FYP - Venkata Krishna Chaitanya Bysani - TP062476 - APD3F2308CSDA.docx

by VENKATA KRISHNA CHAITANYA BYSANI .

Submission date: 21-Apr-2024 04:19PM (UTC+0800)

Submission ID: 2356391924

File name: 41490_VENKATA_KRISHNA_CHAITANYA_BYSANI_._FYP_-_Venkata_Krishna_Chaitanya_Bysani_-_TP062476_-_APD3F2308CSDA__1296874792.docx (8.27M)

Word count: 22647

Character count: 137588

EFFICIENT DETECTION OF HUMAN MOTION MOVEMENTS THROUGH EEG SIGNALS USING DEEP LEARNING

By

VENKATA KRISHNA CHAITANYA BYSANI

TP062476

APD3F2308CSDA

A report submitted in partial fulfilment of the requirements for the degree of

B.Sc. (Hons) Computer Science Specialism in Data Analytics

at Asia Pacific University of Technology and Innovation.

Supervised by Dr. Mukil Alagirisamy

2nd Marker: Assoc Prof. Dr. Raja Rajeswari

2024

DECLARATION OF THESIS CONFIDENTIALITY

Author's full name: **VENKATA KRISHNA CHAITANYA BYSANI**

IC No./Passport No.: **U3283917**

Thesis/Project title: **EFFICIENT DETECTION OF HUMAN MOTION MOVEMENTS THROUGH EEG SIGNALS USING DEEP LEARNING**

I declare that this thesis is classified as:

☐ CONFIDENTIAL

☐ RESTRICTED

☑ OPEN ACCESS

I acknowledged that Asia Pacific University of Technology & Innovation (APU) reserves the right as follows:

1. The thesis is the property of Asia Pacific University of Technology & Innovation (APU).
2. The Library of Asia Pacific University of Technology & Innovation (APU) has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Author's Signature: ..... *Venkata krishna chaitanya* .....

Date: 15 April 2024

Supervisor's Name: **Dr. Mukil Alagirisamy**

Date: 15 April 2024

Signature: 

Please fill in **all** the following details for library cataloguing purposes.

First Name: Venkata krishna chaitanya
Middle Name (only if applicable):
Last Name: Bysani
Title of the Final Year Project / Dissertation / Thesis: Efficient detection of human motion movements through EEG Signals using deep learning
Abstract: The main goal of this research project is to ensure healthy lives and promote well-being. Specifically, human motion will be efficiently detected by the application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), to Electroencephalography (EEG) data. The project highlights the unique insights offered by EEG data while addressing the important medical topic of precisely identifying human motion movements. The study uses a systematic method that includes extensive data collecting, evaluation, and modelling by utilizing the CRISP-DM technique. The selected models for implementation include EEGNet, DeepConvNet, ShallowConvNet, EEG-TCNet, and ATCNet, showcasing a diverse set of approaches to address the research problem effectively. Preliminary findings indicate positive outcomes in the realm of human motion detection through EEG signals. The project's significance lies in its potential to contribute to the field of rehabilitation and medical diagnostics, particularly in the development of advanced assistive technologies for individuals with motor impairments. "The ATCNet demonstrated superior performance with an accuracy of 79.71% and a k-score value of 0.729 in the conducted evaluations."
A few keywords associated with the work: EEG signals, deep learning, Convolutional Neural Networks, human motion detection, CRISP-DM methodology.
General Subject: (e.g Management information systems, Organizational behaviour, Risk management, Computer Software) Computer Science, Data Analytics, Deep learning
Date of Submission: 24 April 2024

Acknowledgement:

I would like to truly acknowledge my parents for their consistent guidance and encouragement during the course of my studies. Without their constant support and inspiration, I could not have kept myself on the correct track as well as would not have achieved this extent. Moreover, I would like to express my sincere gratitude to my buddies, who have been my stable supporters and were always there for me—especially when things became hard. Their cooperation and support were essential to doing this task effectively. Furthermore, I would like to express my appreciation to APU and Dr. Mukil, my supervisor, for giving me the chance and direction to work on my final year project research. Dr. Mukil's beneficial suggestions and opinions were very helpful in increasing the overall standard of this report.

Abstract:

The main goal of this research project is to ensure healthy lives and promote well-being. Specifically, human motion will be efficiently detected by the application of deep learning techniques, particularly Convolutional Neural Networks (CNNs), to Electroencephalography (EEG) data. The project highlights the unique insights offered by EEG data while addressing the important medical topic of precisely identifying human motion movements. The study uses a systematic method that includes extensive data collecting, evaluation, and modelling by utilizing the CRISP-DM technique. The selected models for implementation include EEGNet, DeepConvNet, ShallowConvNet, EEG-TCNet, and ATCNet, showcasing a diverse set of approaches to address the research problem effectively. Preliminary findings indicate positive outcomes in the realm of human motion detection through EEG signals. The project's significance lies in its potential to contribute to the field of rehabilitation and medical diagnostics, particularly in the development of advanced assistive technologies for individuals with motor impairments. "The ATCNet demonstrated superior performance with an accuracy of 79.71% and a k-score value of 0.729 in the conducted evaluations."

Keywords: EEG signals, deep learning, Convolutional Neural Networks, human motion detection, CRISP-DM methodology.

SDG Goal 3: Ensure healthy lives and promote well-being for all at all ages.

Contents

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction:

In today's world, there is a significant number of people facing neurological disabilities, rendering them dependent on others for basic tasks involving their hands and legs. This project, "Efficient Detection of human motion movements through EEG signals using Deep learning," has the capability of offering these people crucial support and enabling them to regain control over their life.

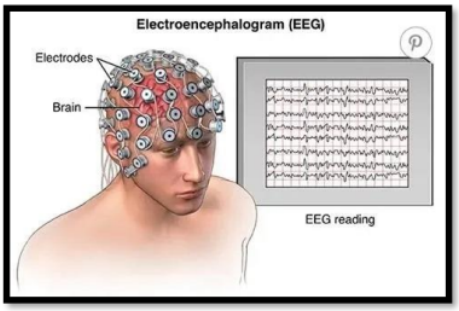


Figure 1 - EEG Signals reading.

The scientific approach of choice for recording the electrical activity produced by the brain using electrodes embedded in the scalp is electroencephalography or EEG. Data may be obtained from identical scalp placements throughout all respondents because of electrodes attached in elastic caps like bathing caps, which allows for much quicker application. (Imotions,2021).

A relatively new method of developing contact between humans and machines, the brain-computer interface (BCI) converts brain activity into instructions to facilitate interaction and control. BCI is an interesting way for paralyzed people to communicate with the outside world since it can identify human intentions. The most popular invasive method for capturing activity in the brain is the electroencephalogram (EEG), which is used by many of the current BCI systems (Hongkong University of Science and technology, N.A). Motor imagery (MI) is one of the most popular BCI paradigms, in which people visualize moving their limbs to operate the system. MI-based BCIs have an extensive number of prospective applications, such as supportive technology for paralyzed patients, neurorehabilitation, and gaming (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023).

Deep learning for MI-based BCI control has drawn more attention in the past few years. Deep learning methods are now being used to create innovative and modern BCI systems since they have been proven to produce advanced results on a range of BCI datasets. In the past several years, deep learning, a class of machine learning algorithms, has transformed a few industries, including speech recognition, computer vision, and natural language processing. Because they can recognize complicated patterns in data and extract features that are challenging to identify using conventional techniques, deep learning algorithms are particularly well suited for BCI applications (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023).

1.2 Problem Background

Detecting human motion movements efficiently through EEG signals using deep learning is crucial due to the inherent challenges and limitations in current technologies, especially in specific environmental contexts. Electroencephalography (EEG) signals play a pivotal role in brain-computer interfaces (BCIs), empowering individuals to control devices or communicate with their minds. However, these signals face various challenges that impede their normal development.

- Limited performance as the number of mental tasks increases:

As users perform more mental tasks, the capability of current EEG-based Brain-Computer Interfaces (BCIs) deteriorate. The intricate structure of the brain, with its multiple functions, complicates data from electroencephalogram (EEG) recording. As the complexity of mental tasks advances, thus does the data, demanding the use of advanced machine learning models for proper task classification. (Soroosh Shahtalebi & Amir Asif, 2020)

- Noise and redundant information in multi-channel EEG:

According to Jing Jin & Chang Liu (2020), inefficiency has been triggered by an elevated amount of noise and redundancy in multi-channel EEG data. Retrieval of relevant data about the user's mental state from such complicated terrain becomes laborious, limiting the overall efficiency of EEG-based BCIs. The data from an EEG, which is prone to ambient influences as well as basic human behaviors such as breathing and emotions, fluctuates rapidly. EEG data recording beneath ideal conditions is hard and leads to massive quantities of noise and redundant information in the dataset.

- Poor time-frequency localization as well as excessive non-stationarity of EEG signals:

Real-time monitoring of fluctuations in the user's mental state has been hampered by the significant non-stationary behaviour and poor time-frequency localization of EEG data. These fundamental characteristics restrict the flexibility of EEG-based BCIs, especially when dealing with dynamic contexts with frequent variations. Overall, EEG data is unstable and very sensitive in nature**.** (Pramod Gaur, Ram Bilas Pachori, 2019).

- Low signal-to-noise ratio (SNR) of EEG signals:

The low Signal-to-Noise Ratio (SNR) of EEG signals makes them vulnerable to external interferences including muscle activity, eye movements, and power line disturbances. This drawback hinders the reliability of EEG-based BCIs, especially in circumstances with considerable amounts of external interference. Even tiny body movements causes random fluctuations in EEG signals, and their reliance on external sounds contributes to their low signal-to-noise ratio, making reliable motion detection difficult. (Pramod Gaur, Ram Bilas Pachori, 2019).

- Feature extraction:

Extraction of important features from EEG data is a complicated procedure that adds to the underperformance of current EEG-based BCIs in comprehending human motion movements. The classification of four unique human motion movements, right hand, left hand, tongue, and both feet, introduces an additional layer of complexity to the already challenging EEG data, causing extraction of pertinent features difficult at times. (Fazel-Rezai, Kumar, & Shenoy, 2018).

1.3 Project Aim:

The aim of this project ("Efficient Detection of human motion movements through EEG signals using Deep learning ") is to develop a model using deep learning that can accurately predict the intended motor imagery task of a user. This could have a significant impact on people with disabilities, as it could enable them to control prosthetic devices or other assistive technologies using their thoughts.

1.4 Objectives:

1. To build a deep learning model that can recognize various motor imagery tasks from EEG data, such as left- and right-hand motions, and tongue and foot movements.
2. To Create a communication pathway for people with neurological disabilities in order that they can communicate properly and express their requirements and wishes.
3. To build effective motor imagery based BCIs to aid in neurorehabilitation, improving motor recovery as well as the quality of life for people with spinal cord injuries and cerebral paralysis.
4. To Develop a machine learning model for classifying motor imagery that achieves at least 80% accuracy on a held-out test set.
5. To Create models using identified algorithms and evaluate them to find the best model.

1.5 Scope:

The project encompasses several key steps in the efficient detection of human motion movements through EEG signals using deep learning. The initial phase involves data collection, where EEG data, crucial for the subsequent stages, is sourced from Kaggle, eliminating the need for manual recording. Following this, the data undergoes a meticulous pre-processing step, involving the removal of artifacts, noise filtration, and adjustments to ensure its compatibility with deep learning models.

The primary task is defined as the identification and classification of human motion movements, specifically targeting right hand, left hand, feet, and tongue motions based on EEG signals. Subsequently, the project advances to the development of a deep learning model. This phase encompasses the design and implementation of a suitable architecture for EEG signal processing. Exploration and experimentation with diverse neural network architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or hybrid models, are undertaken to achieve optimal performance. Attention mechanisms or other relevant techniques are considered to capture temporal dependencies in EEG data.

