Adulthood, 3(4), pp. 330–338.

NOLAN, H., WHELAN, R. and REILLY, R.B., 2010. FASTER: fully automated statistical thresholding for EEG artifact rejection. Journal of Neuroscience Methods, 192(1), pp. 152–162.

OLEJNICZAK, P., 2006. Neurophysiologic basis of EEG. Journal of Clinical Neurophysiology, 23(3), pp. 186–189.

O'SHEA, K. and NASH, R., 2015. An introduction to convolutional neural networks. ArXiv Preprint ArXiv:1511.08458.

PRIETO, A. et al., 2016. Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing, 214, pp. 242–268.

REIERSEN, A.M., 2017. Early identification of autism spectrum disorder: is diagnosis by age 3 a reasonable goal? Journal of the American Academy of Child and Adolescent Psychiatry.

ROY, S., KIRAL-KORNEK, I. and HARRER, S., 2019. ChronoNet: a deep recurrent neural network for abnormal EEG identification. In: Conference on Artificial Intelligence in Medicine in Europe. pp. 47–56.

SABOUR, S., FROSST, N. and HINTON, G.E., 2017. Dynamic routing between capsules. Advances in Neural Information Processing Systems, 30.

SHI, X. et al., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems, 28.

TAWHID, M.N.A., SIULY, S. and LI, T., 2022. A Convolutional Long Short-Term Memory-Based Neural Network for Epilepsy Detection From EEG. IEEE Transactions on Instrumentation and Measurement, 71, pp. 1–11.

TEPLAN, M. and OTHERS, 2002. Fundamentals of EEG measurement. Measurement Science Review, 2(2), pp. 1–11.

THAPALIYA, S., JAYARATHNA, S. and JAIME, M., 2018. Evaluating the EEG and eye movements for autism spectrum disorder. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 2328–2336.

WU, J., 2017. Introduction to convolutional neural networks. National Key Lab for Novel Software Technology. Nanjing University. China, 5(23), p. 495.

Appendix

Appendix A – Literature Review Chapter

A.1 Neural Networks Activation Functions Table

Activation Function	**Description**	**Equation**
Sigmoid Function	The output of the logistic sigmoid function, which has a range of 0 to 1, is 1. When compared to a linear function, the output of the sigmoid activation function will always fall between (0,1) and (-inf, inf). It has a fixed output range, is monotonic, nonlinear, and continuously differentiable. It is not zero-centred, though.	sigmoid(x) = 1 / (1 + exp(-x))
Rectified Linear Unit function (ReLu)	The ReLU equation is: ReLU is the most often used activation function in neural networks. This can be identified as a nonlinear function. ReLU's output is 0 for negative input and x for positive input, respectively.	ReLU(x) = max(0,x)
Softmax Function	A vector of K real values can be transformed into a vector of K real values that totals 1 by using the softmax function. The softmax turns the input values, which can be	$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

	positive, negative, zero, or higher than one, into values between 0 and 1, allowing them to be understood as probabilities.	
Tanh Function	Another type of AF is the tanh function, also known as the hyperbolic tangent function. It has a range between -1 and 1, is smoother, and is centred on zero. The tanh function's main benefit is that it generates a zero-centred output, aiding the backpropagation procedure. Recurrent neural networks have mostly used the tanh function for tasks involving speech recognition and natural language processing.	tanh(x) = sinh(x)/cosh(x) = ((exp(x) - exp(-x))/(exp(x) + exp(-x)))

Appendix B – Requirements Analysis Chapter

B.1 Use Case Descriptions

Use case description-Add EEG data

Use case ID	UC - 01
Use Case Name	Add EEG data need to predict
Priority	High
Actors	User
Descriptions	User imports the EEG dataset that needs to predict whether it is showing ASD or not
Pre- Conditions	The application should be ready to use
Extended Use Cases	None
Included Use Cases	Detect show ASD or not using EEG data
Triggering Event	Users import the files to the system and submit
Main Flow	The system takes the EEG data and passes it to the detection mode
Alternate Flow	None
Exceptional Flow	User import non-EEG data display an error message prompt to the user
Postconditions	None

Use case description-Feed EEG data

Use case ID	UC - 02
Use Case Name	Feed data
Priority	High

Actors	Researcher
Descriptions	The researcher feeds the data that need to train the model and test the model's accuracy and other details.
Pre- Conditions	None
Extended Use Cases	None
Included Use Cases	None
Triggering Event	The researcher adds the dataset to the system environment that going to build the model.
Main Flow	The researcher adds data to the system.
Alternate Flow	None
Exceptional Flow	None
Postconditions	Data should be available in the system to train the model.

Use case description-Filter EEG data

Use case ID	UC - 06
Use Case Name	Filter data according to the bandwidth of the data
Priority	High
Actors	System
Descriptions	Filter the EEG data according to the bandwidth of the dataset
Pre- Conditions	The application should be ready to use Data should be preprocessed to pass through a bandpass filter
Extended Use Cases	Process Data

Included Use Cases	None
Triggering Event	Pre-process of data
Main Flow	When the pre-processing is done data will get filtered according to the pre-processed data bandwidth and save the pre-processed data in the system storage
Alternate Flow	None
Exceptional Flow	None
Postconditions	None

Use case description-Create frequency bands data

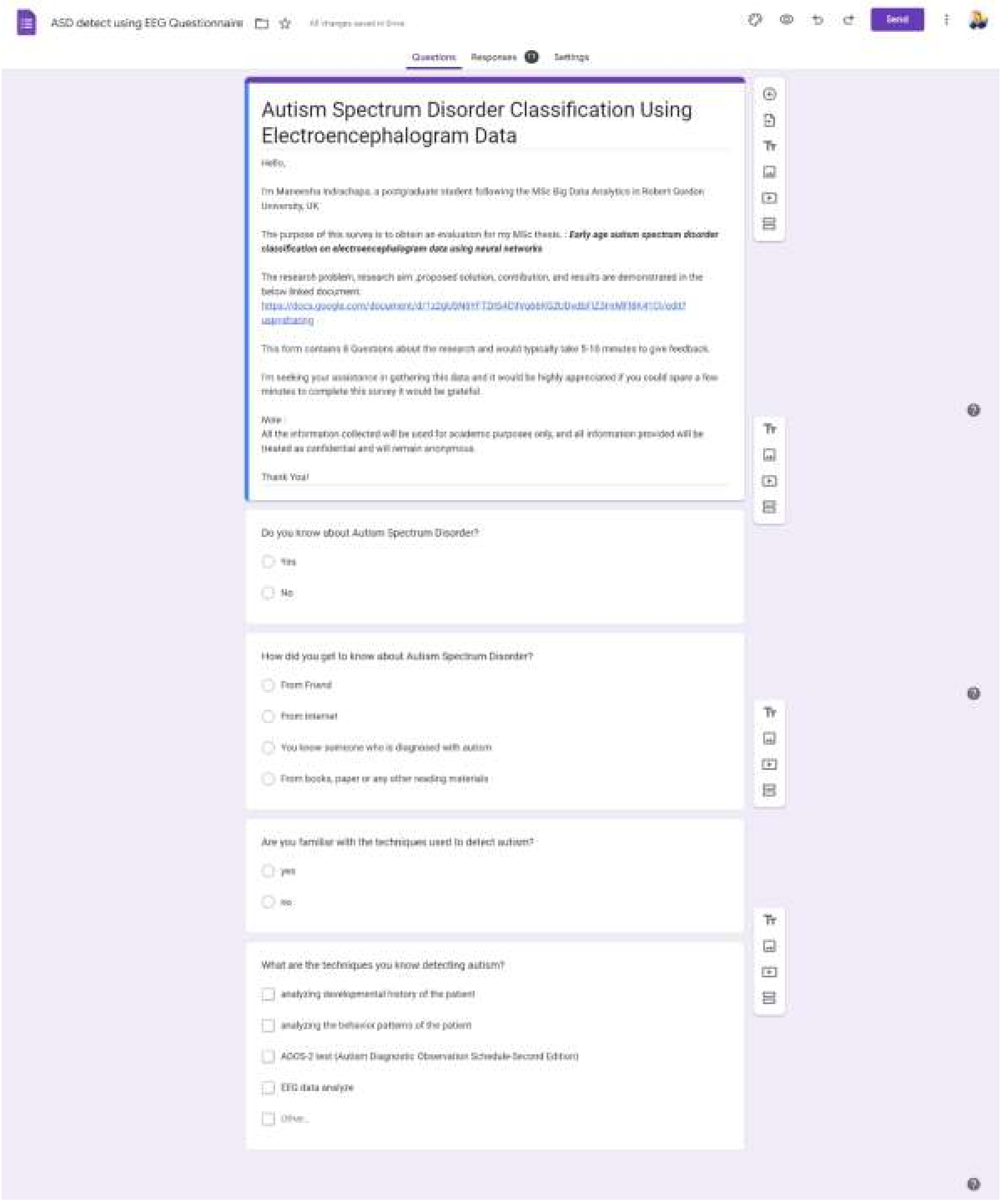
Use case ID	UC - 07
Use Case Name	Create frequency band data
Priority	High
Actors	System
Descriptions	Create the frequency bands data set from the pre-processed filtered data
Pre- Conditions	The application should be ready to use Data should be pre-processed and filtered and pass already created the pre-processed data file
Extended Use Cases	Filtered data
Included Use Cases	None
Triggering Event	Pre-process of data

Main Flow	When the filtering is done, and pre-processed data is saved in the system storage it will get and decomposition it to frequency bands and use it to generate new frequency band data set.
Alternate Flow	None
Exceptional Flow	None
Postconditions	None

Use case description-Show predicted output

Use case ID	UC - 08
Use Case Name	View Prediction
Priority	High
Actors	Researcher and User
Descriptions	Display the prediction to the added EEG data
Pre- Conditions	The application should be ready to use Non-erroneous EEG data should be submitted to the system
Extended Use Cases	None
Included Use Cases	None
Triggering Event	End of prediction done by the trained model
Main Flow	Display the prediction
Alternate Flow	None
Exceptional Flow	None
Postconditions	None

B.2 Requirements Gathering Questionnaire


ASD detect using EEG Questionnaire
All changes saved in Drive
Send
Questions
Responses
Settings

Autism Spectrum Disorder Classification Using Electroencephalogram Data

Hello,

I'm Maneesha Indrachapa, a postgraduate student following the MSc Big Data Analytics in Robert Gordon University, UK

The purpose of this survey is to obtain an evaluation for my MSc thesis. : ***Early age autism spectrum disorder classification on electroencephalogram data using neural networks***

The research problem, research aim ,proposed solution, contribution, and results are demonstrated in the below linked document.
https://docs.google.com/document/d/[illegible]/edit?usp=sharing

This form contains 8 Questions about the research and would typically take 5-10 minutes to give feedback.

I'm seeking your assistance in gathering this data and it would be highly appreciated if you could spare a few minutes to complete this survey it would be grateful.

Note :
All the information collected will be used for academic purposes only, and all information provided will be treated as confidential and will remain anonymous.

Thank You!

Do you know about Autism Spectrum Disorder?

- Yes
- No

How did you get to know about Autism Spectrum Disorder?

- From Friend
- From internet
- You know someone who is diagnosed with autism
- From books, paper or any other reading materials

Are you familiar with the techniques used to detect autism?

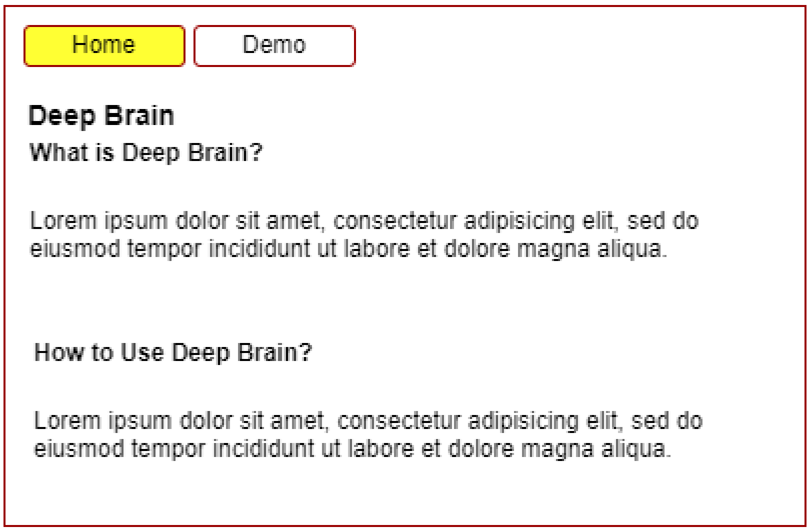
- yes
- no

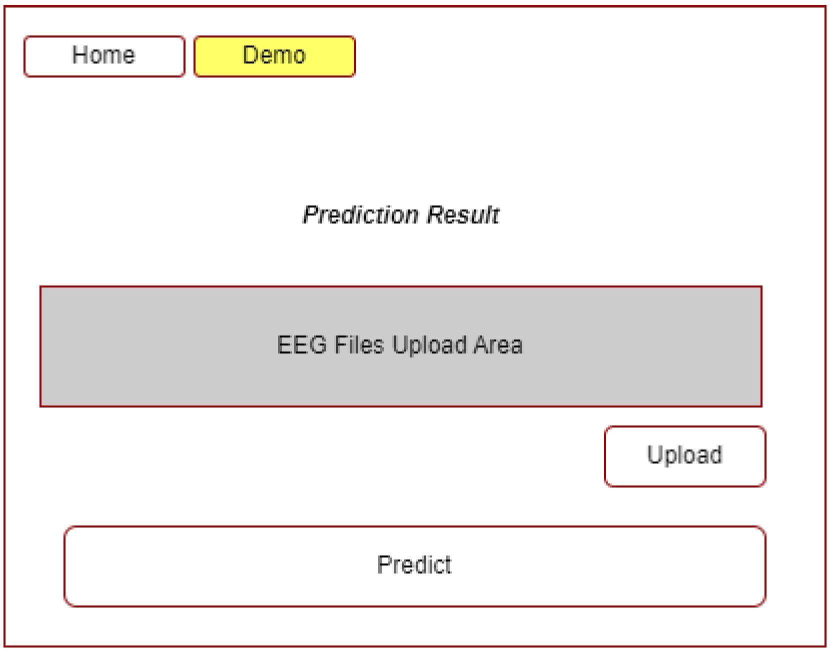
What are the techniques you know detecting autism?

- analyzing developmental history of the patient
- analyzing the behavior patterns of the patient
- ADOS-2 test (Autism Diagnostic Observation Schedule-Second Edition)
- EEG data analyze
- Other...

Appendix C – Design Chapter

C:1 UI Wire Frameworks


Home
Demo
Deep Brain
What is Deep Brain?
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
How to Use Deep Brain?
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Appendix D – Implementation Chapter

D.1 Program Language Selection

Language	Evaluation
Python	High-level object-oriented programming language Python has a considerable number of packages enabling code reuse. Additionally, it comes with a powerful and quickly expanding EEG processing package (named MNE) that may be used to visualize EEG data. Using readily available libraries, it is easy to differentiate the information from the data and clean them. Python offers support for the web frameworks Flask, Django, and Tornado. In addition, it is straightforward, simple to understand, easy to read, and requires little maintenance. If the performance of the local system is poor, Python can also be executed in Google Collab.
MATLAB	The mathematical computations performed by MATLAB while expressing and interacting with Matrix, developing algorithms, and scientific images are all possible. Its layout is user-friendly and the MATLAB language is really powerful. There is a program called EEGLAB that can be used with MATLAB for the analysis of EEG data. Apart from the pros the cons of using MATLAB are using a lot of disk space and needing a high-performance machine to run the program.
R	The R can be used to easily comprehend and explore data using statistical tools and graphs. However, because certain statistical procedures are either not yet accessible in Python or are less fully featured in Python than the R versions, R can be used for statistical analysis when dealing with neuroscience data. R tends to have the most robust and well-formed versions of particularly advanced or newer statistical approaches because it was initially designed by and for statisticians.

D.2 Libraries Selected for the Prototype

Library	version	Usage

NumPy	1.18.4	Python lists are slower than NumPy arrays, which are also smaller. The use of an array saves memory and is simple. NumPy offers a means for specifying the data types and utilizes significantly less RAM to store data.
pandas	1.0.3	One of the most popular tools for data cleaning and analysis in data science and machine learning is called Pandas. Pandas are the ideal tool for handling this chaotic real-world data in this situation. And one of the open-source Python packages constructed on top of NumPy is pandas.
pyarrow	0.17.0	Along with tools for Arrow integration and compatibility with pandas, NumPy, and other programs in the Python ecosystem, the PyArrow library offers a Python API for the features offered by the Arrow libraries.
TensorFlow	2.3.0	TensorFlow has already implemented functionalities for data automation, model training, performance monitoring, and model retraining.
mne	0.20.4	The mne library is only available to use in Python. It can use for MEG, EEG, sEEG, ECoG, NIRS, and other human neurophysiological data exploration, visualization, and analysis. Provides pipelines to clean neurophysiological data.
pywavelets	1.1.1	Pywavelets is a library that can use for wavelet transformation. When analyzing EEG data, it can use to identify the power spectrum data.
matplotlib	3.2.1	Matplotlib can use to generate graphs
Scikit-learn	0.23.0	To work with standard machine learning tasks such as classification and regressions
seaborn	0.12.1	Using this package can demonstrate improved data visualization.

D.3 Standard Electrode Placement vs Dataset Electrode Placement

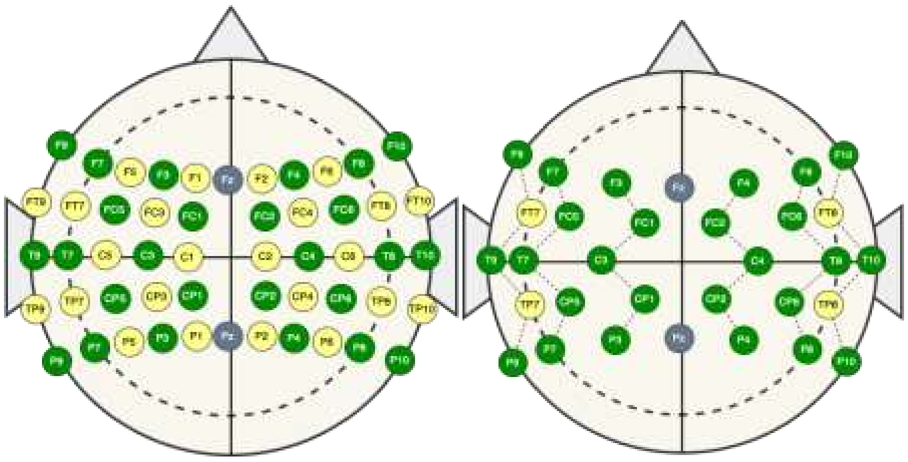
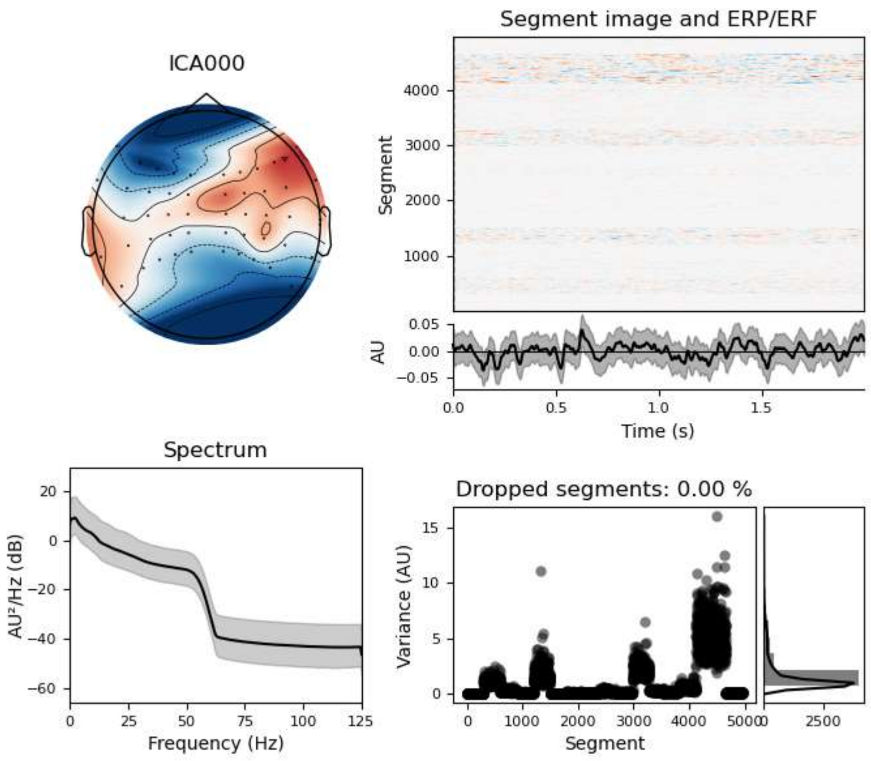


Figure A.1 - Electrode placement for EEG Capture standard10_20(left) and Data set electrode placement(right)

D.4 Independent Component Analysis ICA000 Properties



D.5 Signals Before and After ICA

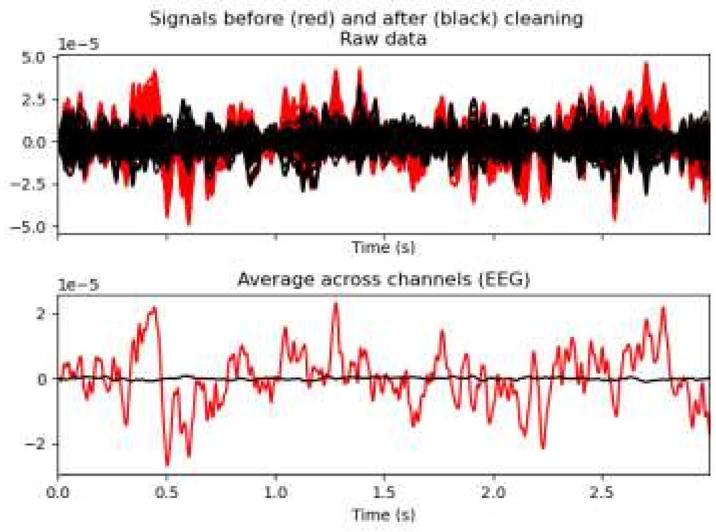


Figure A:2-ICA000 -an overlay of the original signal against the reconstructed signal

D.6 Power Spectrum Diagrams of ASD Negative and Positive Patients

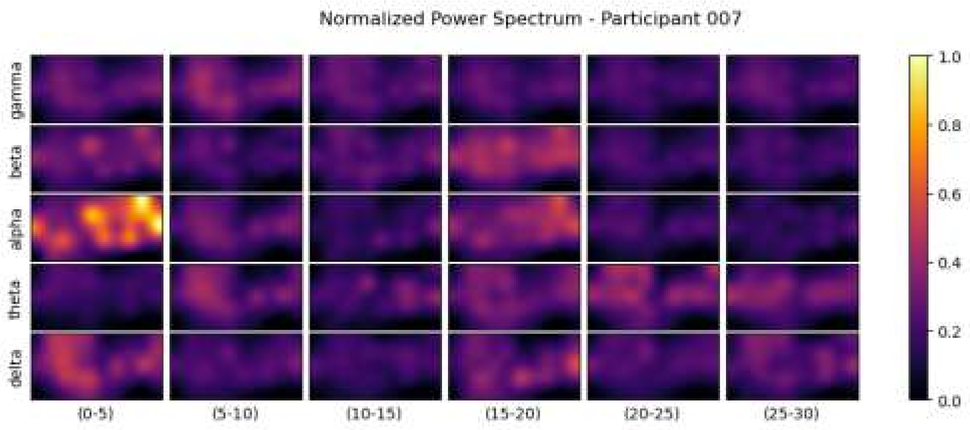


Figure A:3- Power spectrum of ASD negative participant

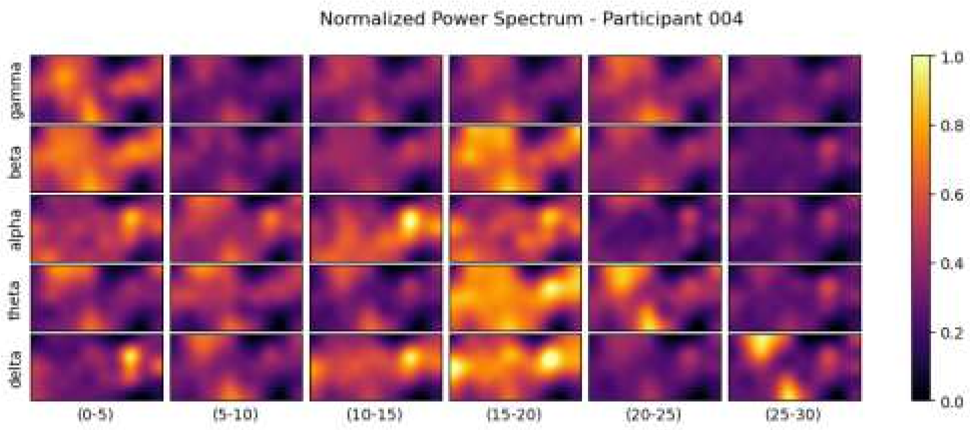
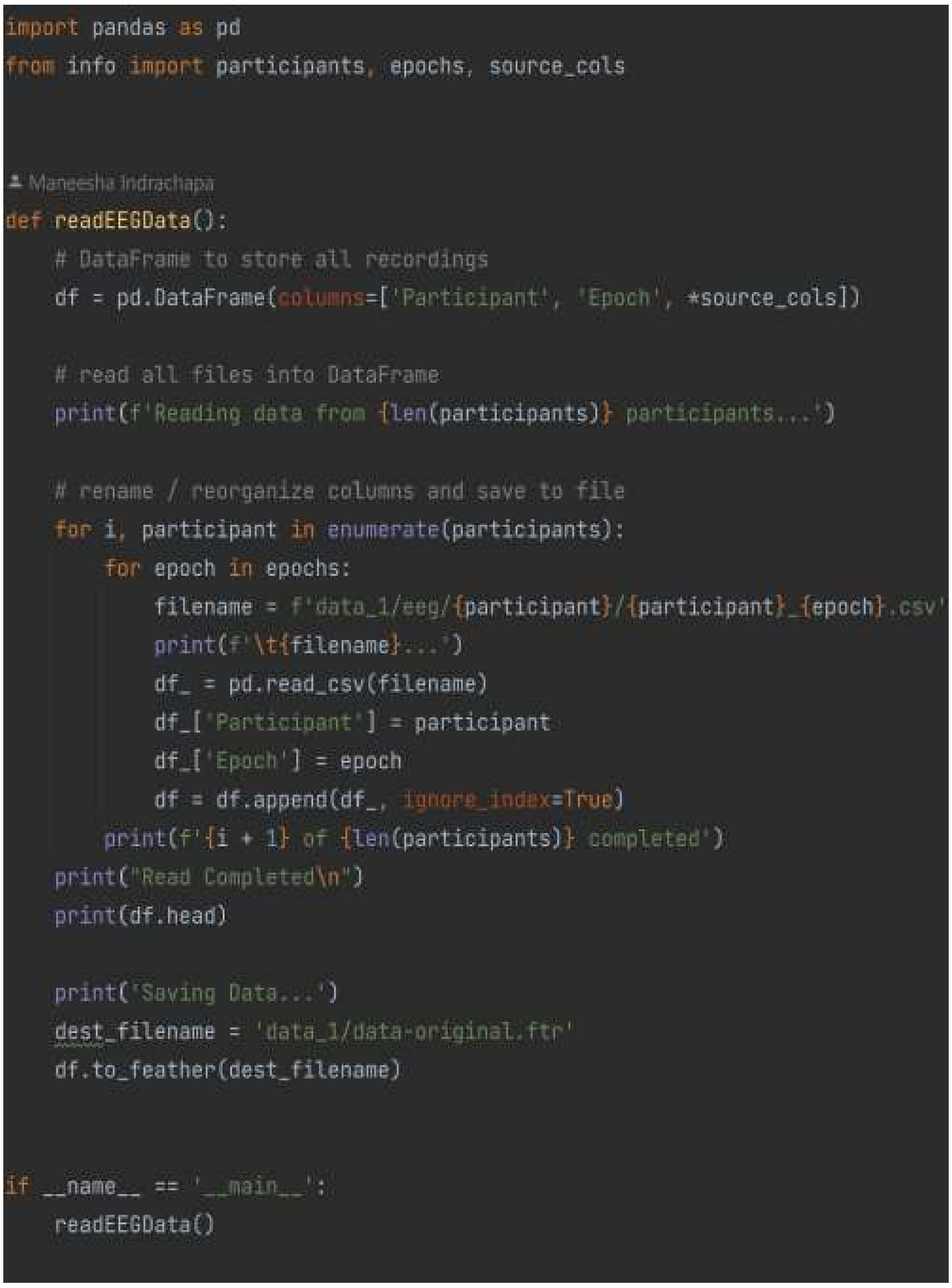