A critical step involves the extraction of significant features from the preprocessed EEG data. Various feature extraction techniques are explored to enhance the representation of EEG signals for precise motion detection. This may involve the incorporation of time-frequency representations, spectrograms, or other domain-specific features to improve model interpretability.

The subsequent stage focuses on the training of the developed models. The dataset is partitioned into training, validation, and test sets, facilitating the training of the deep learning model using appropriate loss functions and optimization algorithms. Strategies such as data augmentation are implemented to enhance the generalization of the model. Evaluation metrics, including accuracy, precision, recall, and F1 score, are defined for comparative analysis across different models and subject-specific evaluations.

It's essential to note that the project concludes with the classification task, and no physical platform, such as prosthetic limbs or wheelchairs, is built during this process. Instead, the project serves as a foundational framework for the development of tools like limbs or devices in the future. Specifically, it focuses on motor-imagery task classification in EEG processing, addressing a fixed four-class problem involving the detection of right hand, left hand, tongue, and both feet movements. The project does not extend to techniques like emotion detection, monitoring drowsiness or attention, monitoring concentration, or tasks related to visual or auditory evoked potentials and imagined speech.

1.6 Potential Benefits:

The potential benefits come into two primary categories: intangible benefits and tangible benefits. Hard benefits, commonly referred to as tangible benefits, can be easily calculated because they are measurable and quantitative. On the other hand, intangible benefits or soft benefits become more difficult to measure accurately. For an in-depth evaluation of any operation's outcomes, both categories are necessary. While intangible benefits include features of significant worth that may not be immediately quantified, tangible benefits give quantitative measurements. It is important to recognize and balance these two kinds of benefits to fully evaluate the total worth and effect of a project. (iSixSigmastaff,2022)

1.6.1 Tangible benefit of this project:

- Improved control of prosthetic limbs: BCIs that analyze EEG data and improve them into signals for prosthetic limbs might assist individuals who suffer from paralysis or other motor disabilities restore power of their movement. It might allow users to be more independent and autonomous in everyday activities like eating, dressing, and self-care. (Jiarong Wang | Luzheng Bi | Weijie Fei , 2023)

- Enhanced rehabilitation for stroke victims: Brain-computer interfaces have the potential to be utilized in stroke victims' treatment for rehabilitation, giving them immediate assistance while they engage in tasks to help them restore lost motor function. This customized strategy has the possibility to enhance the health of patients and speeds up the healing process. (Gege Zhan | Shugeng Chen | Yanyun Ji, 2022)
- Development of new assistive technologies: BCIs could potentially be employed for developing a variety of novel assistive technology products, like as smart wheelchairs or brain-controlled communication devices, that might improve the lives of individuals with disabilities.

1.6.2 Intangible benefits of the project:

- Reduced reliance on caregivers: By permitting people with disabilities to work out tasks on themselves, BCIs could decrease their requirement for care provider's support. These might contribute to better self-assurance, an enhanced standard of life, and a more powerful sense of freedom. (Mamunur Rashid | Norizam Sulaiman | Sabira Khatun, 2020)
- Improved mental well-being: BCIs may improve psychological well-being by minimizing feeling of loneliness and helplessness. The capacity to operate devices as well as interact with the surroundings independently could improve self-worth and encourage a better sense of empowerment.
- Advancements in understanding the brain: Further study into the use of BCI technology may aid in an improved understanding of the function of the brain in regulating movement. The knowledge gained could be employed for developing novel treatments for neurological disorders and enhance our knowledge of human cognition.

1.6.3 Target users:

An identified user group that will be the subject of user experience design or analysis is known as a target user. (John Spacey,2023) The project's target audience consists of those with neurological disorders. Particularly, people with neurological disorders of various kinds represent our target audience. The project's results, which aim to improve the detection of human motion movements using EEG data using deep learning, may be advantageous to these

users. By adjusting the techniques, we use to suit the requirements of people with neurological disabilities, the project aims to promote the development of assistive technology, which may enhance their quality of life.

1.7 Overview of the IR

The project begins in Chapter 1, which also gives an overview of the difficulties in effectively detecting human motion movements using EEG data using deep learning. The researcher outlines the problems that currently exist in this field of study and highlights the need for creative solutions. This chapter provides the foundation for the next chapters by investigating how machine learning may be used to solve problems in the context of human motion detection. The researcher also describes the problems that are bound to be solved to complete the project.

Chapter 2 provides an extensive exploration of the literature, progressing from an overview of the domain to a detailed analysis of algorithms, technologies, and concepts that are relevant to it. Every subtopic in the area is extensively investigated and then similar systems and works are analysed with a focus on their characteristics, errors, and a similarity. subsequently, the chapter closes with a thorough examination of the technical factors, which includes system requirement analysis and optional selections of operating systems, web servers, hardware, software, IDEs, libraries, and database management systems.

Chapter 3 has been highlighted the key role of the Crisp-DM methodology in directing the project effectively. This methodology comprises of six key steps such as Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. It starts with defining project goals and understanding business requirements, followed by determining data quality and relevance. Data is then prepared for modelling, where suitable techniques are applied and evaluated based on project objectives. Successful models or insights are then deployed, showcasing Crisp-DM's structured process for obtaining valuable data-driven insights aligned with business requirements.

Chapter 4, focused on design and implementation, begins by identifying and explaining the dataset used in the project, followed by crucial procedures for data understanding such as creating histograms and visualizations to understand variables as well as observations. The chapter explores into initial data pre-processing, covering aspects such as data transformation, handling missing values, outliers, standardization, normalization, and more to ensure data quality and readiness for analysis using preferred tools. Furthermore, the chapter describes the

building of key models essential for project success: ATCNet, EEGNet, DeepConvNet, ShallowConvNet, and EEG-TCNet. Each model's unique strengths and capabilities are explored in detail, including their architectures, training processes, along with performance evaluations.

In chapter 5, "Results and Discussion," focuses on evaluating as well as comparing the earlier built models based on their accuracies. This evaluation process includes thorough testing and analysis to evaluate each model's performance metrics. Additionally, the best-performing model is selected for deployment on a simple website, showcasing its real-world applicability and user-friendliness. This chapter delves to showcase the outcomes of the model evaluations, facilitating discussions on their strengths, limitations, and practical implications in real-world situations.

In Chapter 6, named "Conclusion," provides a comprehensive summary and reflective analysis of the project on deep learning-based human motion detection from EEG signals. It consists key findings, limitations, achievements, and recommendations for future research. The chapter also references the Gantt chart, lists references, and includes relevant appendices to support the conclusions effectively.

1.8 Project Plan

Project Plan				
Task	**Start Date**	**End Date**	**Duration**	**Status**
Project Proposal Form	Sep 10, 2023	Sep 14,2023	4 Days	Done
“Chapter-1: Introduction”	Sep 14, 2023	Sep 25,2023	11 Days	Done
1.1 Introduction	Sep 14, 2023	Sep 15, 2023	1 Days	Done
1.2 Problem Background	Sep 15, 2023	Sep 16, 2023	1 Days	Done
1.3 Project Aim	Sep 16, 2023	Sep 17, 2023	1 Days	Done
1.4 Objectives	Sep 17, 2023	Sep 18, 2023	1 Days	Done
1.5 Scope	Sep 18, 2023	Sep 21, 2023	3 Days	Done
1.6 1Potential Benefit	Sep 21, 2023	Sep 23, 2023	2 Days	Done
1.7 Overview of the IR	Sep 23, 2023	Sep 24, 2023	1 Days	Done
1.8 Project Plan	Sep 24, 2023	Sep 25, 2023	1 Days	Done
“Chapter-2: Literature Review”	Sep 25, 2023	Oct 15, 2023	20 Days	Done
2.1 Introduction	Sep 25, 2023	Sep 27, 2023	2 Days	Done
2.2 Domain Research	Sep 27, 2023	Oct 7, 2023	10 Days	Done
2.3 Similar Systems	Oct 7, 2023	Oct 10, 2023	3 Days	Done
2.4 Technical Research	Oct 10, 2023	Oct 14, 2023	4 Days	Done

2.5 Summary	Oct 14, 2023	Oct 15, 2023	1 Days	Done

"Chapter-3: Methodology"	Oct 15, 2023	Oct 23, 2023	8 Days	Done
3.1 Introduction	Oct 15, 2023	Oct 18, 2023	3 Days	Done
3.2 Methodology	Oct 18, 2023	Oct 22, 2023	4 Days	Done
3.3 Summary	Oct 22, 2023	Oct 23, 2023	1 Days	Done

"Chapter-4: Design and Implementation"	Feb 01, 2024	Mar 06, 2024	34 Days	Done
4.1 Introduction	Feb 01, 2024	Feb 02, 2024	1 Days	Done
4.2 Data Collection	Feb 02, 2024	Feb 04, 2024	2 Days	Done
4.3 Data Pre-processing	Feb 04, 2024	Feb 07, 2024	3 Days	Done
4.4 Data Understanding	Feb 07, 2024	Feb 08, 2024	1 Days	Done
4.5 Model Building	Feb 08, 2024	Mar 04, 2024	25 Days	Done
4.6 Summary	Mar 04, 2024	Mar 06, 2024	2 Days	Done

"Chapter-5: Results and Discussion"	Mar 06, 2024	Mar 31, 2024	25 Days	Done
5.1 Introduction	Mar 06, 2024	Mar 07, 2024	1 Days	Done
5.2 Model Evaluations and Discussions	Mar 07, 2024	Mar 15, 2024	8 Days	Done
5.3 Model Deployment	Mar 15, 2024	Mar 28, 2024	13 Days	Done
5.4 Summary	Mar 28, 2024	Mar 31, 2024	3 Days	Done

"Chapter-6: Conclusion"	Mar 31, 2024	Apr 15, 2024	15 Days	Done
6.1 Critical Evaluation	Mar 31, 2024	Apr 02, 2024	2 Days	Done
6.2 Limitation	Apr 02, 2024	Apr 07, 2024	5 Days	Done
6.3 Recommendation	Apr 07, 2024	Apr 11, 2024	4 Days	Done
References, Appendices, Gantt chart	Apr 11, 2024	Apr 15, 2024	4 Days	Done

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

The main purpose of this investigation is to evaluate the accuracy of using deep learning to effectively recognize human motion movements from EEG data. This investigation compares the accuracy of the six models used to determine which is the most effective. The project will compare the CRISP-DM and KDD methodologies using machine learning technology, which is mostly created in Python. The results will be shown using an interface after accuracy validation. The project involves collecting a sizable dataset containing information about EEG from open sources such as Kaggle. Key components of the research are covered in the Literature Review section, which emphasizes the use of Deep Learning in combination with EEG inputs for efficient human motion detection. It also investigates related Machine Learning Algorithms along with delves at related works within the area.

2.2 Domain Research

2.2.1 Bio-Medical Field:

The medical domain incorporates a range of educational fields that are devoted to comprehending, identifying, managing, and averting medical issues in humanity. The most unique among all of these subdivisions is biomedical research, which utilises sophisticated data analysis, clinical trials, and laboratory testing to investigate biological processes, illnesses, and therapies. The research we conduct works mainly in the field of neuroscience and concentrates on the brain, which is an essential organ that governs almost every bodily function.

The neurological system's center, the brain, maintains all our internal operations and tasks. The brain rain transmits commands to different bodily parts through complex neural networks built of individual neurons. Our investigation aims at employing electroencephalography (EEG), an imaging method which records brain signals, for decoding neural communications.

Large datasets and an overabundance of data have become vital to the modern medical environment. By availing use of this plethora of information our project attempts to expand the boundaries of health care advancement. In order to be more accurate, we employ EEG data for enhancing rehabilitation methods. Analyzing these signals in the brain attempts to provide novel insights that advance techniques for rehabilitation and enhance results for patients.

In summary, our study on comprehending EEG signals fits in line with neuroscience's goal of discovering the human brain's peculiarities while employing this information to reevaluate rehabilitation techniques for enhanced treatment of patients.

2.2.2 EEG Signals:

Electrical impulses are used to communicate with the neurological system. A method for assessing the electrical activity produced in the brain is electroencephalography (EEG), which provides a window into neural activity and brain function. Electrodes applied to the scalp capture the electrical field produced by the nerve cells to measure the EEG signal. (Elana Zion,2017)

Elana Zion (2017) also says that with a resolution of a single millisecond or less, this non-invasive approach provides high-accuracy time measurements that are essential for catching fast changes in electrical brain activity. The advantage of having non-invasive access to a healthy human brain adds to its appeal for brain research. Furthermore, EEG equipment is inexpensive and easy to use.

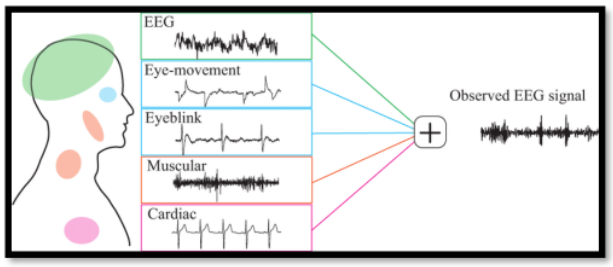


Figure 2 - EEG signals.

EEG has some significant disadvantages: limited spatial resolution, regardless of the advantages. The signal that is collected at the scalp during measurements is the sum of the electric field produced by an extensive number of neurons. A single electrode can only resolve spatial information within around one centimeter of the cortex, which is host to many neurons. It is more difficult to distinguish between closely adjacent regions and to identify the precise source of activity when there is a lack of spatial accuracy. However, advanced methods of EEG analysis have been developed to improve the precision of signal source estimates, hence reducing this obstacle.

Furthermore, according to Michael X Cohen (2017), this will contribute to advancements in fundamental neuroscience, cognitive neuroscience, clinical diagnoses, and the development of data analysis techniques.

2.2.3 Brain Computer Interface (BCI):

BCIs, often referred to as brain–machine interfaces (BMIs) or smart brains, create a direct channel of communication between the brain's electrical activity and an external device like a computer or robotic limb. A Brain-Computer Interface (BCI) acts as a bridge for the brain and a device, establishing brain signals to control external actions such as moving a cursor or moving a prosthetic limb. This interface enables a direct line of contact, allowing the brain's signals to directly affect the target object. In the case of cursor control, for instance, the signal travels straight from the brain to the guidance system of the cursor, bypassing the traditional route through the body's neuromuscular system. They are used for many different things, such as mapping, studying, aiding, enhancing, or restoring human cognitive or sensory-motor skills. BCI's, which are conceptualized as a human-machine interface, blur the line between the brain and the machine by doing away with the necessity for bodily movement. (TechTarget contributor, 2023).

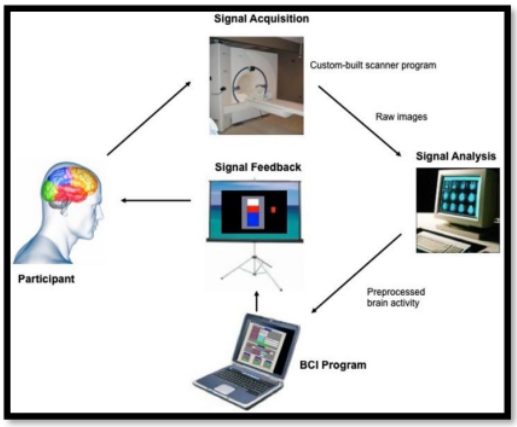


Figure 3 - BCI

Furthermore, BCIs come in three types: non-invasive, like EEG (measured on the scalp); invasive, such as ECoG (measured on the cortex); and intracortical. These systems convert electrophysiological signals into actions, enabling the operation of devices for various purposes. They help patients with movement and communication issues, support rehabilitation efforts for motor and cognitive problems, and make activities like word processing, cursor

movement, robot and prosthesis control, and motor rehabilitation easier (Daly and Wolpaw, 2008).

Additionally, rebuilding neuromuscular connections and promoting plasticity in the brain appear to be potential benefits of integrating active motor BCIs with neurofeedback. Although our study acknowledges the existence of invasive BCIs, it focuses on non-invasive EEG-based motor BCIs, in line with the suggested project approach for effective deep learning-based human motion movement recognition from EEG signals. This focus on EEG aligns with the objectives of the project by offering non-invasive, practical, and direct insights into the electrical activity of the brain. (Weijie Fei | Jiarong Wang | Luzheng Bi, 2023)

2.2.4 Machine learning:

Machine learning, within the realms of artificial intelligence (AI) and computer science, centres on employing data and algorithms to replicate the learning process observed in humans, with a continuous enhancement of its precision over time. Machine learning involves the utilization of computational techniques to leverage user experience data for making precise predictions. "Experience" refers to history or previous user data, that is easily available for data analysis. The primary theme of machine learning is to develop efficient as well as accurate algorithms. Implementing machine learning applications improves user experiences, reducing difficult methods of complex algorithms for developers. (Mohri et al., 2018)

Machine learning methods:

There are three main types of machine learning models.

The use of labelled datasets to train algorithms for precise data classification or outcome prediction essentially describes supervised learning. The model modifies its weights when input data is entered until an ideal fitting is achieved. Either classification- or regression-based problems may be involved. It is a regression-based problem where the output value is continuously distributed. If the output is limited to discrete values, the challenge is dependent on categorization. Neural networks, naïve Bayes, logistic regression, random forest, linear regression, and support vector machines (SVM) are a few techniques used in supervised learning. (geeksforgeeks,2023)

Unsupervised learning analyzes and groups unlabeled datasets using machine learning methods. These algorithms find hidden connections or patterns in the data without requiring

human assistance. It may also be applied to dimensionality reduction, which lowers the number of features in a model. Two popular methods for this are singular value decomposition (SVD) and principal component analysis (PCA). Neural networks, probabilistic clustering techniques, and k-means clustering are a few more algorithms utilized in unsupervised learning. (geeksforgeeks,2023)

A suitable choice between supervised and unsupervised learning is offered by semi-supervised learning. It guides classification and feature extraction from a larger, unlabeled data set during training by using a smaller, labelled data set. The issue of insufficient labelled data for a supervised learning system can be resolved through semi-supervised learning. If enough data cannot be labelled due to cost, this is also beneficial. (geeksforgeeks,2023)

Reinforcement machine learning, akin to supervised learning, doesn't rely on sample data for training. Instead, it learns through trial and error, reinforcing a sequence of successful outcomes to develop the optimal recommendation or policy for a given problem. (geeksforgeeks,2023)

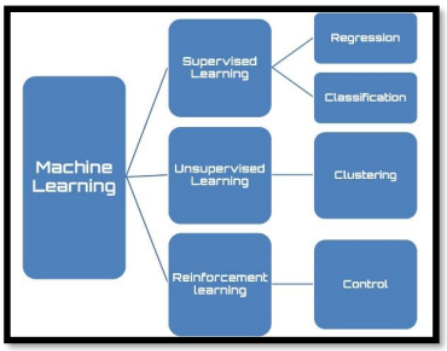


Figure 4 - Types of ML (packt,n.d)

Machine Learning makes computers learn on their own with the use of large datasets, predicting or classifying based on input data. In this project "Efficient Detection of Human Motion Movements through EEG Signals Using Deep Learning," machine learning, especially deep learning on EEG signals, finds and recognises human motion. This improves brain-computer interfaces by decoding comprehending EEG patterns. Widely adopted, 41% of organizations address challenges with Machine Learning and AI, benefiting sectors like data security, finance, healthcare, fraud detection, and retail through various learning methods.

2.3 Similar works:

Hamdi Altaheri, Ghulam Muhammad, and Mansour Alsulaiman used the BCI Competition IV-2a dataset in the research they conducted, which included 5184 trials carried out on nine patients using 22 EEG electrodes and three EOG electrodes. Each trial was connected to one of four motor imagery (MI) tasks, which included visualizing the tongue, right hand left hand, and both foot movements. For the classification of these MI tasks, the researchers used an Attention-based Temporal Convolutional Network (ATCNet), which consists of three main blocks: the Temporal Convolution Block (TC), designed to extract high-level temporal features, the Multihead Self-Attention Block (AT), that highlights important information within the temporal sequence, and the Convolution Block (CV), responsible for encoding raw MI-EEG signals into concise temporal sequences. The accuracy and Kappa score measures used in performance evaluation showed that using the BCI Competition IV-2a dataset, the ATCNet model was superior to the state-of-the-art methods. (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023).

In the research Conducted by Jing Jin and Chang Liu aimed to build a model that could recognize EEG channels with improved temporal as well as spatial characteristics. Three different datasets—the BCI Competition IV dataset 1, the BCI Competition III dataset IVa, and the BCI Competition III dataset IIIa—were used to validate their methodology. They extracted features from EEG signals using bi spectrum analysis, a method that is proficient at extracting non-linear and non-Gaussian information, including the Sum of Logarithmic Amplitudes (SLA) and the First Order Spectral Moment (FOSM). These features were subsequently utilized in the Bispectrum-based Channel Selection (BCS) procedure to identify the best channels for a BCI based on Motor Imagery (MI), or motor imagery. In order to extract suitable features from the selected channels, the Common Spatial Pattern (CSP) method was subsequently used. Using accuracy as the performance criteria, Support Vector Machine (SVM) and Linear Discriminatory Analysis (LDA) classifiers were used for task classification. When compared to conventional 3C-CSP approaches, the findings showed considerable performance improvements, with accuracy rates for datasets 1, 2, and 3 of 83.8%, 86.3%, and 77.8%, respectively. (Jing Jin | Chang Liu, 2020)

In research including both healthy volunteers and stroke patients, V.K. Benzy and A.P. Vinod used BCI based on EEG to interpret the directions of imagined hand movements. A motorized arm support was given control instructions using the decoded left/right-hand movement imagining. The synchronization parameter Phase Locking Value (PLV), derived from EEG, was employed by the researchers to figure out the directionality of the MI task. The time bin relevant to the MI task was chosen using event-related desynchronization/synchronization (ERD/ERS) analysis on the Mu and Beta frequency bands of EEG. By choosing the most significant channel pairings that offered the greatest variation in PLV characteristics, the differences between the two orientations of MI tasks were discovered. In both healthy people (63.7% to 82.8%) and stroke patients (56.9% to 74.4%), the classification accuracy was shown to be enhanced by employing ERD/ERS analyses, according to the researchers. Additionally, a stroke patient feedback session was held, increasing the classification accuracy to 68.63%. According to their research, EEG-based BCI may be utilized to operate prosthetic devices along with other assistive technologies for both healthy individuals and stroke patients. (V.K. Benzy | A.P. Vinod, 2020)