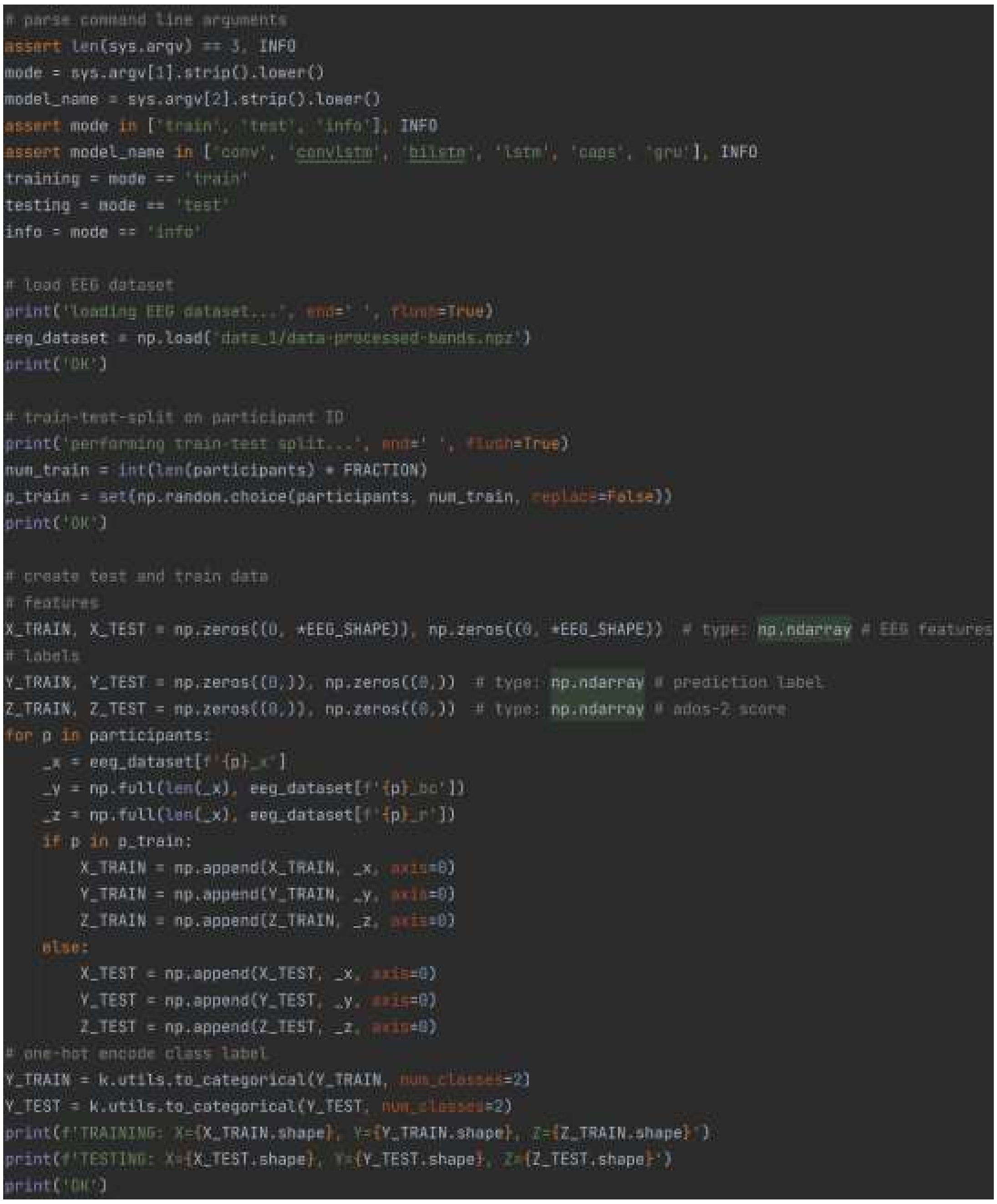
Figure A:4- Power spectrum of ASD-positive participant

D.7 Source Code for Reading all .csv File Data and Combine



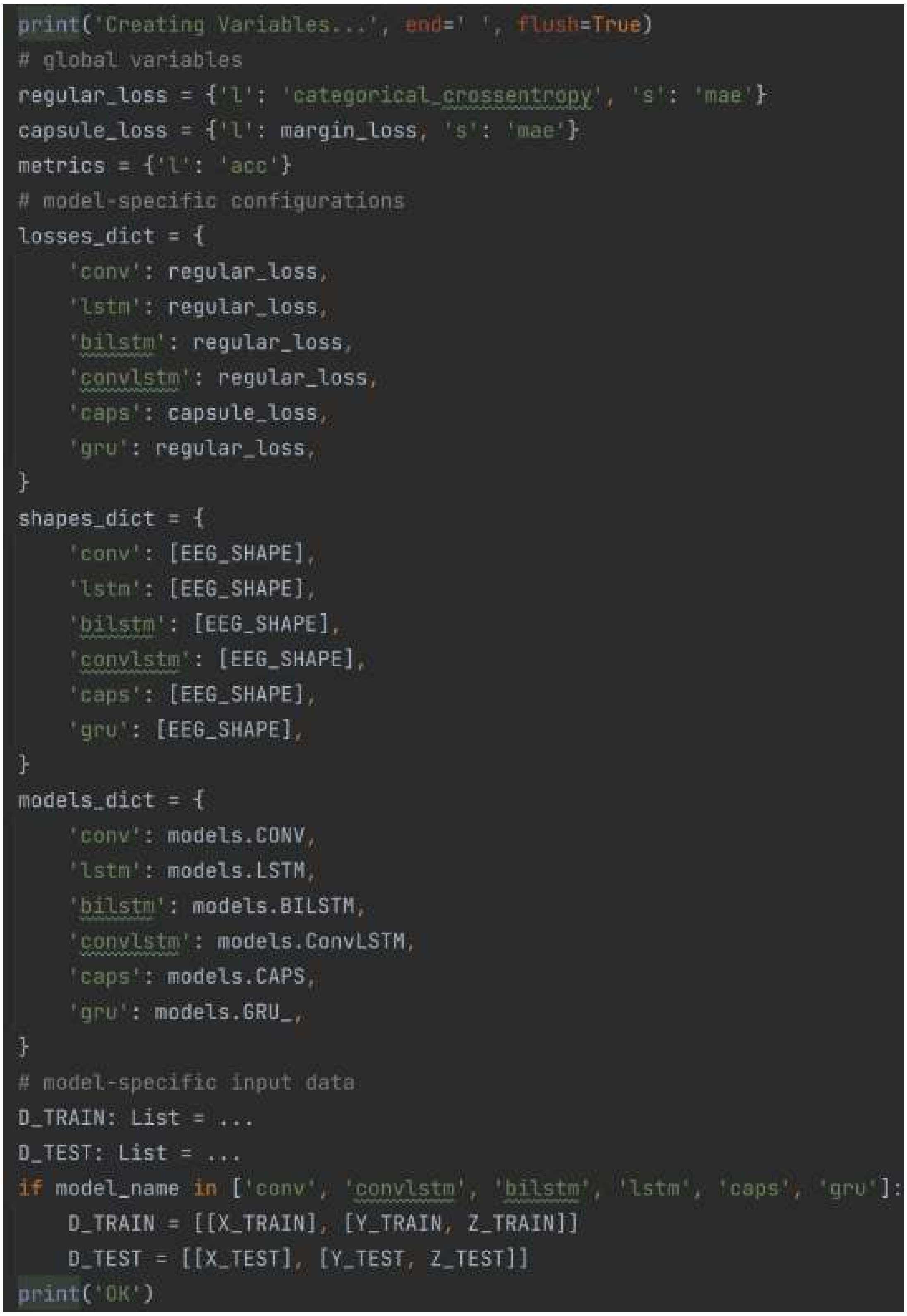
```
import pandas as pd
from info import participants, epochs, source_cols


Maneesha Indrachapa
def readEEGData():
    # DataFrame to store all recordings
    df = pd.DataFrame(columns=['Participant', 'Epoch', *source_cols])

    # read all files into DataFrame
    print(f'Reading data from {len(participants)} participants...')

    # rename / reorganize columns and save to file
    for i, participant in enumerate(participants):
        for epoch in epochs:
            filename = f'data_1/eeg/{participant}/{participant}_{epoch}.csv'
            print(f'\t{filename}...')
            df_ = pd.read_csv(filename)
            df_['Participant'] = participant
            df_['Epoch'] = epoch
            df = df.append(df_, ignore_index=True)
        print(f'{i + 1} of {len(participants)} completed')
    print("Read Completed\n")
    print(df.head)

    print('Saving Data...')
    dest_filename = 'data_1/data-original.ftr'
    df.to_feather(dest_filename)


if __name__ == '__main__':
    readEEGData()
```

D.8 Train Test Data Split Code Snippet



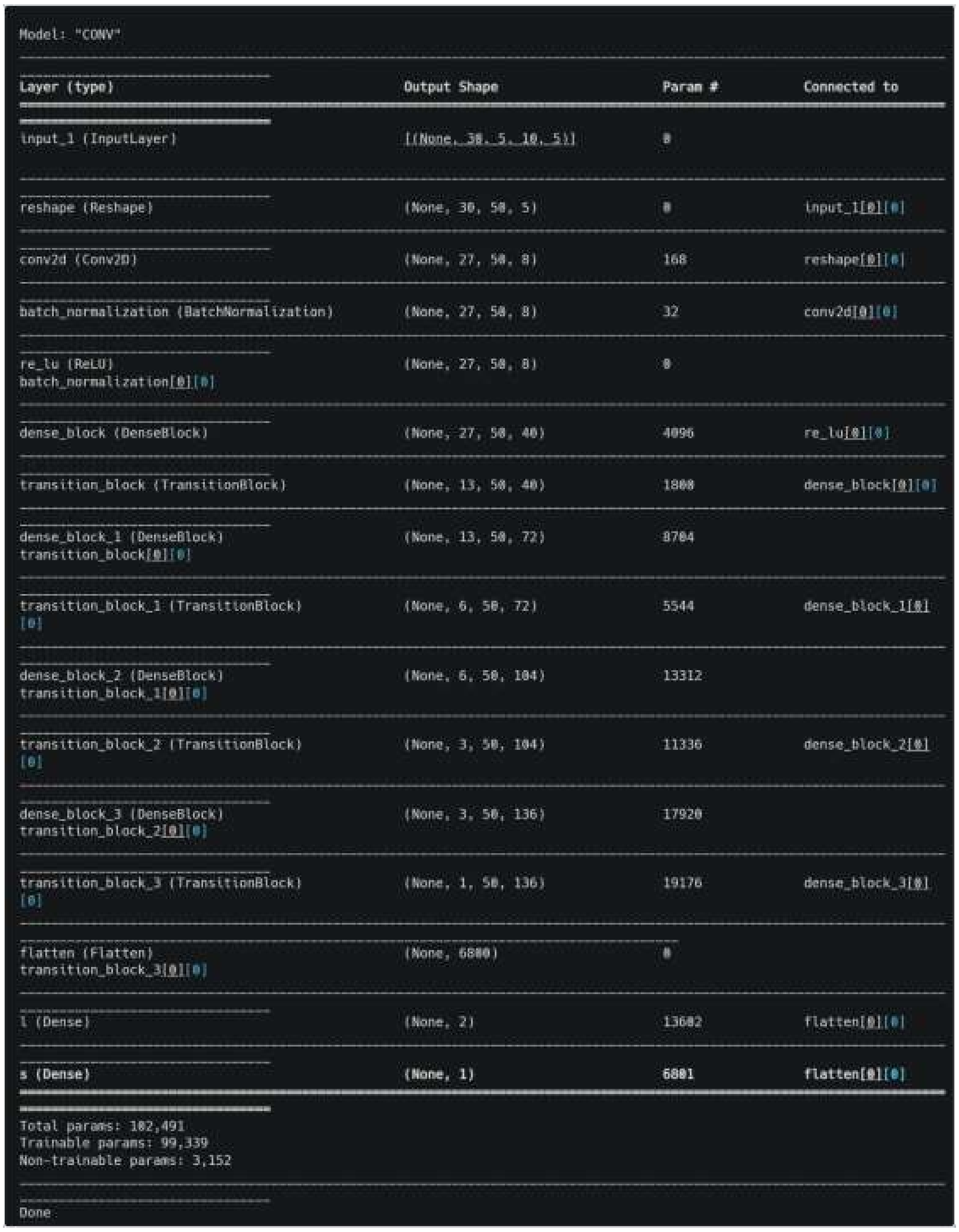
```python
# parse command line arguments
assert len(sys.argv) == 3, INFO
mode = sys.argv[1].strip().lower()
model_name = sys.argv[2].strip().lower()
assert mode in ['train', 'test', 'info'], INFO
assert model_name in ['conv', 'convlstm', 'bilstm', 'lstm', 'caps', 'gru'], INFO
training = mode == 'train'
testing = mode == 'test'
info = mode == 'info'

# load EEG dataset
print('loading EEG dataset...', end=' ', flush=True)
eeg_dataset = np.load('data_1/data-processed-bands.npz')
print('OK')

# train-test-split on participant ID
print('performing train-test split...', end=' ', flush=True)
num_train = int(len(participants) * FRACTION)
p_train = set(np.random.choice(participants, num_train, replace=False))
print('OK')

# create test and train data
# features
X_TRAIN, X_TEST = np.zeros((0, *EEG_SHAPE)), np.zeros((0, *EEG_SHAPE))  # type: np.ndarray # EEG features
# labels
Y_TRAIN, Y_TEST = np.zeros((0,)), np.zeros((0,))  # type: np.ndarray # prediction label
Z_TRAIN, Z_TEST = np.zeros((0,)), np.zeros((0,))  # type: np.ndarray # ados-2 score
for p in participants:
    _x = eeg_dataset[f'{p}_x']
    _y = np.full(len(_x), eeg_dataset[f'{p}_bc'])
    _z = np.full(len(_x), eeg_dataset[f'{p}_r'])
    if p in p_train:
        X_TRAIN = np.append(X_TRAIN, _x, axis=0)
        Y_TRAIN = np.append(Y_TRAIN, _y, axis=0)
        Z_TRAIN = np.append(Z_TRAIN, _z, axis=0)
    else:
        X_TEST = np.append(X_TEST, _x, axis=0)
        Y_TEST = np.append(Y_TEST, _y, axis=0)
        Z_TEST = np.append(Z_TEST, _z, axis=0)
# one-hot encode class label
Y_TRAIN = k.utils.to_categorical(Y_TRAIN, num_classes=2)
Y_TEST = k.utils.to_categorical(Y_TEST, num_classes=2)
print(f'TRAINING: X={X_TRAIN.shape}, Y={Y_TRAIN.shape}, Z={Z_TRAIN.shape}')
print(f'TESTING: X={X_TEST.shape}, Y={Y_TEST.shape}, Z={Z_TEST.shape}')
print('OK')
```