Tianwei Shi, Ling Ren, and Wenhua Cui conducted a concentrated investigation in their research to solve two-class classification issues in motor imagery EEG. Their research focused on categorizing EEG signals associated with left-hand vs. right-hand, left-hand vs. foot, and right-hand vs. foot images. The most effective time-frequency filter type and filtering range was chosen for the study's preliminary processing using wavelet packet analysis. The filtered EEG data received additional refining in the feature extraction process utilizing the Common Spatial Pattern (CSP) and Adaptive Auto-regressive (AAR) methods. Band energy, sample entropy, and order accumulation were all investigated as potential distinguishing factors for the categorization of motor imagery. The study also performed a comparison of the classification results obtained using Bayesian, common space, and linear discrimination classifiers. Additionally, the training set data allowed for the co-space model's best spatial filter to be derived, which maximizes the variation among the two job categories. The classification performance showed a benefit from the improved distance criterion. his improved distance criterion has a positive effect on classification rate and proves that the proposed method can effectively extract the features of EEG signals during motor imagery. (Tianwei Shi | Ling Ren | Wenhua ,2020)

On the other research conducted by two famous authors, to discriminate between left- and right-hand motor imagery despite the dataset's four initial classifications, Lukasz Radzinski, and Tomasz Kocejko's research used the BCI Competition IV dataset 2a. To reduce higher-frequency noise and eye movement artifacts, band-pass filtering was first used during preprocessing. Their model used a variety of methods, such as Deep and Shallow Convolutional Neural Networks (CNNs), the Common Spatial Pattern (CSP), the Filter Bank Common Spatial Pattern (FBCSP), and data augmentation approaches. With the use of spatial filters that increase the variance ratio between the two classes, CSP was able to maximize signal separation. Prior to spatial separation via CSP, FBCSP, an improved form of CSP, employed frequency filtering across multiple banks. The study also looked at CNNs for classifying Motor Imagery (MI), with Deep ConvNet focusing on deep learning and Shallow ConvNet emulating FBCSP's results. Frequency shifting was used for data augmentation, and the augmented set was included with the initial training data. A detailed investigation of methodologies involved comparisons, and Shallow Cropped Aug+CSP showed the greatest accuracy at 79.17%. (Lukasz Radzinski | Tomasz Kocejko, 2022)

On the other research conducted, Hao Zhu, Dylan Forenzo, and Bin He utilized two of the largest publicly available datasets, namely the MBT-42 and Med-62 datasets. In the MBT-42 dataset, subjects were tasked with performing left or right cursor movement tasks, while the Med-62 dataset involved 62 subjects participating in cursor movement control tasks of three types: left/right (LR) movement only, up/down (UD) movement only, and combined 2D movement (2D). The study entailed a comparative analysis of five deep learning models: EEGNet, Shallow ConvNet, Deep ConvNet, MB3D, and ParaAtt. EEGNet, a convolutional neural network, comprised three convolutional layers and one fully connected layer, aiming to encode various EEG feature extraction concepts, including optimal spatial filtering and filter-bank construction. Deep and Shallow ConvNet, two variations of convolutional neural networks, featured distinct architectural designs. Multi-Branch 3D CNN (MB3D) accepted CxT matrices as input, transforming the 2D input into a 3D tensor and subsequently utilizing 3D convolutional layers for generating predictions. Parallel Self-Attention Network (ParaAtt) incorporated attention modules, effectively capturing global relationships among input entries. The research encompassed both within-subject and cross-subject analyses, ultimately

demonstrating that EEGNet outperformed other methods for both datasets (Hao Zhu | Dylan Forenzo | Bin He, 2022).

In the other study conducted by Xin Deng, Boxian Zhang, and Nian Yu used data from two open-access datasets: the BCI Competition IV 2a dataset from 2008, which had four motor imagery classes, and the BCI Competition III IIIa dataset, which featured the same classes but was recorded using 60 electrodes. They used the Filter-Bank Common Spatial Pattern (FBCSP), EEGNet, TSGL-EEGNet, Model Saving, and Ensemble Methods in their investigation, which combined all five of these techniques. While EEGNet used a convolutional architecture with feature selection using Temporal-constrained Sparse Group Lasso (TSGL), FBCSP added frequency domain features to the CSP technique to improve it. Model Saving techniques reduced training time, and Ensemble Methods, particularly stacking, improved the performance of the final model. With an average accuracy of 88.89% and an average kappa value of 0.8519, the TSGL-EEGNet stacking approach outperformed both FBCSP and EEGNet by 7.48% and 11.04%, respectively (Xin Deng | Boxian Zhang | Nian Yu, 2021).

The table is shown below.

Authors	**Title**	**Dataset Used**	**Analysis Techniques**	**Evaluation Techniques**	**Outcomes**	**Limitations**
Hamdi Altaheri, Ghulam Muhammad, Mansour Alsulaiman, 2023	Physics-Informed Attention Temporal Convolutional Network for EEG-Based Motor Imagery Classification.	BCI Competition IV dataset 2a. Four class classification task- right hand, left hand, tongue and both feet.	Attention-based Temporal Convolutional Network (ATCNet). Algorithm contains three blocks. Temporal block to extract high-level temporal features. Multihead Self-Attention block highlights important information within temporal sequence. Convolution	Accuracy, Kappa score	The proposed model demonstrated a powerful ability to extract MI features from a raw EEG signal without artifact removal and with minimal preprocessing using a limited-size and challenging	The proposed algorithm has an accuracy of 70.97 % for subject independent tasks which should be boosted further.

			block converts the data into encoded form.		dataset. Made a comparison with recent other powerful algorithms like EEGNet, EEG-TCNet, TCNet-sion.	
Jing Jin, Cha Liu, 2020	Bispectrum-Based Channel Selection for Motor Imagery Based Brain-Computer Interfacing.	Three different datasets – BCI Competitio n IV dataset 1, BCI Competitio n III dataset IVa, and BCI Competitio n III dataset IIIa	Bispectrum analysis is plied and Sum of Logarithm Amplitudes (SLA) and the First Order Spectral Moment (FOSM) terms are used for selection of t channels. Common Spatial Pattern (CSP) is used to extract suitable features from the selected nnels. Support Vector Machine (SVM) and Linear Discriminatory Analysis (LDA) techniques are used for classification from derived features.	Accuracy, F-Score	SLA and FOSM features, extracted from bispectrum, were used to calculate the F score for each channel. According to the F score, channels without redundant information are selected. Compared to the 3C-CSP and CSP-rank methods, the proposed BCS algorithm can achieve significantl y better results. Overall, the BCS algorithm is an efficient method to improve the performanc e of MI based BCIs.	SLA and FOSM features had the same contribution to the process of F score calculation. Considering different weights, may improve pe rmanc e. The effect of different frequency ranges of the filter on the calculation of bispectrum is not to be considered.

V.K. Benzy, A.P. [illegible]od, 2020	Motor Imagery Hand Movement Direction Decoding Using Brain Computer Interface to Aid Stroke Recovery and Rehabilitation.	The dataset consists of records of 12 healthy subjects, 20 stroke patients. Two class problem- left hand and right hand.	Event Related Desynchronization Analysis is made on Mu and Beta frequency bands of EEG. The relevant time bin is selected based on this analysis. Based on Phase Locking Value (PLV) value, the most significant channel pairings are chosen.	Accuracy, ROC curve, Area under ROC curve (AUC)	This is the first study that explicitly explores the idea of decoding imagined hand movement direction from stroke patients to aid stroke recovery and rehabilitation. The analyses on the proposed methodology demonstrate the ability of MI based-BCI system in identifying directional information healthy subjects and stroke patients using ERD/ERS analysis.	The mean classification accuracy obtained for 16 patients is 74.44% in the calibration session and 68.63% in feedback session. To apply this for stroke rehabilitation, it may require movement intention-based information which it lacks.
Tianwei Shi, Ling Ren, Wenhua ,2020	Feature Extraction of Brain–Computer Interface Electroencephalogram Based on Motor Imagery	Two-class classification issues are solved. [illegible]ey can be left-hand vs right-hand, left-hand vs foot, right-hand vs foot.	EEG feature extraction algorithm based on common spatial pattern (CSP) and adaptive auto-regressive (AAR), and demonstrates feasibility of band energy, sample entropy and order accumulation to be the characteristics of motor	Distance criterion, accuracy, band energy, Power spectral density	The simulation results show that the proposed method can effectively extract the features of EEG signals during motor imagery.	The limitations of the study include the sensitivity of EEG signals to external reactions, the reduced classificatio accuracy due to a limited number of electrodes, and the potential for

			imagery classification, and finally compares the classification effects of linear discrimination classifier, common space classifier and Bayesian classifier.			error in classification when EEG signal noise is large.
Lukasz Radzinski, Tomasz Kocejko, 2022	Deep learning approach on surface EEG based Brain Computer Interface.	BCI Competition IV data 2a. Two class problem-left hand and right despite of initial four class initial classifications.	Deep and Shallow Convolutional Neural Networks (CNNs), the Common Spatial Pattern (CSP), the Filter Bank Common Spatial Pattern (FBCSP), and data augmentation approaches.	Accuracy	Training the convolutional models with data proceeded by CSP transformation or with frequency shift data augmentation improved the accuracy of movement imagery classification.	A maximum of only 80% accuracy is achieved with the above-mentioned techniques.
Hao Zhu, Dylan Forenzo, Bin He, 2022	On the Deep Learning Models for EEG-Based Brain-Computer Interface Using Motor Imagery.	Two open access datasets MBT-42 and Med-62 are used.	Five deep learning models are used, they are EEGNet, Shallow ConvNet, Deep ConvNet, MB3D, and ParaAtt.	Accuracy, model training and inference time, t-statistic	The research encompassed both within-subject and cross-subject analyses, ultimately demonstrating that EEGNet outperformed other methods for both datasets.	Building a unified model which can be trained on various types of data objective is not achieved.
Xin Deng, Boxian Zhang, Nan Yu, 2021	Advanced TSGL-EEGNet for Motor Imagery EEG-Based Brain-Computer Interfaces.	Two open access datasets BCI Competition IV dataset	They used the Filter-Bank Common Spatial Pattern (FBCSP), EEGNet,	Accuracy, Kappa value, training time	With an average accuracy of 88.89% and an average kappa value	Potential limitations may include the need for further validation

		2a and BCI Competition III IIIa dataset.	TSGL-EEGNet, Model Saving, and Ensemble Methods in their investigation, which combined all five of these techniques.		of 0.8519, the TSGL-EEGNet stacking approach outperformed both FBCSP and EEGNet by 7.48% and 11.04%, respectively 21 he use of the average-validation method and simplified bagging method has been shown to improve the performance of the model.	of the proposed TSGL-EEGNet model on larger and more diverse datasets. Additionally, the inter 14 ability of the deep learning models and the sensitivity of the models to individual subjects are discussed.

2.4 Technical Research:

System requirement analysis:

Python Language:

Python is a popular programming language with many features and applications that was developed by Guido van Rossum in the 1980s. Python is a great option for a variety of development scenarios because of its dynamic semantics, object-oriented, high-level design, and interpretation. (Python, n.d)

Figure 5 - Logo of the python (Python, n.d)

Python's extensive standard libraries empower users to engage in tasks such as document handling, networking, and database management. This capability stands out as just one of the many strengths of the language. Moreover, Python benefits from a vibrant developer community that actively contributes additional libraries, enhancing the language's inherent efficiency. (Coursera,2023)

Python finds applications in various domains, like website development, machine learning, data analysis, and scientific computing. Python is the best choice for scripting and automation tasks, showing its versatility and user friendly in nature. (python, n.d).

The following details provide further information on the characteristics and features of the Python programming language.

- Dynamic Semantics: Python is a good choice for Rapid Application Development (RAD) because of its dynamic semantics, advanced built-in data structures, dynamic typing, and dynamic binding. (geeksforgeeks,2023)
- Availability: All the major platforms include free source or binary versions of the Python interpreter and a large standard library which is accessible for unrestricted distribution. (geeksforgeeks,2023)
- Active Community: The continuous development and maintenance of Python is made possible by a sizable and dynamic community. (geeksforgeeks,2023)
- Versatile and open source: The fact that Python is free, open source, as well as versatile has led to its widespread use. (geeksforgeeks,2023)
- Dynamic Typing: Python's dynamic typing allows for flexibility in runtime variable usage. (geeksforgeeks,2023)
- Hundreds of libraries and frameworks: Development tasks are made simpler with a large library and framework collections. (geeksforgeeks,2023)

Additionally, Python makes functional programming easier, allowing programmers to write clear, understandable code. Lambda expressions, list comprehensions, and the map, filter, and reduce functions are a few examples of these functions. (W3 schools, n.d)

R Language:

R is a graphical and statistical computing platform and programming language. At the University of Auckland in New Zealand, it was developed in the 1990s by Ross Ihaka and Robert Gentleman. Researchers, data scientists, and statisticians commonly use R to analyze and show data. (Programiz, n.d)

Figure 6 - R programming language (The Indian wire staff, 2019)

The following details provide further information on the characteristics and features of the R programming language.

- Statistical Language: R offers a wide range of statistical and mathematical functions and is particularly made for statistical computation and data analysis. (Techvidvan, n.d)
- Extensive Libraries: R is well known for having a vast library and package collection. Numerous statistical techniques, machine learning algorithms, and data manipulation tools are covered by these packages. (Techvidvan, n.d)
- Open source: Since R is an open-source language, it encourages creativity and teamwork. The source code may be freely viewed, modified, and distributed by users. (Techvidvan, n.d)
- Data Visualization: R's remarkable skills in data visualization are one of its best characteristics. It provides strong capabilities for making a wide variety of charts, plots, and graphs that visually represent data. (Techvidvan, n.d)

- Community Support: The community of statisticians, data scientists, and academics using R is large and vibrant. Through forums and online resources, this community offers support and continues to contribute to the language's continued growth. (Techvidvan, n.d)
- Interoperability: R is compatible with C, C++, and Java, among other languages. With this feature, customers may take use of existing codebases in various languages for certain functions. (Techvidvan, n.d)
- Data Frames: Data frames are a tabular data format that makes dealing with structured datasets easier, and R supports them. Managing relational data requires the use of this functionality. (Techvidvan, n.d)
- Platform Independence: Because of its platform-independent architecture, R is adaptable to various operating systems, encompassing Linux, macOS, and Windows. (Techvidvan, n.d)

The most widely used computer language for statistical analysis and modelling is R. R has advantages as well as disadvantages much like other programming languages. Since R is a language that continuously evolves, most of its disadvantages will gradually disappear with updated versions.

The following are some advantages of the R programming language:

- Statistical Analysis: R is a favoured tool for academics, statisticians, and data scientists working on intricate statistical modelling and data exploration because of its superior performance in statistical analysis. (Javatpoint,2023)
- Integration: Users can integrate the positive aspects of R with the capability of other programming languages because of R's smooth language integration. (Javatpoint,2023)
- Data Visualization: Its strong data visualization features improve the sharing of findings by enabling the production of informative charts that are suitable for publishing. (Javatpoint,2023)

The following are the disadvantages of the R programming language.

- Speed: R may occasionally perform more slowly than languages like C or Java, particularly when handling big datasets or complex computations. (Javatpoint,2023)

- Data Size limitations: Because of R's memory limitations, large datasets may present difficulties that call for code optimization for effective memory management. (Javatpoint,2023)
- Limited Libraries for non-statistical tasks: R is a powerful tool in statistical fields, but its applicability in non-statistical domains may be limited by the lack of libraries for general-purpose programming tasks. (Javatpoint,2023)

Comparison Between R and Python Programming language:

Parameter	R	Python
Primary Use	Statistical Computing, Data Analysis	General-Purpose Programming, Data Science
Syntax	focused on Statistics, Data Analysis	Versatile Syntax, Readable and clean
Ease of Learning	the more difficult learning curve for beginners	Beginners may find it somewhat simpler, Readable
Community Support	dynamic community with an emphasis on data	large, varied community across several domains
Data visualization	powerful capacities and abundant plotting libraries	large visualization libraries (like as Seaborn and Matplotlib)
Data Manipulation	potent tools for manipulating data	The Pandas library offers powerful tools.
Packages and Libraries	Comprehensive collection for statistical techniques	huge environment with a variety of libraries
Speed	Could take longer for some calculations.	Faster in general, especially with huge datasets
Memory Usage	Increased memory use and possible inefficiency	decreased use and effective memory management
Learning Curve	Greater difficulty in learning for non-statisticians	the comparatively lower learning curve

Usage in Industry	often employed in statistics, research, and academia	widespread usage in business for a variety of purposes

2.4.1 Best Language/Software for my Project:

Python is a programming language that is particularly strong at data analysis. It has good experimental and computational characteristics and can handle big datasets with ease. Its abilities include machine learning and artificial intelligence (AI). Python is more user-friendly than R because of its simple syntax, which is especially beneficial for beginners. Python has a built-in library collection and can be easily integrated with machine learning packages. Because of this, Python is the tool of choice for data analysts who use it to create machine learning algorithms as well as connect with collections of ML libraries.

After a careful analysis, the researcher has determined that Python is a better programming language than R for designing and executing the project on "Efficient Detection of Human Motion Movements through EEG Signals using Deep Learning" with ML library collections.

2.4.2 IDE (Interactive Development Environment) chosen:

An Integrated Development Environment (IDE) is a type of software. package which combines essential development utilities like a code editor, compiler, debugger, and terminal in a user-friendly interface. It streamlines tasks such as software editing, building, testing, and packaging, making the programming experience smoother for developers. With the use of an Integrated Development Environment (IDE), programmers may simplify several of the tasks associated with creating computer programs. (codecademyteam, n.d)

Some of the common characteristics of IDE are shown below:

- Editing Source Code: Programmers use an Integrated Development Environment (IDE) to write code, made up of instructions that instruct a computer on particular tasks. This code is written in a particular programming language, like Java or Python. An IDE offers features such as autocomplete for language keywords and syntax highlighting, contributing to code readability and ease of writing. (Joel Falconer, 2022)
- Compiler: A compiler transforms human-readable code to machine-specific code that allows it to be processed on multiple operating systems such as Linux, Windows, and

Mac OS. Several IDEs offer embedded compilers for each language it handles. (codecademyteam, n.d)

- Debugger: A software testing and debugging tool which may assist developers detect any errors or flaws within their scripts. (Joel Falconer, 2022)
- Code navigation: Integrated Development Environments (IDEs) include capabilities that include code folding, class and method navigation, and refactoring tools to assist you explore and evaluate code. (Joel Falconer, 2022)

IDEs enhance developer efficiency by automating tasks like code editing, debugging, and testing, streamlining the development process. They come pre-configured with tools for rapid application development, offering customization options for individual preferences. Notable IDEs include Visual Studio, JetBrains' PyCharm, and Android Studio.

2.4.2.1 Pycharm:

Python programming can be done using numerous IDEs and one of the most popular and widely used Integrated Development Environment (IDE) is PyCharm. It was made by the Czech company JetBrains and offers for all the operating systems like Windows, macOS, and Linux. PyCharm has available in two modes: Community Edition and Professional Edition. The Community Edition is offered for free, however the Professional Edition necessitates an annual membership. (Simran Kaur Arora, 2023)

Figure 7 - Logo of the PyCharm (Qanelph, 2022)

PyCharm serves as a widely recognised IDE for developing in Python since it includes an extensive number of features which assist programmers to write, debug, and evaluate their programme code more efficiently. PyCharm's major characteristics include the following:

- Smart Code Completion: PyCharm enables smart compilation of code, that enables developers to produce code faster and more effectively. It can suggest keywords, variables, features, as well as additional code components according to the specifics of the source code. (Simran Kaur Arora, 2023)
- Code Inspections: PyCharm may evaluate your source code for capacity issues and other vulnerabilities. It can identify capacity syntactical problems, typology flaws, as well as several logical errors. (Simran Kaur Arora, 2023)

Merits of PyCharm:

The main features of PyCharm which outshine compared to other IDEs are given below.

- Increased Productivity: PyCharm's smart code finishing touch, code inspections, and various additional capabilities may help you in creating code with greater efficiency and precision. This might end up with substantial boosts in performance. (geeksforgeeks,2023)
- Reduced errors: PyCharm's programmes testing enables you to recognise and connect possible flaws to the source code prior to executing it. This makes it possible for developers to develop more trustworthy and outstanding code. (geeksforgeeks,2023)
- It also enables you to capture music and correlate bugs with greater speed. (geeksforgeeks,2023)

Demerits of Pycharm:

Although PyCharm is a renowned and packed with functions integrated development environment (IDE) for Python, it does come with specific drawbacks and restrictions that programmers should be conscious of. These are some intriguing PyCharm potential dangers:

- Resource Intensive: PyCharm has become renowned thanks to its robust abilities nevertheless these attributes come at an expense with regard to of systems resources. The integrated development environment (IDE) might seem slightly resource-rich, particularly the Professional edition, but it can also lead to reduced performance on

earlier or less powerful systems. If you are working with a device with limited resources, you might encounter latencies and delays. (geeksforgeeks,2023)

- Steep Learning Curve: While PyCharm is incredibly effective, it could come with a lengthy learning curve, especially among amateurs. For somebody inexperienced in coding or Python, the sheer number of choices and configurations offered might be overwhelming. Users might also need effort to acquire knowledge about and configure the IDE to fulfil their needs. (geeksforgeeks,2023)
- Limited Language Support: While PyCharm was developed primarily for Python development, it might not represent the most suitable option for those who frequently interact with several programming languages. (geeksforgeeks,2023)

2.4.2.2 Visual Studio:

One powerful tool for developers that integrates the whole development process on one platform is Visual Studio. Together with app deployment, this all-inclusive integrated development environment (IDE) makes code drafting, editing, debugging, and generating code easier. Beyond tasks involving code, Visual Studio includes compilers, tools for code completion, source control, extensions, and other capabilities that improve the effectiveness of every stage of the software development cycle. (Microsoft, 2023)

The following common characteristics of visual studio are shown below:

- Rich Debugging tools: Strong debugging features in Visual Studio include code profiling and interactive debugging. (Softwarekeep, n.d)
- Integrated Development Environment (IDE): An integrated environment for writing, debugging, and deploying programs is offered by Visual Studio. (Softwarekeep, n.d)
- Comprehensive toolset: Compilers, code completion tools, source control, and other features supporting the whole development lifecycle are all included. (Softwarekeep, n.d)

Figure 8 - Logo of the visual studio (Fernando Munoz, 2018)

Using Visual Studio as an IDE has the following advantages:

- Tons of features: It decreases the requirement for third-party integrations by providing a large range of features, tools, and libraries. (Enrique Corrales, 2023)
- Vast Community and resources: Because it offers so many features, tools, and libraries, fewer third-party integrations are needed. (Enrique Corrales, 2023)
- Productivity: Because Visual Studio is integrated, developers can complete numerous activities on a single platform, which increases productivity. (Enrique Corrales, 2023)

While Visual Studio has plenty working for it and is a popular IDE for numerous, there are certain disadvantages that could convince people to go elsewhere. Among the disadvantages of Visual Studio are:

- Challenging for Beginners: For those who are new to coding, Visual Studio Code may be intimidating because of its large feature set, which may be more than they need right now. There can be a learning curve involved in using the editor effectively. (Muhammad Ahmad Mir, 2023)
- Higher Configuration Demands: It may seem like Visual Studio Code requires more setup than alternative code editors. This feature could be problematic for those that prefer a simpler configuration. (Muhammad Ahmad Mir, 2023)
- Resource Intensive: System resources can be heavily used by Visual Studio Code, particularly when managing several extensions or more complex projects. This feature could be problematic for developers using computers that are dated or underpowered. (Muhammad Ahmad Mir, 2023)

2.4.3 Libraries/Tools chosen:

Python has been chosen as the programming language for the "Efficient Detection of Human Motion Movements through EEG Signals using Deep Learning" project. The essential libraries and resources listed below will help ensure that this project is completed successfully:

2.4.3.1 NumPy:

The core Python library for scientific computing is called NumPy. A multidimensional array object, different derived objects (like masked arrays and matrices), and a variety of routines for quick array operations—like sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more—are all provided by this Python library. (NumPy, n.d)

Figure 9 – NumPy (Nick McCullum, 2020)

The Nd array object is the core of the NumPy package. This contains homogenous data types in n-dimensional arrays, and several operations are carried out in compiled code for efficiency.