D.9 Model-Specific Global Configuration Code Snippet



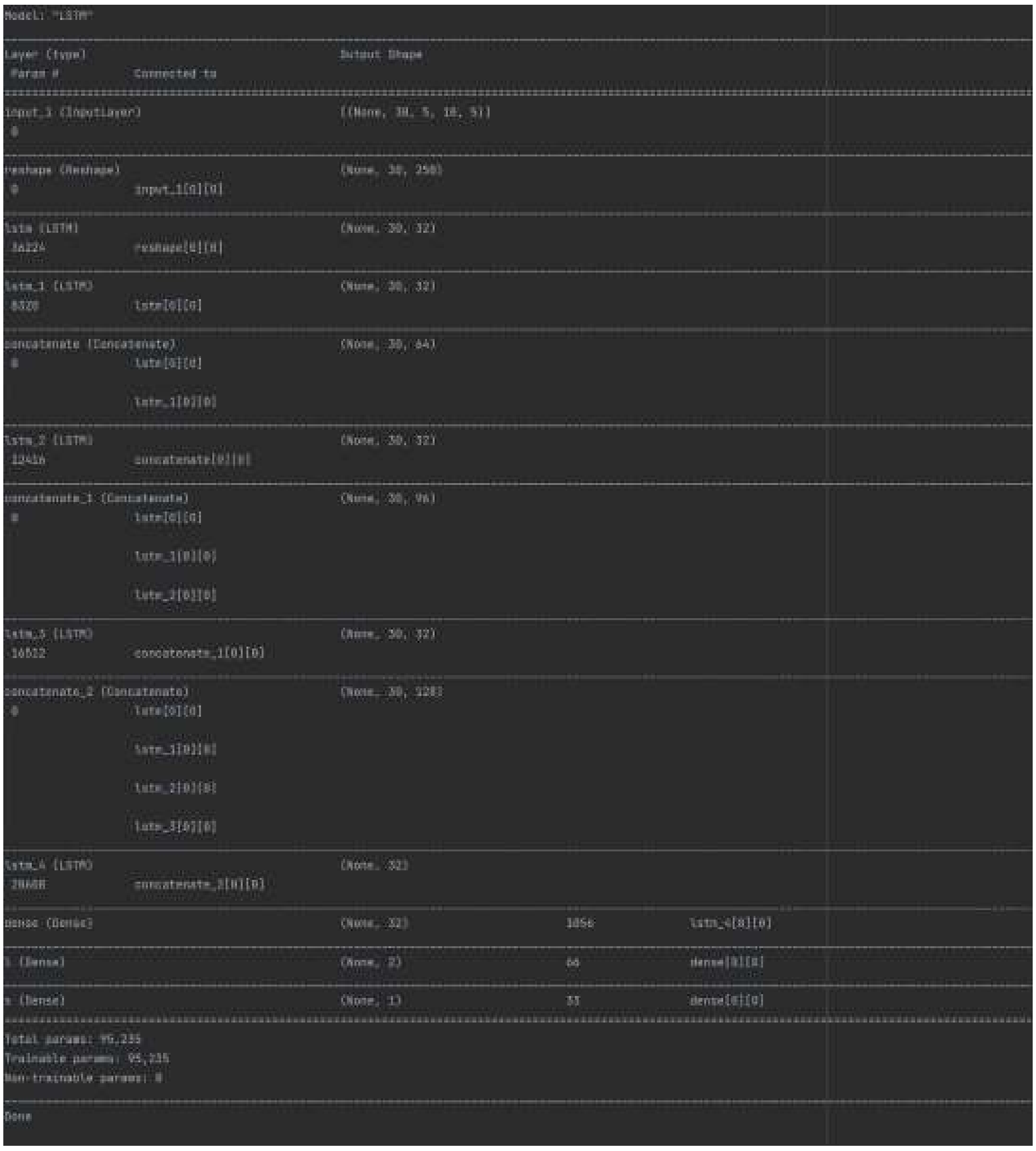
```python
print('Creating Variables...', end=' ', flush=True)
# global variables
regular_loss = {'l': 'categorical_crossentropy', 's': 'mae'}
capsule_loss = {'l': margin_loss, 's': 'mae'}
metrics = {'l': 'acc'}
# model-specific configurations
losses_dict = {
    'conv': regular_loss,
    'lstm': regular_loss,
    'bilstm': regular_loss,
    'convlstm': regular_loss,
    'caps': capsule_loss,
    'gru': regular_loss,
}
shapes_dict = {
    'conv': [EEG_SHAPE],
    'lstm': [EEG_SHAPE],
    'bilstm': [EEG_SHAPE],
    'convlstm': [EEG_SHAPE],
    'caps': [EEG_SHAPE],
    'gru': [EEG_SHAPE],
}
models_dict = {
    'conv': models.CONV,
    'lstm': models.LSTM,
    'bilstm': models.BILSTM,
    'convlstm': models.ConvLSTM,
    'caps': models.CAPS,
    'gru': models.GRU_,
}
# model-specific input data
D_TRAIN: List = ...
D_TEST: List = ...
if model_name in ['conv', 'convlstm', 'bilstm', 'lstm', 'caps', 'gru']:
    D_TRAIN = [[X_TRAIN], [Y_TRAIN, Z_TRAIN]]
    D_TEST = [[X_TEST], [Y_TEST, Z_TEST]]
print('OK')
```

D.10 Convolutional Model Summary



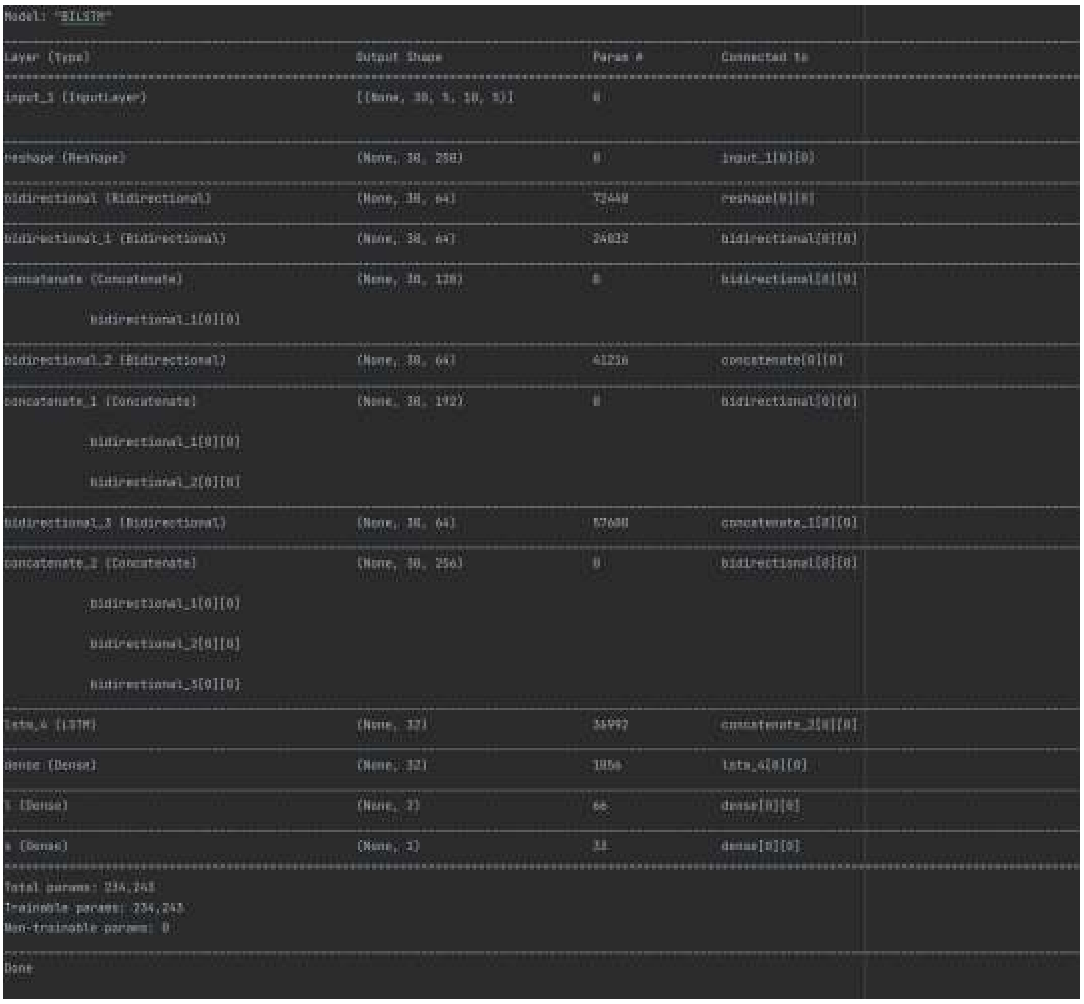
```
Model: "CONV"
__________________________________________________________________________________________________
Layer (type)                             Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)                     [(None, 30, 5, 10, 5)]    0

__________________________________________________________________________________________________
reshape (Reshape)                        (None, 30, 50, 5)         0           input_1[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                          (None, 27, 50, 8)         168         reshape[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNormalization) (None, 27, 50, 8)         32          conv2d[0][0]
__________________________________________________________________________________________________
re_lu (ReLU)                             (None, 27, 50, 8)         0
batch_normalization[0][0]
__________________________________________________________________________________________________
dense_block (DenseBlock)                 (None, 27, 50, 40)        4096        re_lu[0][0]
__________________________________________________________________________________________________
transition_block (TransitionBlock)       (None, 13, 50, 40)        1800        dense_block[0][0]
__________________________________________________________________________________________________
dense_block_1 (DenseBlock)               (None, 13, 50, 72)        8704
transition_block[0][0]
__________________________________________________________________________________________________
transition_block_1 (TransitionBlock)     (None, 6, 50, 72)         5544        dense_block_1[0]
[0]
__________________________________________________________________________________________________
dense_block_2 (DenseBlock)               (None, 6, 50, 104)        13312
transition_block_1[0][0]
__________________________________________________________________________________________________
transition_block_2 (TransitionBlock)     (None, 3, 50, 104)        11336       dense_block_2[0]
[0]
__________________________________________________________________________________________________
dense_block_3 (DenseBlock)               (None, 3, 50, 136)        17920
transition_block_2[0][0]
__________________________________________________________________________________________________
transition_block_3 (TransitionBlock)     (None, 1, 50, 136)        19176       dense_block_3[0]
[0]
__________________________________________________________________________________________________
flatten (Flatten)                        (None, 6800)              0
transition_block_3[0][0]
__________________________________________________________________________________________________
l (Dense)                                (None, 2)                 13602       flatten[0][0]
__________________________________________________________________________________________________
s (Dense)                                (None, 1)                 6801        flatten[0][0]
==================================================================================================
Total params: 102,491
Trainable params: 99,339
Non-trainable params: 3,152
__________________________________________________________________________________________________
Done
```

D.11 LSTM Model Information



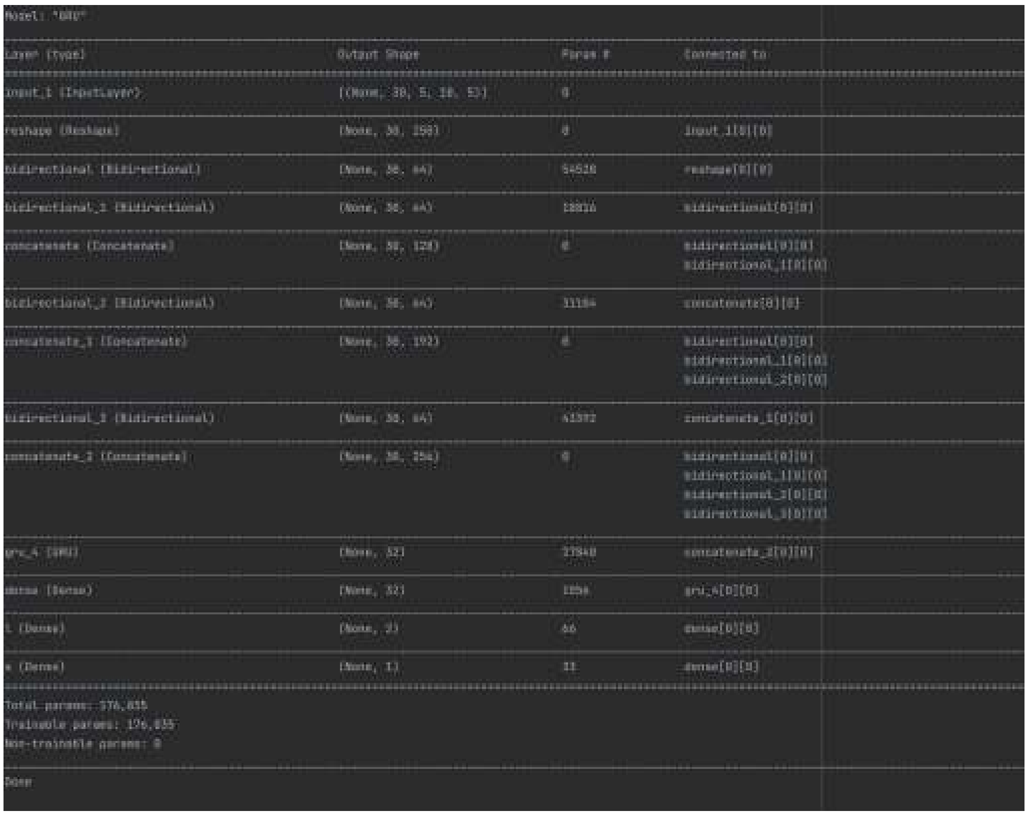
```
Model: "LSTM"
________________________________________________________________________________________________________________
Layer (type)                                    Output Shape
 Param #          Connected to
================================================================================================================
input_1 (InputLayer)                            [(None, 30, 5, 10, 5)]
 0
________________________________________________________________________________________________________________
reshape (Reshape)                               (None, 30, 250)
 0                input_1[0][0]
________________________________________________________________________________________________________________
lstm (LSTM)                                     (None, 30, 32)
 36224            reshape[0][0]
________________________________________________________________________________________________________________
lstm_1 (LSTM)                                   (None, 30, 32)
 8320             lstm[0][0]
________________________________________________________________________________________________________________
concatenate (Concatenate)                       (None, 30, 64)
 0                lstm[0][0]

                  lstm_1[0][0]
________________________________________________________________________________________________________________
lstm_2 (LSTM)                                   (None, 30, 32)
 12416            concatenate[0][0]
________________________________________________________________________________________________________________
concatenate_1 (Concatenate)                     (None, 30, 96)
 0                lstm[0][0]

                  lstm_1[0][0]

                  lstm_2[0][0]
________________________________________________________________________________________________________________
lstm_3 (LSTM)                                   (None, 30, 32)
 16512            concatenate_1[0][0]
________________________________________________________________________________________________________________
concatenate_2 (Concatenate)                     (None, 30, 128)
 0                lstm[0][0]

                  lstm_1[0][0]

                  lstm_2[0][0]

                  lstm_3[0][0]
________________________________________________________________________________________________________________
lstm_4 (LSTM)                                   (None, 32)
 20608            concatenate_2[0][0]
________________________________________________________________________________________________________________
dense (Dense)                                   (None, 32)                1056            lstm_4[0][0]
________________________________________________________________________________________________________________
1 (Dense)                                       (None, 2)                 66              dense[0][0]
________________________________________________________________________________________________________________
n (Dense)                                       (None, 1)                 33              dense[0][0]
================================================================================================================
Total params: 95,235
Trainable params: 95,235
Non-trainable params: 0
________________________________________________________________________________________________________________
Done
```

D.12 Bi- LSTM Model Information



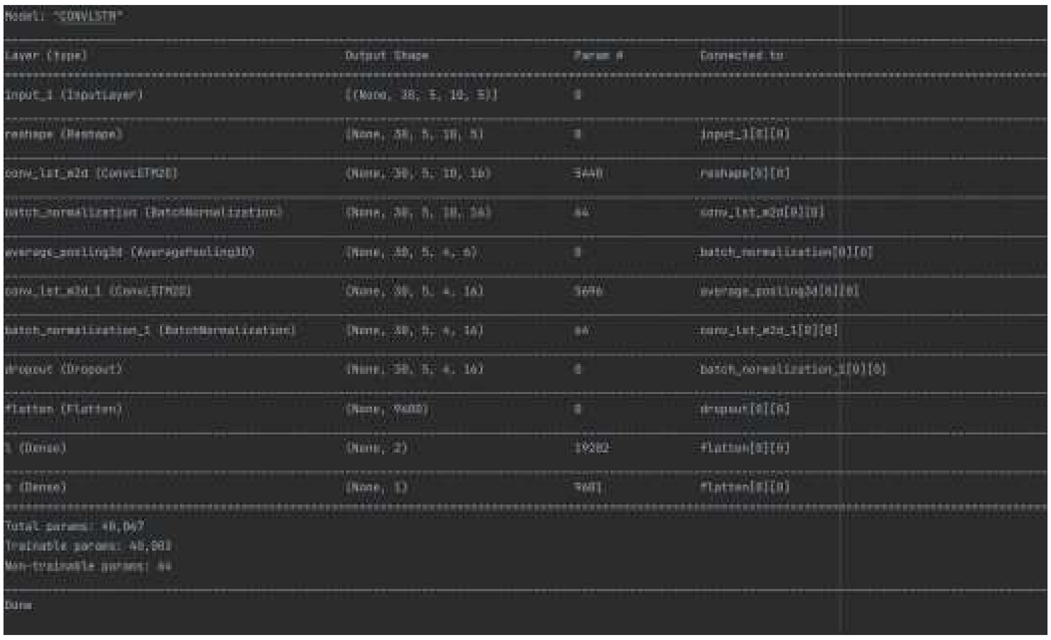
```
Model: "BILSTM"
____________________________________________________________________________________________________
Layer (Type)                      Output Shape               Param #       Connected to
====================================================================================================
input_1 (InputLayer)              [(None, 30, 5, 10, 5)]     0

____________________________________________________________________________________________________
reshape (Reshape)                 (None, 30, 250)            0             input_1[0][0]
____________________________________________________________________________________________________
bidirectional (Bidirectional)     (None, 30, 64)             72448         reshape[0][0]
____________________________________________________________________________________________________
bidirectional_1 (Bidirectional)   (None, 30, 64)             24832         bidirectional[0][0]
____________________________________________________________________________________________________
concatenate (Concatenate)         (None, 30, 128)            0             bidirectional[0][0]

            bidirectional_1[0][0]
____________________________________________________________________________________________________
bidirectional_2 (Bidirectional)   (None, 30, 64)             41216         concatenate[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)       (None, 30, 192)            0             bidirectional[0][0]

            bidirectional_1[0][0]

            bidirectional_2[0][0]
____________________________________________________________________________________________________
bidirectional_3 (Bidirectional)   (None, 30, 64)             57600         concatenate_1[0][0]
____________________________________________________________________________________________________
concatenate_2 (Concatenate)       (None, 30, 256)            0             bidirectional[0][0]

            bidirectional_1[0][0]

            bidirectional_2[0][0]

            bidirectional_3[0][0]
____________________________________________________________________________________________________
lstm_4 (LSTM)                     (None, 32)                 36992         concatenate_2[0][0]
____________________________________________________________________________________________________
dense (Dense)                     (None, 32)                 1056          lstm_4[0][0]
____________________________________________________________________________________________________
t (Dense)                         (None, 2)                  66            dense[0][0]
____________________________________________________________________________________________________
s (Dense)                         (None, 1)                  33            dense[0][0]
====================================================================================================
Total params: 234,243
Trainable params: 234,243
Non-trainable params: 0
____________________________________________________________________________________________________
Done
```

D.13 GRU Model Information



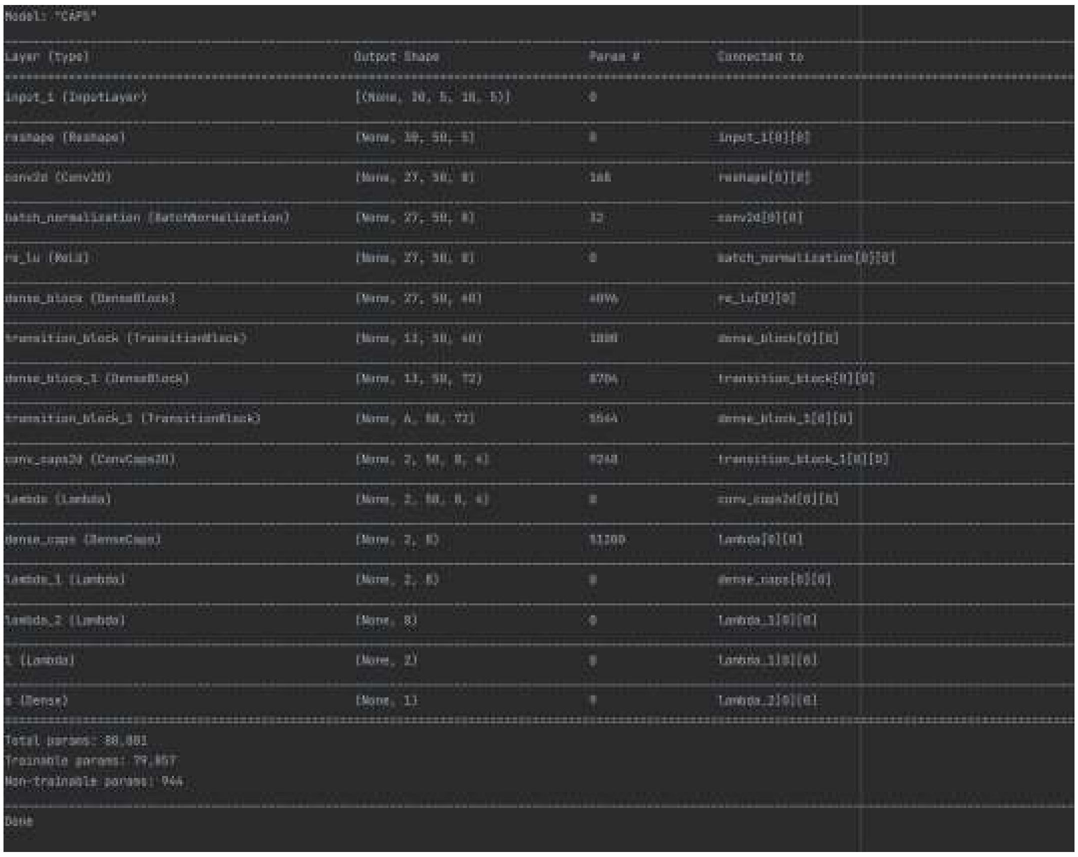
```
Model: "GRU"
________________________________________________________________________________________________
Layer (type)                          Output Shape               Param #     Connected to
================================================================================================
input_1 (InputLayer)                  [(None, 30, 5, 10, 5)]     0
________________________________________________________________________________________________
reshape (Reshape)                     (None, 30, 250)            0           input_1[0][0]
________________________________________________________________________________________________
bidirectional (Bidirectional)         (None, 30, 64)             54528       reshape[0][0]
________________________________________________________________________________________________
bidirectional_1 (Bidirectional)       (None, 30, 64)             18816       bidirectional[0][0]
________________________________________________________________________________________________
concatenate (Concatenate)             (None, 30, 128)            0           bidirectional[0][0]
                                                                             bidirectional_1[0][0]
________________________________________________________________________________________________
bidirectional_2 (Bidirectional)       (None, 30, 64)             31104       concatenate[0][0]
________________________________________________________________________________________________
concatenate_1 (Concatenate)           (None, 30, 192)            0           bidirectional[0][0]
                                                                             bidirectional_1[0][0]
                                                                             bidirectional_2[0][0]
________________________________________________________________________________________________
bidirectional_3 (Bidirectional)       (None, 30, 64)             43392       concatenate_1[0][0]
________________________________________________________________________________________________
concatenate_2 (Concatenate)           (None, 30, 256)            0           bidirectional[0][0]
                                                                             bidirectional_1[0][0]
                                                                             bidirectional_2[0][0]
                                                                             bidirectional_3[0][0]
________________________________________________________________________________________________
gru_4 (GRU)                           (None, 32)                 27840       concatenate_2[0][0]
________________________________________________________________________________________________
dense (Dense)                         (None, 32)                 1056        gru_4[0][0]
________________________________________________________________________________________________
l (Dense)                             (None, 2)                  66          dense[0][0]
________________________________________________________________________________________________
s (Dense)                             (None, 1)                  33          dense[0][0]
================================================================================================
Total params: 176,835
Trainable params: 176,835
Non-trainable params: 0
________________________________________________________________________________________________
Done
```

D.14 ConvLSTM Model Information



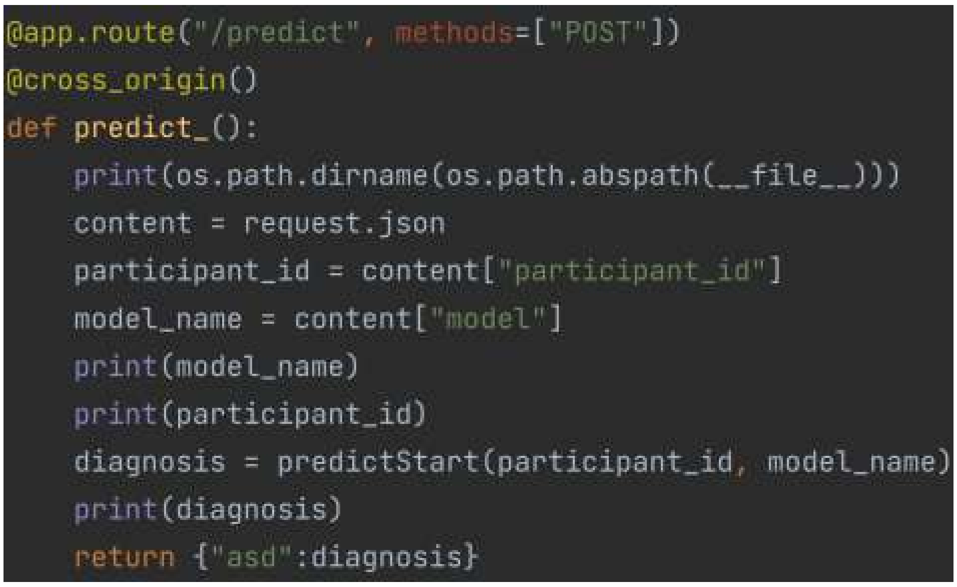
```
Model: "CONVLSTM"
________________________________________________________________________________________________
Layer (type)                                 Output Shape              Param #    Connected to
================================================================================================
input_1 (InputLayer)                         [(None, 30, 5, 10, 5)]    0
________________________________________________________________________________________________
reshape (Reshape)                            (None, 30, 5, 10, 5)      0          input_1[0][0]
________________________________________________________________________________________________
conv_lst_m2d (ConvLSTM2D)                    (None, 30, 5, 10, 16)     3440       reshape[0][0]
________________________________________________________________________________________________
batch_normalization (BatchNormalization)     (None, 30, 5, 10, 16)     64         conv_lst_m2d[0][0]
________________________________________________________________________________________________
average_pooling3d (AveragePooling3D)         (None, 30, 5, 4, 6)       0          batch_normalization[0][0]
________________________________________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)                  (None, 30, 5, 4, 16)      5696       average_pooling3d[0][0]
________________________________________________________________________________________________
batch_normalization_1 (BatchNormalization)   (None, 30, 5, 4, 16)      64         conv_lst_m2d_1[0][0]
________________________________________________________________________________________________
dropout (Dropout)                            (None, 30, 5, 4, 16)      0          batch_normalization_1[0][0]
________________________________________________________________________________________________
flatten (Flatten)                            (None, 9600)              0          dropout[0][0]
________________________________________________________________________________________________
l (Dense)                                    (None, 2)                 19202      flatten[0][0]
________________________________________________________________________________________________
s (Dense)                                    (None, 1)                 9601       flatten[0][0]
================================================================================================
Total params: 38,067
Trainable params: 38,003
Non-trainable params: 64
________________________________________________________________________________________________
Done
```

D.15 CapsNet Model Information



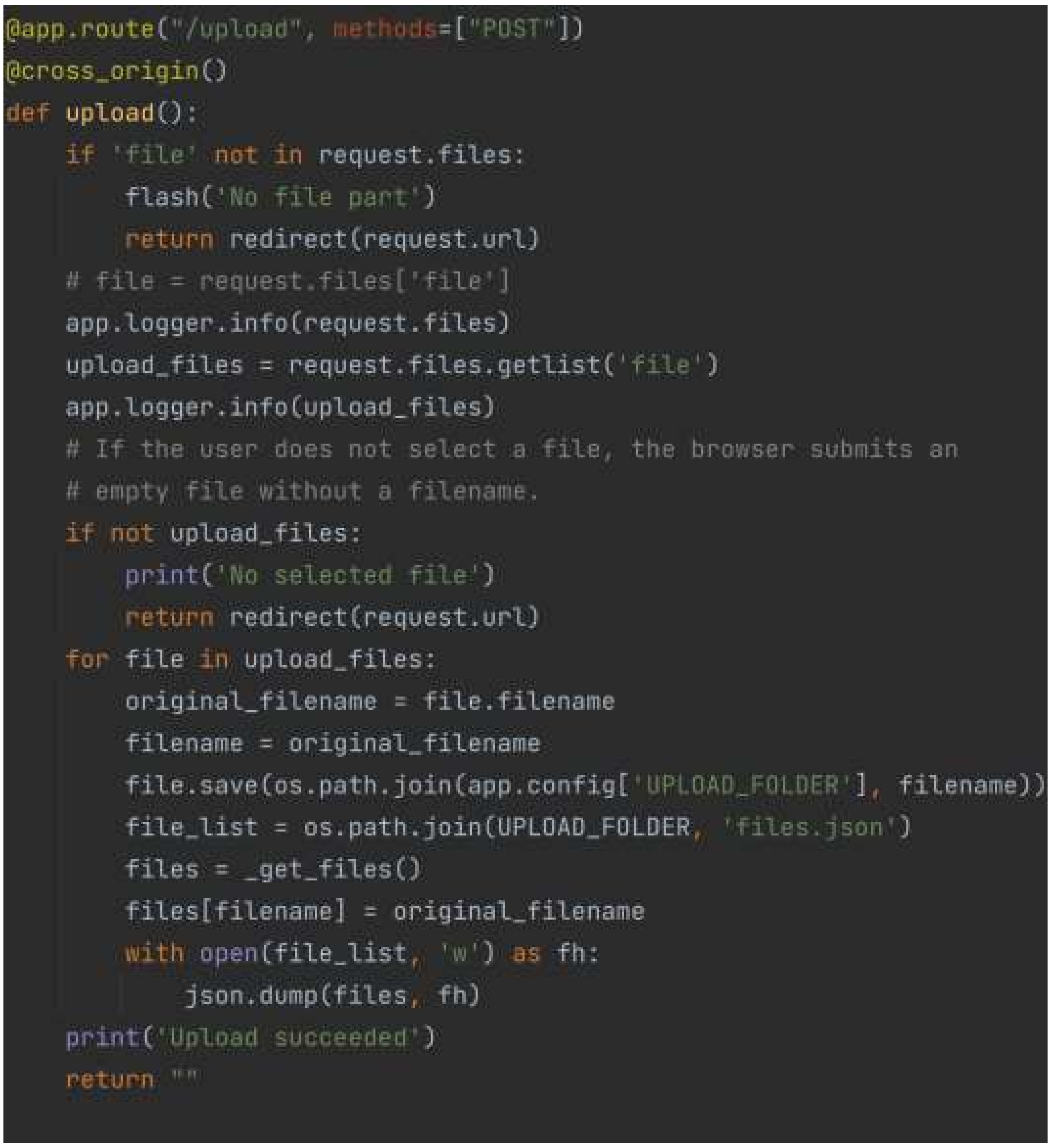
```
Model: "CAPS"
__________________________________________________________________________________________________
Layer (type)                              Output Shape            Param #     Connected to
==================================================================================================
input_1 (InputLayer)                      [(None, 30, 5, 10, 5)]  0
__________________________________________________________________________________________________
reshape (Reshape)                         (None, 30, 50, 5)       0           input_1[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                           (None, 27, 50, 8)       168         reshape[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNormalization)  (None, 27, 50, 8)       32          conv2d[0][0]
__________________________________________________________________________________________________
re_lu (ReLU)                              (None, 27, 50, 8)       0           batch_normalization[0][0]
__________________________________________________________________________________________________
dense_block (DenseBlock)                  (None, 27, 50, 40)      4096        re_lu[0][0]
__________________________________________________________________________________________________
transition_block (TransitionBlock)        (None, 13, 50, 40)      1880        dense_block[0][0]
__________________________________________________________________________________________________
dense_block_1 (DenseBlock)                (None, 13, 50, 72)      8704        transition_block[0][0]
__________________________________________________________________________________________________
transition_block_1 (TransitionBlock)      (None, 6, 50, 72)       5564        dense_block_1[0][0]
__________________________________________________________________________________________________
conv_caps2d (ConvCaps2D)                  (None, 2, 50, 8, 4)     9248        transition_block_1[0][0]
__________________________________________________________________________________________________
lambda (Lambda)                           (None, 2, 50, 8, 4)     0           conv_caps2d[0][0]
__________________________________________________________________________________________________
dense_caps (DenseCaps)                    (None, 2, 8)            51200       lambda[0][0]
__________________________________________________________________________________________________
lambda_1 (Lambda)                         (None, 2, 8)            0           dense_caps[0][0]
__________________________________________________________________________________________________
lambda_2 (Lambda)                         (None, 8)               0           lambda_1[0][0]
__________________________________________________________________________________________________
l (Lambda)                                (None, 2)               0           lambda_1[0][0]
__________________________________________________________________________________________________
s (Dense)                                 (None, 1)               9           lambda_2[0][0]
==================================================================================================
Total params: 88,801
Trainable params: 79,857
Non-trainable params: 944
__________________________________________________________________________________________________
Done
```