2.4.3.2 Pandas:

Pandas is a Python library that provides quick and adaptable data structures that make working with relational or labelled data simple. It aims to be the most effective open-source data manipulation tool and acts as a basic building block for real-world data analysis. Python code is used to optimize Pandas, which is renowned for its speed. Being dependent on stats models, it is essential to the statistical computing ecosystem and has a track record of production reliability, especially in financial applications. (Pandas,2023)

Figure 10 - Pandas (eLearn, 2023)

Pandas works well with a wide variety of data types, including:

- Tabular data, like an Excel spreadsheet or SQL table, with columns containing different categories of data. (Pandas,2023)
- Time series data, not limited to certain frequencies and either sorted or unordered.
- All kinds of matrix data with defined row and column labels, whether it is homogeneous or heterogeneous. (Pandas,2023)
- Any type of statistical or observational data sets can be stored in a pandas' data structure without the need for specific labelling. (Pandas,2023)

For quicker data preparation in "Efficient Detection of Human Motion Movements through EEG Signals Using Deep Learning," the Python pandas' library is essential. It makes effective use of Data Frames and other strong structures to manipulate and clean the EEG information.

Furthermore, pandas easily interface with other project libraries, ensuring a faster procedure for effective data preparation and consequent training of deep learning models.

2.4.3.3 Scikit-Learn:

A machine learning library for Python called Scikit-Learn is freely accessible. With a variety of algorithms for tasks including classification, regression, clustering, and dimensionality reduction, it supports both supervised and unsupervised machine learning. Built on the SciPy foundation, Scikit-learn is a Python machine-learning module available under the 3-Clause BSD license. Launched in 2007 as a Google Summer of Code project by David Cournapeau, the project has received contributions from many volunteers throughout the years. (pypi, n.d)

Figure 11 - Scikit-Learn (Pulp learning, 2021)

It was created using well-known libraries like NumPy and SciPy, and it works well with other well-used libraries like Pandas and Seaborn. (Datagy, 2022)

Furthermore, Scikit-learn's extensive toolkit and techniques for model evaluation and classification of EEG signal data make machine learning jobs simpler in this project.

2.4.3.4 Matplotlib:

Matplotlib is a great Python visualization library for two-dimensional array charts. Based on NumPy arrays, Matplotlib is a multi-platform data visualization package intended to be used with the larger SciPy stack. In the year 2002, John Hunter launched the announcement. The ability to visually access vast volumes of data in a format that is simple to understand is one of visualization's biggest advantages. Many plots, including line, bar, scatter, histogram, and others, are available in Matplotlib. (Geeksforgeeks, 2023)

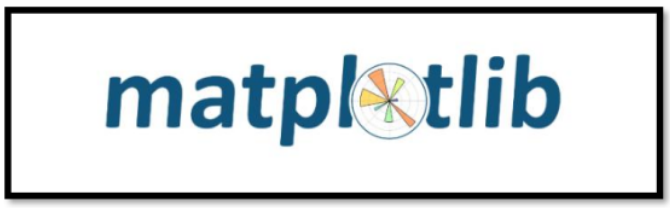


Figure 12 – Matplotlib (Pranay, 2016)

Types of Matplotlib:

- Bar graph using Matplotlib: Bar graphs can be shown either vertically or horizontally and are used to compare data and monitor changes over time. The magnitude of the value that the bar conveys is reflected in its length. (Sai Manikanth Thiyari, 2020)
- Histogram using Matplotlib: Bar graphs are used to compare two things, whereas histograms are used to visualize data distribution, especially in situations involving large lists or arrays. Bins, that indicate a range of values separated into intervals, can be used to depict the age distribution of the population in order to provide an illustration. (Sai Manikanth Thiyari, 2020)
- Scatterplot using Matplotlib: When comparing data variables and evaluating the connection between dependent and independent variables, scatter plots are the preferred method. One variable determines the location on the horizontal axis, while another determines the position on the vertical axis, resulting in a collection of points representing the information. (Sai Manikanth Thiyari, 2020)
- Area plot using Matplotlib: Area plots, often referred to as stack plots, are used to track changes over time in two or more related groups that together make up a single category. They are like line plots. (Sai Manikanth Thiyari, 2020)
- Pie chart using Matplotlib: A pie chart is a circular graphic that has been broken into segments or slices. Each pie slice represents a different category and is used to show proportionate or percentage-based statistics. (Sai Manikanth Thiyari, 2020)

In our project, Matplotlib plays a key role in producing visuals that facilitate the interpretation and presentation of EEG signal data. It enables the creation of readable and instructive plots, including line charts and histograms, which improve our comprehension of the patterns and trends in the data.

2.4.3.5 TensorFlow:

Strong machine learning and deep learning framework TensorFlow was created by Google. Many applications, including large-scale supervised and unsupervised learning tasks, are supported by this originally numerical calculation-focused software. TensorFlow supports smooth application execution with a high-performance C++ core engine and an easy-to-use Python-based API for development. Through tasks like image recognition and natural language processing, its dynamic environment enables researchers and developers. A final version of TensorFlow was released at the beginning of 2017, and since its introduction in 2015, it has grown to be a major player in the machine learning field. (W3schools, n.d)

Figure 13 - (Simran Kaur Arora, 2023)

2.4.3.6 Keras:

Keras is an open-source Python library, works as a high-level, user-friendly API for building as well as training neural networks. It is Designed to facilitate quick experimentation and iteration, Keras operates on top of TensorFlow, lowering the learning curve for deep learning tasks. (Eleanor Thomas, 2023)

Figure 14 - Keras (Javatpoint, n.d)

The Keras library, which is developed on top of TensorFlow, acts as a user-friendly API for creating and training neural networks, supporting effective model building and testing in our project, "Efficient Detection of Human Motion Movements through EEG signals using deep learning".

2.4.4 Database Management chosen:

No Database has been selected for this project.

2.4.5 Operating system chosen:

Windows:

As a bridge between software and hardware components, operating system selection is essential to the development of a project. Because it is easy to build for and compatible with the hardware and software resources that are available, Windows—more especially, Windows 11—has been

selected as the operating system for this study. Both the programming language and the selected Integrated Development Environment (IDE) are compatible. Choosing Windows ensures a smooth working process with Windows-compatible IDEs and makes it easier to integrate popular frameworks like TensorFlow and PyTorch.

Figure 15 - Image of the Windows Logo

This selection improves usability and accessibility for prospective end users who work with EEG signal processing. The decision is supported by the Windows platform's rich documentation and active community, which facilitate a more seamless development process while effectively handling project challenges. To wrap things up, utilizing the Windows operating system seems to be a sensible choice that will promote effectiveness and ensure that the "Efficient Detection of human motion movements through EEG signals using Deep learning" project is carried out successfully.

2.4.6 Web server chosen:

No web server has been selected for this project.

2.4.7 web browser chosen:

Figure 16 - Image of the chrome logo

Chrome is an internet browser that has been utilized for many years by many people worldwide. For off-network browsing, this is the recommended browser. A free web browser created by Google; Google Chrome is used to view webpages on the internet. For its UI benefits, it is very commonly utilized by developers. Since it has the best speed, uses the least amount of RAM, and performs well overall—a reason why the majority of people use it—the researcher decided to utilize it for the development and analysis of this project. (Elise Moreau,2022)

2.5 Summary:

In this chapter, Researcher discussed seven projects closely related to our project, providing information into the domains relevant to our project. moreover, to choose a programming language for this project, development methodology, tools, and system, I strive to compare two of them in this chapter. After comparing, Researcher have decided that the Python language and operating system are the best fit for my project. In this chapter, I've also specified a few tools and libraries for my project.

CHAPTER 3: METHODOLOGY

3.1 Introduction:

The planned strategy that investigators use to guide their investigations—from defining the study topic to analyzing findings—is referred to as methodology. This systematic methodology provides systematic, thorough research with findings that are ethically correct, giving researchers a framework for conducting studies effectively and producing trustworthy results.

In practical terms, research methodology is a process of describing the techniques as well as methods for identifying and analyzing information on a particular research topic. It's a deliberate process where researchers design their study, select the best research tools, and consider crucial aspects like research design, data collection methods, and analysis techniques within an large framework. (Divya Sreekumar, 2023)

The approach consists of several important stages, such as research design, data collection, analysis, validity and reliability, and ethics. Part of the research design is determining which method, such surveys, or experiments, is most appropriate to handle the study issue. Data collection is the process of selecting the most effective ways to gather the necessary information, such as surveys or interviews.

By applying appropriate statistical or qualitative methods, data analysis evaluates the gathered information. To make sure that the results are both valid and reliable, the study's quality will be evaluated using the term's validity and reliability. Finally, ethics includes making sure that the study is carried out ethically and in accordance with ethical values, such as obtaining participants' consent and protecting their privacy.

3.2 Methodology:

3.2.1 Introduction of Methodology:

The analysis can be conducted with methodologies such as Crisp-DM, SEMMA, and KDD. These methodologies provide structured frameworks for data processing, modelling, and evaluation, offering researchers versatile methodologies to derive meaningful insights based on project requirements as well as data characteristics.

3.2.2 Methodology choice and Justification:

For this project, "Efficient Detection of Human Motion Movements through EEG Signals Using Deep Learning," I prefer the 'CRISP-DM' methodology. While KDD encompasses several steps like data selection, preprocessing, transformation, data mining, and evaluation, CRISP-DM is a popular as well as efficient data mining and analytics approach with six distinct phases: business understanding, data understanding, data preparation, modelling, evaluation, and deployment. By ensuring that the project moves through clearly defined phases—from comprehending the business challenge to implementing the model—it improves the process's overall efficacy and efficiency.

3.2.3 Describe the activities, processes, and techniques in each phase towards the chosen methodology in detail:

The open standard process framework model called Cross-Industry Standard Process for Data Mining, or CRISP-DM, is intended especially for data mining project planning. It offers an organized, generally recognized method to direct the preparation and implementation of data mining initiatives in a range of sectors. A group of business specialists created the CRISP-DM technique in the late 1990s to provide a methodical approach to data mining. It includes every crucial phase of a data mining project, from understanding the business issue to putting the model into use. (Yudha Vijaya,2021)

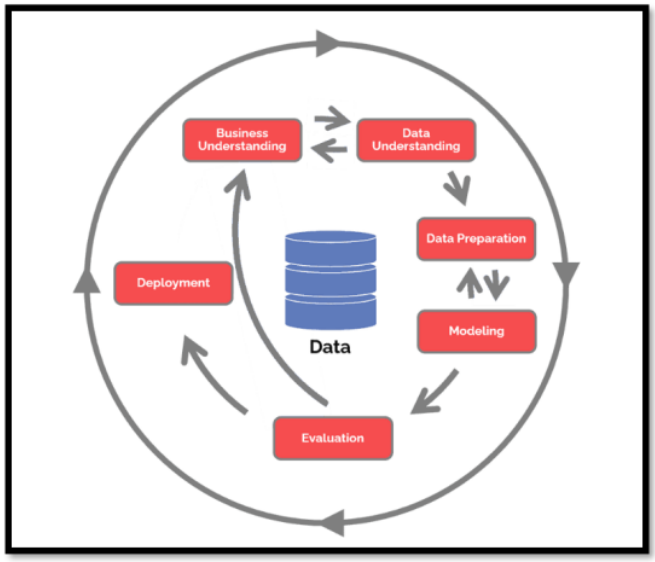


Figure 17 - CRISP DM (Nick Hotz, 2023)

The CRISP-DM methodology is divided into six stages:

- Business Understanding: Understanding the requirements, the problem at hand, and the business objectives are the main priorities during the first stage of CRISP-DM. The main objectives are to clarify the business problem, establish success criteria, and specify the project's objectives. In this phase, the issue domain, important players, and the project's scope are all thoroughly analysed. (Nick Hotz, 2023)
- Data Understanding: The goal of this phase is to gather and examine project-related data. This entails a thorough analysis of the type of data, a quality evaluation, and the detection of any potential problems or difficulties. To truly understand the structure, quality, and substance of the data, the stage includes data collection, exploration, and analysis. This phase includes essential tasks including identifying data sources, evaluating data quality, and determining possible preprocessing requirements. (Nick Hotz, 2023)
- Data Preparation: Cleaning, transforming, and formatting the gathered data are all part of the critical phase of preparing it for modeling. This comprises choosing input variables for the model, generating derived variables, and figuring out the best method for data sampling. The aim is to make sure that the data is well-structured and polished for efficient analysis. (Nick Hotz, 2023)

- Modelling: The prepared data is subjected to a variety of modelling approaches, such as statistical models or machine learning algorithms, during the modelling phase. Using the selected methodologies, models are created at this step, and their performance is evaluated using validation procedures. The goal is to use the provided data to create prediction models that work well. (Nick Hotz, 2023)
- Evaluation: Models are evaluated after they are built to make sure they are in accordance with company objectives. In this step, model performance is evaluated and the best models for the issue are determined. It includes looking over the procedure and choosing the next course of action depending on the findings of the evaluation.

Deployment: During the deployment phase, a strategy for implementation, integration of methodologies, and monitoring of model efficiency are all part of the decision-making process which includes the conceptual framework. The final phases concentrate on implementing selected models into operations, which include integration, continuous monitoring, and making sure that business problems are resolved successfully. This includes creating a final report, reviewing the project, and organizing deployment, monitoring, and maintenance. (Nick Hotz, 2023).

3.3 Summary:

"The methodology chapter describes the use of the Crisp-DM methodology for the project 'Efficient detection of human motion movements through EEG signals using deep learning.' Crisp-DM is chosen for its structured approach, aligning with the goal of employing deep learning techniques for motion detection from EEG signals. The chapter details how Crisp-DM's phases are systematically implemented for comprehensive analysis to meet project objectives."

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1 Introduction:

Efficient Detection of Human Motion Movements through EEG signals using Deep Learning" presents a structured methodology focused on accurately categorizing human motion patterns based on EEG signals. This chapter showcases the crucial stages involved in the process, starting with data collection from reputable sources such as Kaggle. The collected data undergoes rigorous pre-processing to ensure reliability by eliminating noise and outliers. Various deep learning architectures are then employed to construct models capable of

extracting relevant features from EEG signals. These models are trained using prepared datasets to establish associations between input signals as well as corresponding movements. Performance evaluation of the models uses key metrics including accuracy and precision.By following this systematic methods, the classification system provides valuable insights into human behaviour, with potential applications spanning across domains like healthcare as well as sports analysis.

4.2 Data Collection:

Researchers used the BCI Competition IV Dataset 2a, a worldwide dataset containing EEG data from nine people (Subjects), for this research. This dataset is in the (.mat) format and was downloaded from Kaggle (Link: 'https://www.kaggle.com/datasets/kgreeshmalakshmi/eeg-based-bci-four-class-wheelchair-application'). Moreover, Researcher used the SciPy load mat function to import these datasets into the PyCharm IDE. Four motor imaging tasks—imagining movement for the left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4)—are components of the cue-based BCI paradigm included in the dataset. Each subject took part in two sessions, each of which was recorded on a different day and consisted of six runs punctuated with small rests. There are 48 trials in a single run (12 trials for each of the four potential classes), for a total of 288 trials in a session.

The data is collected at sampling frequency of 250 Hz. They are bandpass filtered between 0.5 Hz to 100 Hz with the 50 Hz notch filter enabled. The dataset has 25 features in which 22 features are related to EEG signals and 3 features are related to EOG signals which record eye movements.

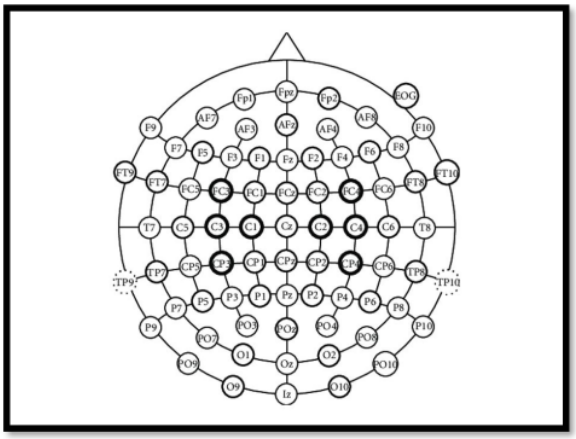
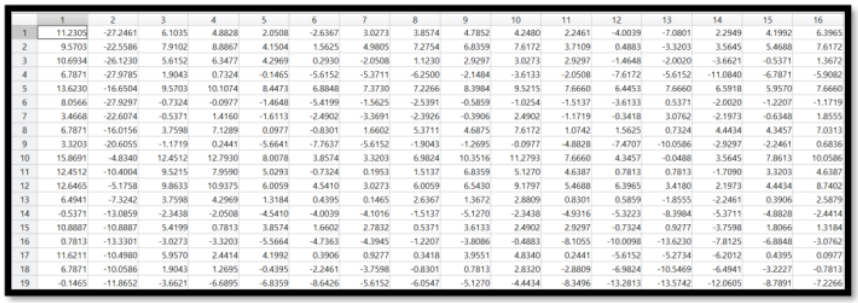


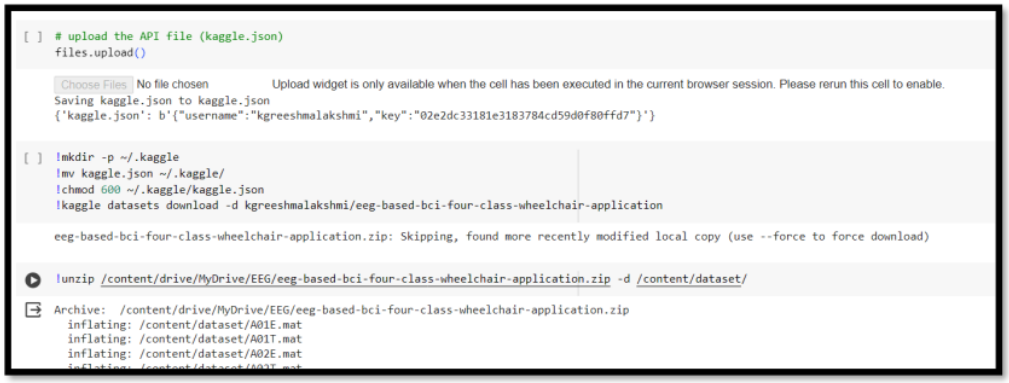
Figure 18

The EEG signals in this dataset are recorded using 22 electrodes positioned on the scalp. The specific electrode locations are as follows: Fz, FC3, FC1, FCz, FC2, FC4, C5, C3, C1, Cz, C2, C4, C6, CP3, CP1, CPz, CP2, CP4, Pz. These electrodes are labelled based on their scalp location, where "F" stands for frontal, "C" for central, "P" for parietal, and "z" for the midline. Odd-numbered electrodes (e.g., C3, CP3) are on the left side of the head, while even-numbered electrodes (e.g., C4, CP4) are on the right side. The number indicates the distance from the midline, with lower numbers closer to the midline and higher numbers farther away.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	11.2305	-27.2461	6.1035	4.8828	2.0508	-2.6367	3.0273	3.8574	4.7852	4.2480	2.2461	-4.0039	-7.0801	2.2949	4.1992	6.3965
2	9.5703	-22.5586	7.9102	8.8867	4.1504	1.5625	4.9805	7.2754	6.8359	7.6172	3.7109	0.4883	-3.3203	3.5645	5.4688	7.6172
3	10.6934	-26.1230	5.6152	6.3477	4.2969	0.2930	-2.0508	1.1230	2.9297	3.0273	2.9297	-1.4648	-2.0020	-3.6621	-0.5371	1.3672
4	6.7871	-27.9785	1.9043	0.7324	-0.1465	-5.6152	-5.3711	-6.2500	-2.1484	-3.6133	-2.0508	-7.6172	-5.6152	-11.0840	-6.7871	-5.9082
5	13.6230	-16.6504	9.5703	10.1074	8.4473	6.8848	7.3730	7.2266	8.3984	9.5215	7.6660	6.4453	7.6660	6.5918	5.9570	7.6660
6	8.0566	-27.9297	-0.7324	-0.0977	-1.4648	-5.4199	-1.5625	-2.5391	-0.5859	-1.0254	-1.5137	-3.6133	0.5371	-2.0020	-1.2207	-1.1719
7	3.4668	-22.6074	-0.5371	1.4160	-1.6113	-2.4902	-3.3691	-2.3926	-0.3906	2.4902	-1.1719	-0.3418	3.0762	-2.1973	-0.6348	1.8555
8	6.7871	-16.0156	3.7598	7.1289	0.0977	-0.8301	1.6602	5.3711	4.6875	7.6172	1.0742	1.5625	0.7324	4.4434	4.3457	7.0313
9	3.3203	-20.6055	-1.1719	0.2441	-5.6641	-7.7637	-5.6152	-1.9043	-1.2695	-0.0977	-4.8828	-7.4707	-10.0586	-2.9297	-2.2461	0.6836
10	15.8691	-4.8340	12.4512	12.7930	8.0078	3.8574	3.3203	6.9824	10.3516	11.2793	7.6660	4.3457	-0.0488	3.5645	7.8613	10.0586
11	12.4512	-10.4004	9.5215	7.9590	5.0293	-0.7324	0.1953	1.5137	6.8359	5.1270	4.6387	0.7813	0.7813	-1.7090	3.3203	4.6387
12	12.6465	-5.1758	9.8633	10.9375	6.0059	4.5410	3.0273	6.0059	6.5430	9.1797	5.4688	6.3965	3.4180	2.1973	4.4434	8.7402
13	6.4941	-7.3242	3.7598	4.2969	1.3184	0.4395	0.1465	2.6367	1.3672	2.8809	0.8301	0.5859	-1.8555	-2.2461	0.3906	2.5879
14	-0.5371	-13.0859	-2.3438	-2.0508	-4.5410	-4.0039	-4.1016	-1.5137	-5.1270	-2.3438	-4.9316	-5.3223	-8.3984	-5.3711	-4.8828	-2.4414
15	10.8887	-10.8887	5.4199	0.7813	3.8574	1.6602	2.7832	0.5371	3.6133	2.4902	2.9297	-0.7324	0.9277	-3.7598	1.8066	1.3184
16	0.7813	-13.3301	-3.0273	-3.3203	-5.5664	-4.7363	-4.3945	-1.2207	-3.8086	-0.4883	-8.1055	-10.0098	-13.6230	-7.8125	-6.8848	-3.0762
17	11.6211	-10.4980	5.9570	2.4414	4.1992	0.3906	0.9277	0.3418	3.9551	4.8340	0.2441	-5.6152	-5.2734	-6.2012	0.4395	0.0977
18	6.7871	-10.0586	1.9043	1.2695	-0.4395	-2.2461	-3.7598	-0.8301	0.7813	2.8320	-2.8809	-6.9824	-10.5469	-6.4941	-3.2227	-0.7813
19	-0.1465	-11.8652	-3.6621	-6.6895	-6.8359	-8.6426	-5.6152	-6.0547	-5.1270	-4.4434	-8.3496	-13.2813	-13.5742	-12.0605	-8.7891	-7.2266