D.16 Flask Backend API Endpoint for Predictions



```python
@app.route("/predict", methods=["POST"])
@cross_origin()
def predict_():
    print(os.path.dirname(os.path.abspath(__file__)))
    content = request.json
    participant_id = content["participant_id"]
    model_name = content["model"]
    print(model_name)
    print(participant_id)
    diagnosis = predictStart(participant_id, model_name)
    print(diagnosis)
    return {"asd":diagnosis}
```

D.17 Flask API Endpoint for File Upload

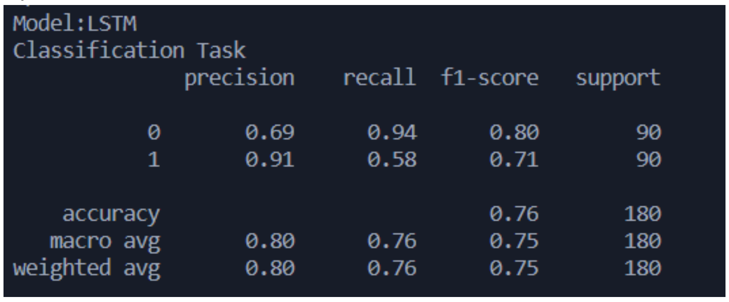


```
@app.route("/upload", methods=["POST"])
@cross_origin()
def upload():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(request.url)
    # file = request.files['file']
    app.logger.info(request.files)
    upload_files = request.files.getlist('file')
    app.logger.info(upload_files)
    # If the user does not select a file, the browser submits an
    # empty file without a filename.
    if not upload_files:
        print('No selected file')
        return redirect(request.url)
    for file in upload_files:
        original_filename = file.filename
        filename = original_filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        file_list = os.path.join(UPLOAD_FOLDER, 'files.json')
        files = _get_files()
        files[filename] = original_filename
        with open(file_list, 'w') as fh:
            json.dump(files, fh)
    print('Upload succeeded')
    return ""
```

Appendix E – Testing Chapter

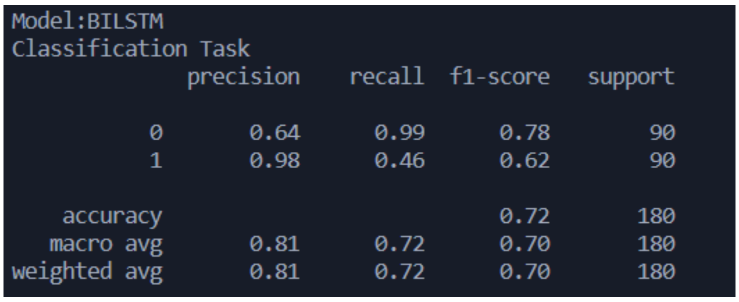
E.1: Trained Model Accuracies

LSTM model accuracy



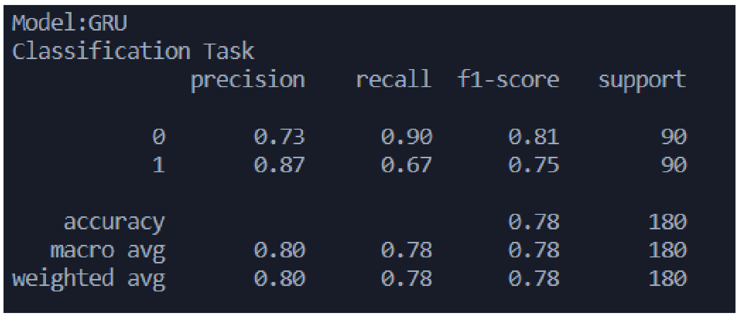
```
Model:LSTM
Classification Task
              precision    recall  f1-score   support

           0       0.69      0.94      0.80        90
           1       0.91      0.58      0.71        90

    accuracy                           0.76       180
   macro avg       0.80      0.76      0.75       180
weighted avg       0.80      0.76      0.75       180
```

Bi-LSTM model accuracy



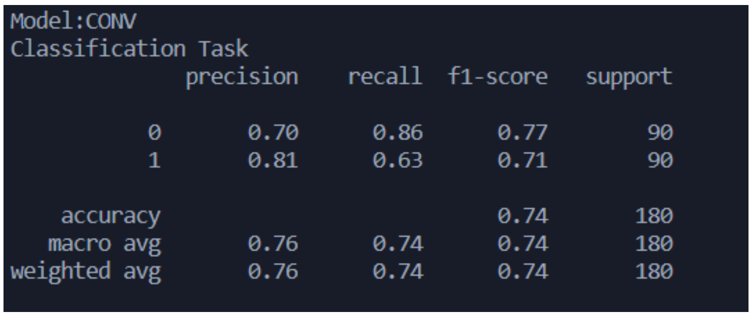
```
Model:BILSTM
Classification Task
              precision    recall  f1-score   support

           0       0.64      0.99      0.78        90
           1       0.98      0.46      0.62        90

    accuracy                           0.72       180
   macro avg       0.81      0.72      0.70       180
weighted avg       0.81      0.72      0.70       180
```

GRU model accuracy



```
Model:GRU
Classification Task
              precision    recall  f1-score   support

           0       0.73      0.90      0.81        90
           1       0.87      0.67      0.75        90

    accuracy                           0.78       180
   macro avg       0.80      0.78      0.78       180
weighted avg       0.80      0.78      0.78       180
```

CONV model accuracy



```
Model:CONV
Classification Task
              precision    recall  f1-score   support

           0       0.70      0.86      0.77        90
           1       0.81      0.63      0.71        90

    accuracy                           0.74       180
   macro avg       0.76      0.74      0.74       180
weighted avg       0.76      0.74      0.74       180
```

E.2 Environment Specifications Performance Testing Carried out

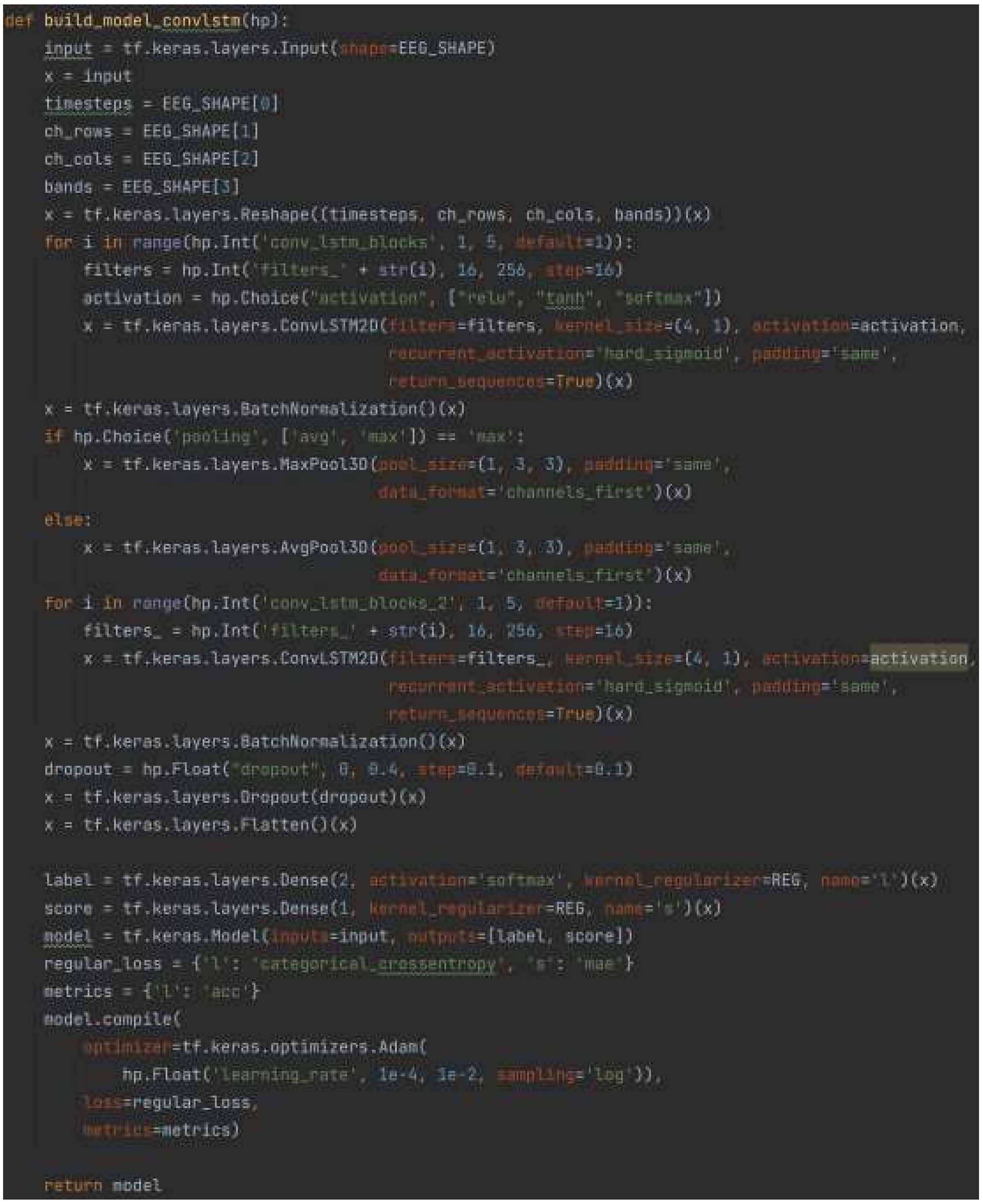
Processor	- Intel(R) Core (TM) i7-8750H CPU @ 2.20GHz 2.20 GHz
Installed RAM	- 8.00 GB (7.86 GB usable)
System type	- 64-bit operating system, x64-based processor
Installed VGA	- Nvidia GeForce 4.00Gb GTX-1050-Ti
Google Chrome Version	- Version 108.0.5359.125 (Official Build) (64-bit)

Appendix F- Evaluation Chapter

F.1 Brute-Force Number of Layers in ConvLSTM

No. of ConvLSTM Layers	No.of Dense Layer	Learning Rate	Accuracy
2	1	0.0005	93%
2	2	0.0005	75%
2	3	0.0005	73.5%
2	4	0.0005	76.87%
4	1	0.0005	69%
4	2	0.0005	67.5%
4	3	0.0005	58.7%
4	4	0.0005	52.35%

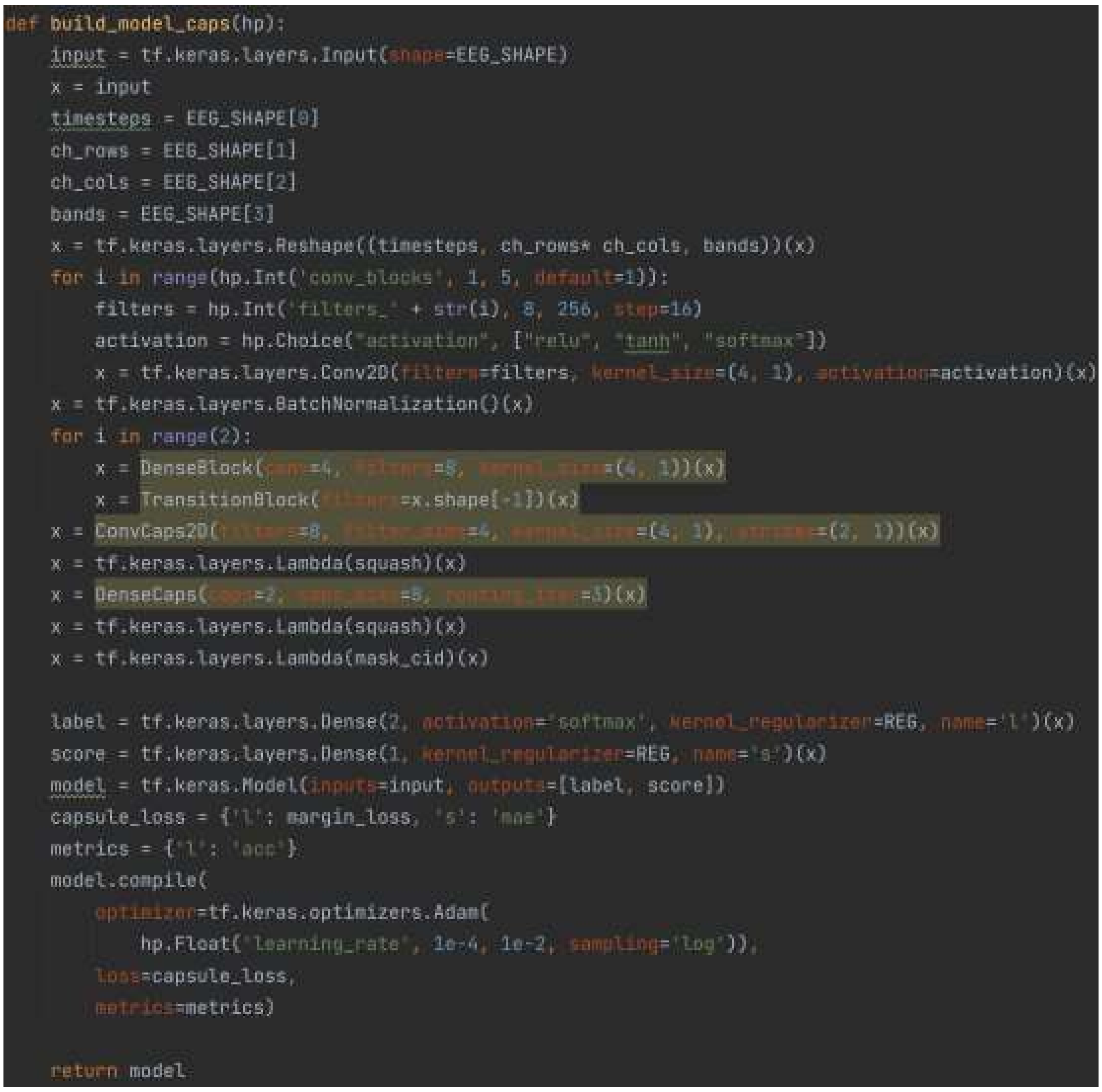
F.2 Hyperparameter tuning for ConvLSTM



```python
def build_model_convlstm(hp):
    input = tf.keras.layers.Input(shape=EEG_SHAPE)
    x = input
    timesteps = EEG_SHAPE[0]
    ch_rows = EEG_SHAPE[1]
    ch_cols = EEG_SHAPE[2]
    bands = EEG_SHAPE[3]
    x = tf.keras.layers.Reshape((timesteps, ch_rows, ch_cols, bands))(x)
    for i in range(hp.Int('conv_lstm_blocks', 1, 5, default=1)):
        filters = hp.Int('filters_' + str(i), 16, 256, step=16)
        activation = hp.Choice("activation", ["relu", "tanh", "softmax"])
        x = tf.keras.layers.ConvLSTM2D(filters=filters, kernel_size=(4, 1), activation=activation,
                                       recurrent_activation='hard_sigmoid', padding='same',
                                       return_sequences=True)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    if hp.Choice('pooling', ['avg', 'max']) == 'max':
        x = tf.keras.layers.MaxPool3D(pool_size=(1, 3, 3), padding='same',
                                      data_format='channels_first')(x)
    else:
        x = tf.keras.layers.AvgPool3D(pool_size=(1, 3, 3), padding='same',
                                      data_format='channels_first')(x)
    for i in range(hp.Int('conv_lstm_blocks_2', 1, 5, default=1)):
        filters_ = hp.Int('filters_' + str(i), 16, 256, step=16)
        x = tf.keras.layers.ConvLSTM2D(filters=filters_, kernel_size=(4, 1), activation=activation,
                                       recurrent_activation='hard_sigmoid', padding='same',
                                       return_sequences=True)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    dropout = hp.Float("dropout", 0, 0.4, step=0.1, default=0.1)
    x = tf.keras.layers.Dropout(dropout)(x)
    x = tf.keras.layers.Flatten()(x)

    label = tf.keras.layers.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(x)
    score = tf.keras.layers.Dense(1, kernel_regularizer=REG, name='s')(x)
    model = tf.keras.Model(inputs=input, outputs=[label, score])
    regular_loss = {'l': 'categorical_crossentropy', 's': 'mae'}
    metrics = {'l': 'acc'}
    model.compile(
        optimizer=tf.keras.optimizers.Adam(
            hp.Float('learning_rate', 1e-4, 1e-2, sampling='log')),
        loss=regular_loss,
        metrics=metrics)

    return model
```

Figure D-0.1 -Hyperparameter tuning for ConvLSTM

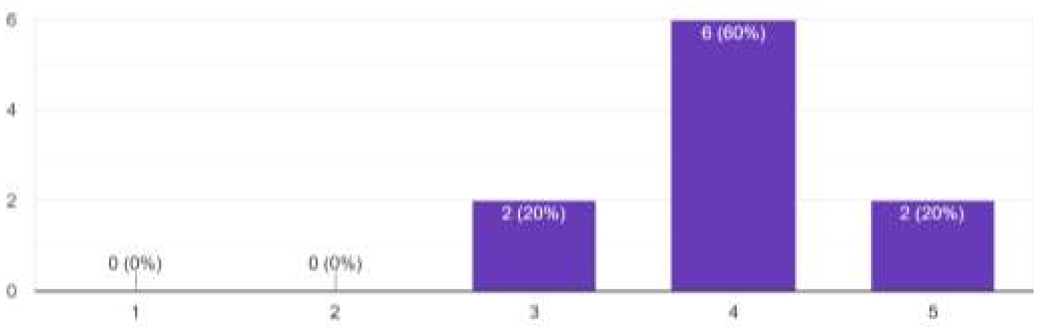
F.3 Hyperparameter tuning for CapsNet

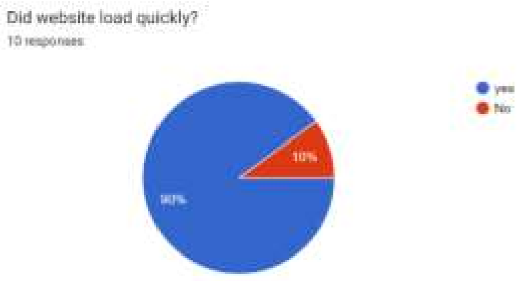


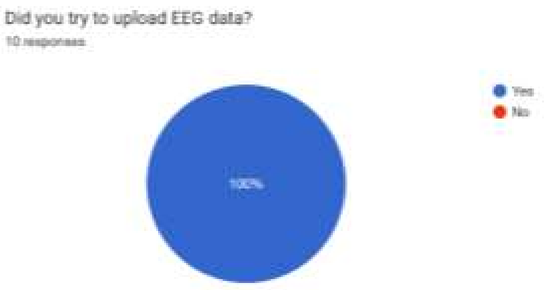
```
def build_model_caps(hp):
    input = tf.keras.layers.Input(shape=EEG_SHAPE)
    x = input
    timesteps = EEG_SHAPE[0]
    ch_rows = EEG_SHAPE[1]
    ch_cols = EEG_SHAPE[2]
    bands = EEG_SHAPE[3]
    x = tf.keras.layers.Reshape((timesteps, ch_rows* ch_cols, bands))(x)
    for i in range(hp.Int('conv_blocks', 1, 5, default=1)):
        filters = hp.Int('filters_' + str(i), 8, 256, step=16)
        activation = hp.Choice("activation", ["relu", "tanh", "softmax"])
        x = tf.keras.layers.Conv2D(filters=filters, kernel_size=(4, 1), activation=activation)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    for i in range(2):
        x = DenseBlock(conv=4, filters=8, kernel_size=(4, 1))(x)
        x = TransitionBlock(filters=x.shape[-1])(x)
    x = ConvCaps2D(filters=8, filter_dims=4, kernel_size=(4, 1), strides=(2, 1))(x)
    x = tf.keras.layers.Lambda(squash)(x)
    x = DenseCaps(caps=2, caps_dims=8, routing_iter=3)(x)
    x = tf.keras.layers.Lambda(squash)(x)
    x = tf.keras.layers.Lambda(mask_cid)(x)

    label = tf.keras.layers.Dense(2, activation='softmax', kernel_regularizer=REG, name='l')(x)
    score = tf.keras.layers.Dense(1, kernel_regularizer=REG, name='s')(x)
    model = tf.keras.Model(inputs=input, outputs=[label, score])
    capsule_loss = {'l': margin_loss, 's': 'mae'}
    metrics = {'l': 'acc'}
    model.compile(
        optimizer=tf.keras.optimizers.Adam(
            hp.Float('learning_rate', 1e-4, 1e-2, sampling='log')),
        loss=capsule_loss,
        metrics=metrics)

    return model
```

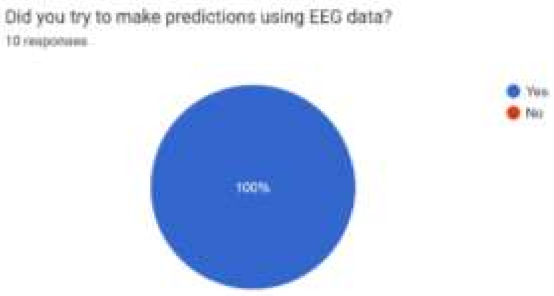
F.4 Prototype Quality Evaluation

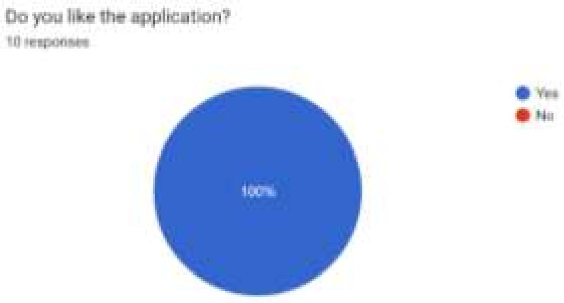
What is your first impression when you access the website?
10 responses



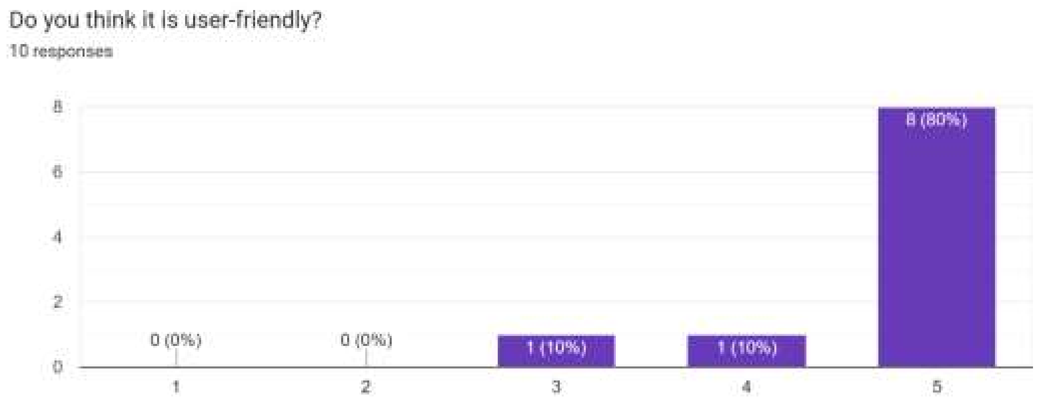


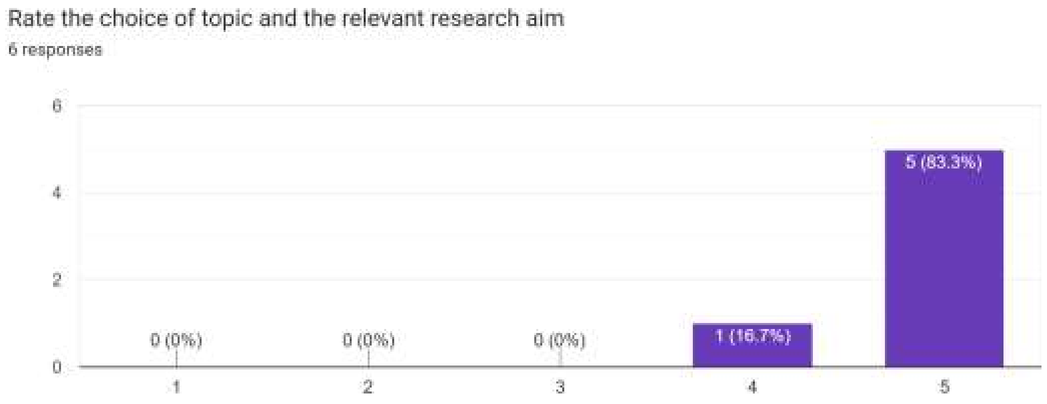




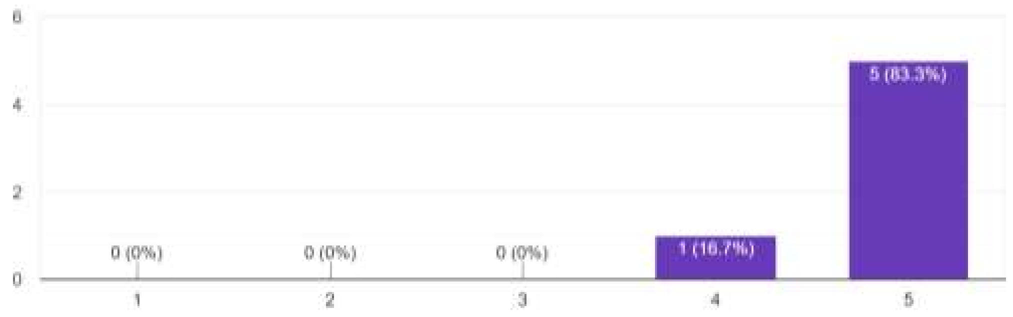


F.5 Qualitative Evaluation Results

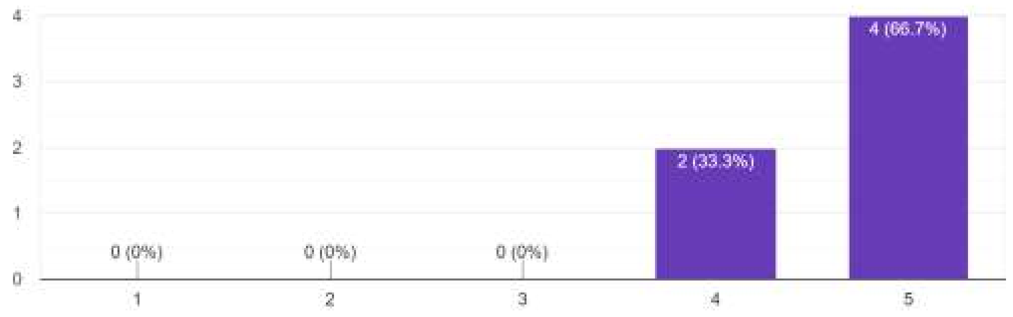




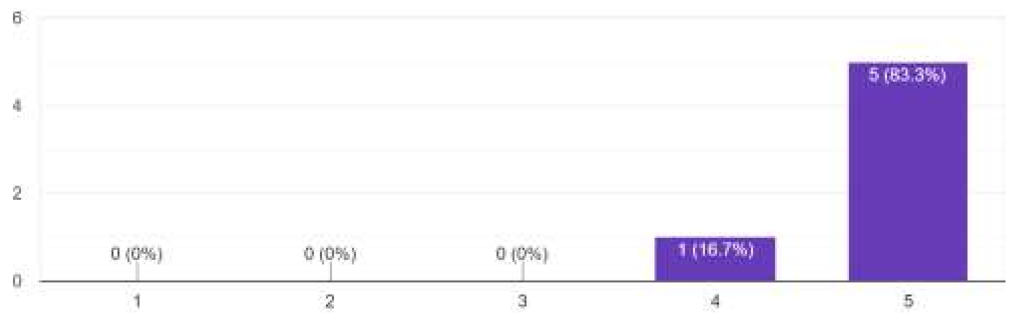
How would you rate the technical approach
6 responses



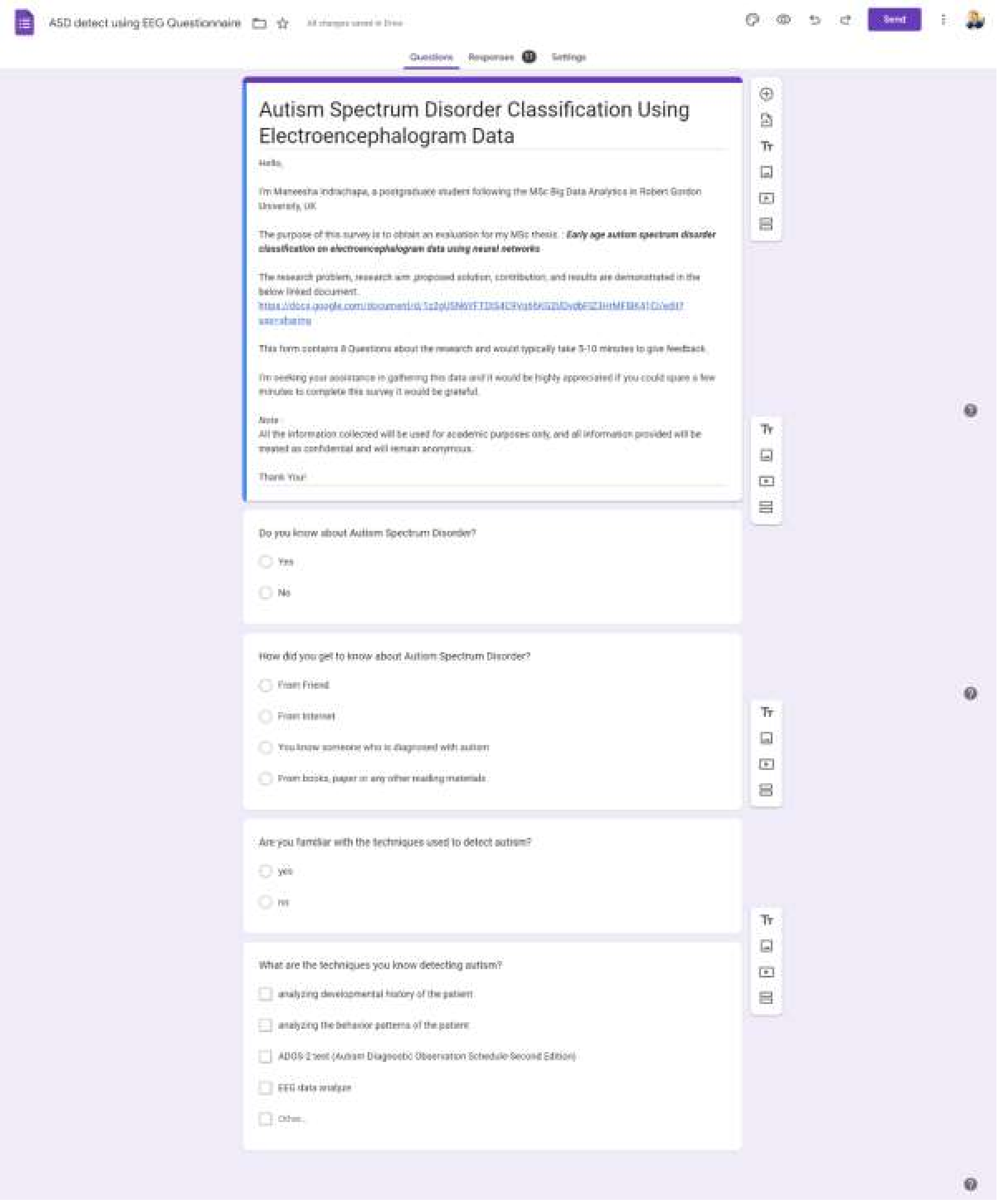
How would you rate the evaluation criteria
6 responses



How would you rate the user experience of the prototype
6 responses



F.6 Qualitative Evaluation Questionnaire


ASD detect using EEG Questionnaire
All changes saved in Drive
Send
Questions
Responses
Settings

Autism Spectrum Disorder Classification Using Electroencephalogram Data

Hello,

I'm Maneesha Indrachapa, a postgraduate student following the MSc Big Data Analytics in Robert Gordon University, UK

The purpose of this survey is to obtain an evaluation for my MSc thesis : ***Early age autism spectrum disorder classification on electroencephalogram data using neural networks***

The research problem, research aim ,proposed solution, contribution, and results are demonstrated in the below linked document.
https://docs.google.com/document/d/1z2qUSNkVFTDlS4C9Vq6bKG2lDvdbFlZ3HrMFB6A1Co/edit?usp=sharing

This form contains 8 Questions about the research and would typically take 5-10 minutes to give feedback.

I'm seeking your assistance in gathering this data and it would be highly appreciated if you could spare a few minutes to complete this survey it would be grateful.

Note :
All the information collected will be used for academic purposes only, and all information provided will be treated as confidential and will remain anonymous.

Thank You!

Do you know about Autism Spectrum Disorder?

- ○ Yes
- ○ No

How did you get to know about Autism Spectrum Disorder?

- ○ From Friend
- ○ From Internet
- ○ You know someone who is diagnosed with autism
- ○ From books, paper or any other reading materials

Are you familiar with the techniques used to detect autism?

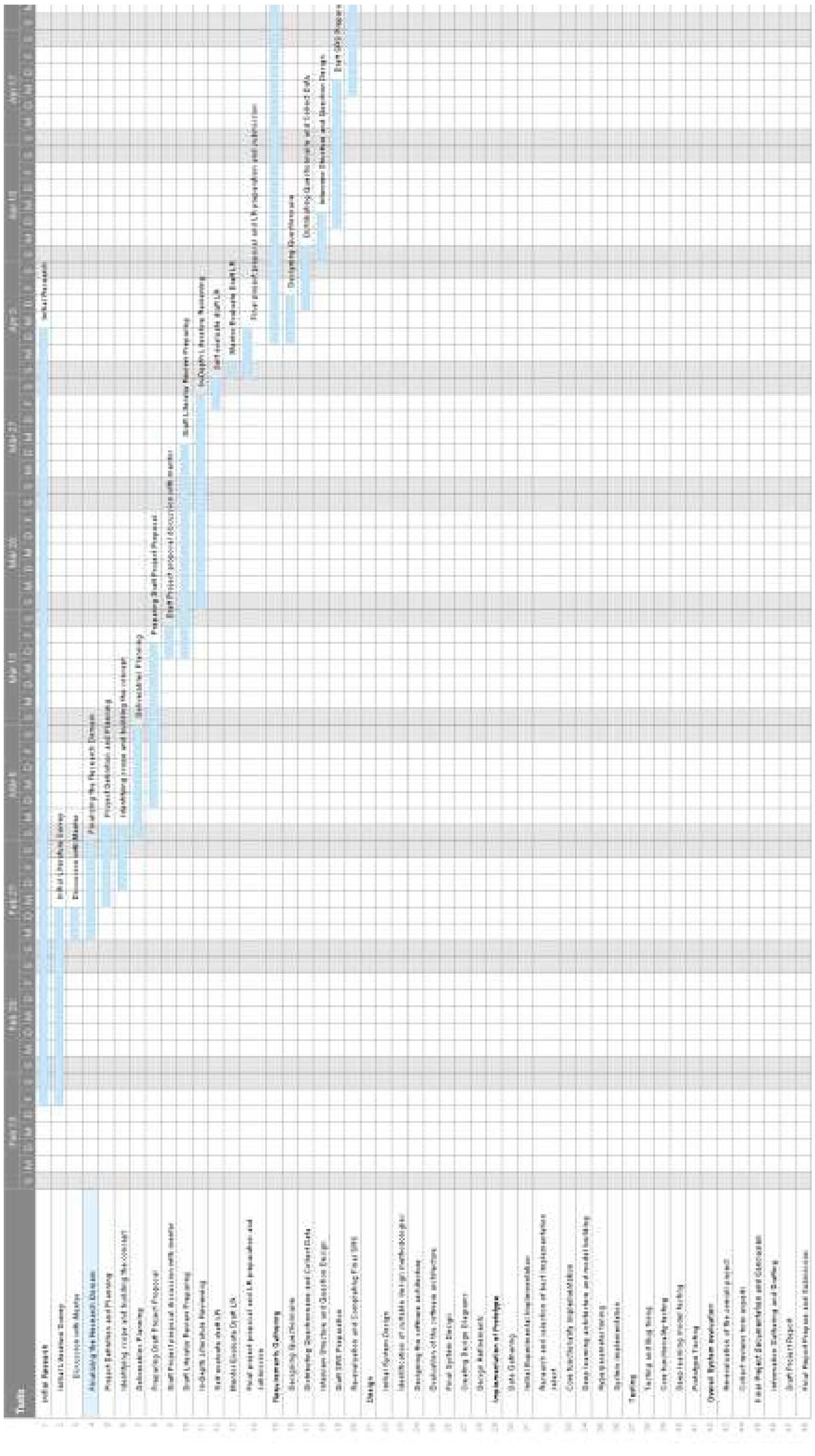
- ○ yes
- ○ no

What are the techniques you know detecting autism?

- ☐ analyzing developmental history of the patient
- ☐ analyzing the behavior patterns of the patient
- ☐ ADOS-2 test (Autism Diagnostic Observation Schedule-Second Edition)
- ☐ EEG data analyze
- ☐ Other...

Appendix G

G.1 Gantt Chart



Maneesha Indrachapa [2016894]

Appendix H

H.1 Installation Guide

Python and Libraries Install

For this project, model training Python 3.7.x was used. To download and install Python 3.7.x please use this link

After successfully installing the Python go inside the src folder and check it has a file named *requirements.txt* run the below command to install the required libraries to run the 'eeg_data_asd.py" script.

```
pip install -r 'reqiurements.txt'
```

After all the libraries are installed can run the python scripts as mentioned in the user guide.

Install Front-end Libraries

For this project, Angular-CLI 9.1.15 was used. Install the respective version on your machine and go inside the project and run the below command to install the necessary libraries

```
npm install
```

Install Back-end Server

For this project Flask was used as the backend server. Install Flask and Flask_CORS packages using the pip command.

H.2 User Guide

How to Train Models

Go to the source directory and run the below command to read and combine all data.

```
python eeg_data_asd.py R
```

After appending all the data to the "data-original. ftr" file run the below command to pre-process EEG data

```
python eeg_data_asd.py P
```

After that pre-processing pipeline will start and pre-processed data will save as a "data-clean. ftr" file. Then to create the frequency bands dataset run the below command.

```
python eeg_data_asd.py G
```

To Train, Test and get model info run the below command.

```
python train_test.py [parameter] [model_name]
```

In the above command *[parameter]* need to be *[train | test | info]* and *[model_name]* need to be *[conv | lstm | bilstm | caps | gru |convlstm]*

How to Tune Hyperparameters

To tune, the hyperparameters run the below command while in the source folder

```
python hyperparameter_select.py [model_name]
```

Visualize the Frequency Bands of Each Participant

To visualize, each participant's frequency bands power spectrum graphs run the below command to generate the graphs

```
python vizualize.py
```

Run Web Application

To run the front-end and back-end of the web application, run the below commands in the respective folders,

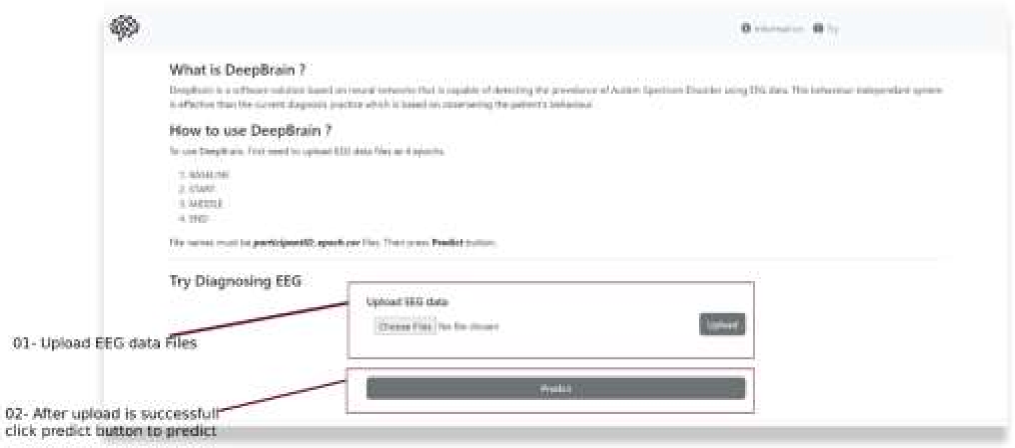
For front-end go inside AutismSpectrumDisorderDetectUI and run the below command

```
ng serve
```

For the back end go inside AutismSpectrumDisorderAPI and run the below command

```
flask --app server run
```

After running the application go to *https://localhost:4200/* link to access the website.



What is DeepBrain ?

DeepBrain is a software solution based on neural networks that is capable of detecting the prevalence of Autism Spectrum Disorder using EEG data. This behaviour independent system is effective than the current diagnosis practice which is based on observing the patient's behaviour.

How to use DeepBrain ?

To use DeepBrain, First need to upload EEG data files as 4 epochs.

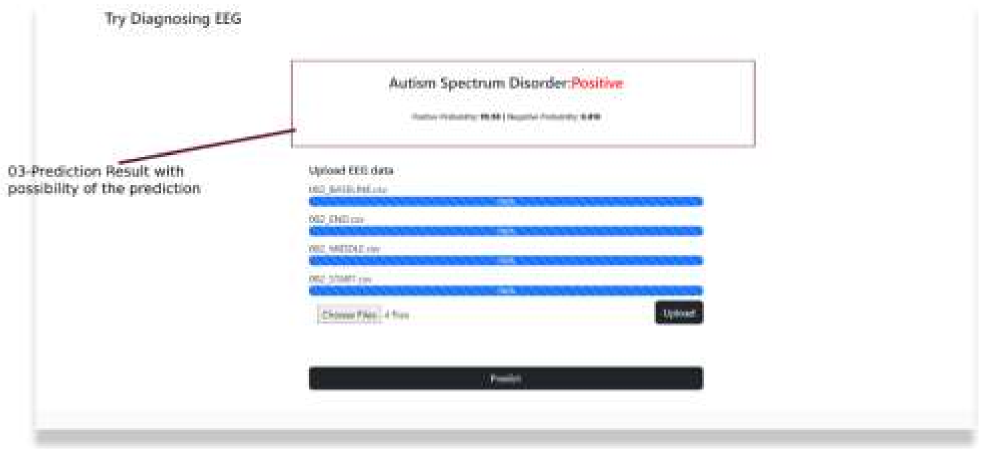
1. BASELINE
2. START
3. MIDDLE
4. END

File names must be **participantID_epoch.csv** files. Then press **Predict** button.

Try Diagnosing EEG

Upload EEG data
Choose Files
No file chosen
Upload
Predict
01- Upload EEG data Files
02- After upload is successfull click predict button to predict

1- On the website to do a prediction you need to upload 4 EEG files for each epoch. BASELINE, START, MIDDLE and END EEG files and the files should be in ".csv" format.
2- After uploading the files click the "Predict" button to do the prediction.


Try Diagnosing EEG
Autism Spectrum Disorder:Positive
03-Prediction Result with possibility of the prediction
Upload EEG data
002_END.csv
Choose Files 4 files
Upload

3-Prediction result with the accuracy of the prediction is shown.