Figure 19 - Values inside the dataset of subject-1.



```
[ ] # upload the API file (kaggle.json)
    files.upload()

    Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
    Saving kaggle.json to kaggle.json
    {'kaggle.json': b'{"username":"kgreeshmalakshmi","key":"02e2dc33181e3183784cd59d0f80ffd7"}'}

[ ] !mkdir -p ~/.kaggle
    !mv kaggle.json ~/.kaggle/
    !chmod 600 ~/.kaggle/kaggle.json
    !kaggle datasets download -d kgreeshmalakshmi/eeg-based-bci-four-class-wheelchair-application

    eeg-based-bci-four-class-wheelchair-application.zip: Skipping, found more recently modified local copy (use --force to force download)

    !unzip /content/drive/MyDrive/EEG/eeg-based-bci-four-class-wheelchair-application.zip -d /content/dataset/

    Archive:  /content/drive/MyDrive/EEG/eeg-based-bci-four-class-wheelchair-application.zip
      inflating: /content/dataset/A01E.mat
      inflating: /content/dataset/A01T.mat
      inflating: /content/dataset/A02E.mat
```

Figure 20 - Code snippet to include the dataset.

Dataset is loaded into colab using Kaggle API. Loading datasets into Google Colab via the Kaggle API simplifies the process of accessing Kaggle datasets for analysis and machine learning as shown in the figure 20. After installing the Kaggle API in Colab, users upload their Kaggle API token obtained from the Kaggle website. With the token in place, datasets can be directly downloaded into Colab using the **!kaggle datasets download** command, specifying the dataset's username and name. The downloaded dataset is typically in zip format and can be unzipped using Python's **zipfile** module. This streamlined process eliminates the need for

manual downloading and uploading of datasets, enhancing efficiency in data analysis and model development tasks within the Colab environment.

```python
def get_data(path, subject, classes_labels='all', LOSO=False, isStandard=True, isShuffle=True):
    # Load training and testing data
    X_train, y_train = load_kaggle_data(path, subject + 1, True)
    X_test, y_test = load_kaggle_data(path, subject + 1, False)

    # Shuffle the data if required
    if isShuffle:
        X_train, y_train = shuffle(X_train, y_train, random_state=42)
        X_test, y_test = shuffle(X_test, y_test, random_state=42)

    # Reshape data for CNN
    N_tr, N_ch, T = X_train.shape
    X_train = X_train.reshape(N_tr, 1, N_ch, T)
    y_train_onehot = to_categorical(y_train)

    N_tr, N_ch, T = X_test.shape
    X_test = X_test.reshape(N_tr, 1, N_ch, T)
    y_test_onehot = to_categorical(y_test)

    # Standardize the data if required
    if isStandard:
        X_train, X_test = standardize_data(X_train, X_test, N_ch)

    return X_train, y_train, y_train_onehot, X_test, y_test, y_test_onehot
```

Figure 21 - Code snippet of the function to load the dataset.

After obtaining the datasets from Kaggle using the Kaggle API in Google Colab, the next step includes loading the MATLAB (.mat) files for further processing. Notably, the dataset includes separate files designated for training and testing purposes, hence eliminating the necessity for data splitting. To streamline this process, a function called get_data() is developed to facilitate the loading of both training and testing data. Once the data is loaded, it proceeds to the data pre-processing stage.

4.3 Data Preprocessing:

The data in usage has already been pre-processed using a bandpass filter with the 50 Hz notch filter enabled, filtered between 0.5 Hz and 100 Hz.

Generally, data pre-processing includes data cleaning, data transformation as well as data reduction. Data cleaning accounts for outliers removal, missing value treatment, conversion of features into appropriate format. Transforming data into a format suitable for analysis is known as data transformation. Normalization, standardization, and discretization are common methods used in data transformation. Reducing the dataset while maintaining all relevant information is known as data reduction. Techniques including feature selection, feature

extraction, sampling, and clustering can be used to reduce data. (Geeksforgeeks, 2023). (Geeksforgeeks, 2023).

In EEG processing, only data standardization is usually required, as other pre-processing steps such as noise reduction and outlier removal are not typically applicable to EEG data. Standardization assures that the features (i.e., EEG signals) are on a similar scale, preventing any feature from controlling the learning process.

```
[ ]  def standardize_data(X_train, X_test, channels):
         for j in range(channels):
               scaler = StandardScaler()
               scaler.fit(X_train[:, 0, j, :])
               X_train[:, 0, j, :] = scaler.transform(X_train[:, 0, j, :])
               X_test[:, 0, j, :] = scaler.transform(X_test[:, 0, j, :])

         return X_train, X_test
```

Figure 22 - Code snippet for the standardization function.

To standardize the data for EEG processing, a function called standardize_data() is developed as shown in figure22. This function utilizes the StandardScaler() function from the scikit-learn package, a widely-used library for machine learning in Python. The StandardScaler() function standardizes the data by scaling each feature to have a mean of 0 and a standard deviation of 1, assuring stability in feature scales across the dataset. (Lari Giba,2024)

4.4 Data Understanding:

In the dataset used for Effective Efficient Detection of Human Motion Movements, the data consists of 25 numeric features representing voltage values. Conventional plots such as histograms, scatter plots, and bar graphs might not be able to provide meaningful insights or help in accurately comprehending the dataset due to the nature of the data. Since the features are numeric and signify voltage values, these plots might not accurately visualize the relationships or patterns within the data. Furthermore, the next step involves modelling the data, where techniques such as machine learning algorithms are used to extract insights and detect human motion movements effectively.

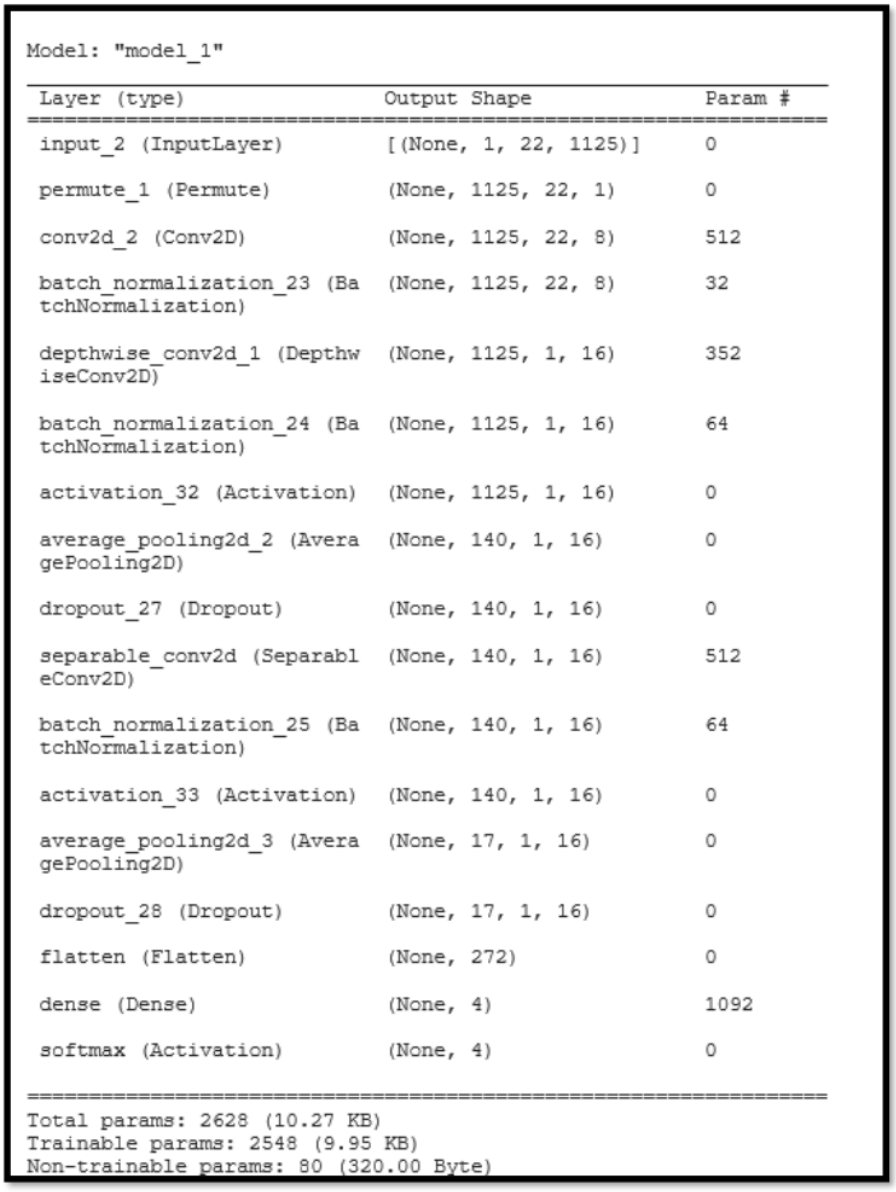
4.5 Model Building:

The models that researchers have utilized to this project are as follows:

1. EEGNet
2. DeepConvNet
3. ShallowConvNet
4. EEG-TCNet
5. ATCNet

4.5.1 EEGNet:

Electroencephalography (EEG) Ensemble Convolutional Neural Network, or EEGNet for short, is a convolutional neural network (CNN) structure designed specifically for EEG-based brain-computer interfaces (BCIs). It is an optimized and cost-effective CNN structure. Because of the design's use of temporal, depth wise, and separable convolution, the model can understand frequency filters as well as obtain filters for space that are particular to certain frequencies.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 1, 22, 1125)]     0

 permute_1 (Permute)         (None, 1125, 22, 1)       0

 conv2d_2 (Conv2D)           (None, 1125, 22, 8)       512

 batch_normalization_23 (Ba  (None, 1125, 22, 8)       32
 tchNormalization)

 depthwise_conv2d_1 (Depthw  (None, 1125, 1, 16)       352
 iseConv2D)

 batch_normalization_24 (Ba  (None, 1125, 1, 16)       64
 tchNormalization)

 activation_32 (Activation)  (None, 1125, 1, 16)       0

 average_pooling2d_2 (Avera  (None, 140, 1, 16)        0
 gePooling2D)

 dropout_27 (Dropout)        (None, 140, 1, 16)        0

 separable_conv2d (Separabl  (None, 140, 1, 16)        512
 eConv2D)

 batch_normalization_25 (Ba  (None, 140, 1, 16)        64
 tchNormalization)

 activation_33 (Activation)  (None, 140, 1, 16)        0

 average_pooling2d_3 (Avera  (None, 17, 1, 16)         0
 gePooling2D)

 dropout_28 (Dropout)        (None, 17, 1, 16)         0

 flatten (Flatten)           (None, 272)               0

 dense (Dense)               (None, 4)                 1092

 softmax (Activation)        (None, 4)                 0

=================================================================
Total params: 2628 (10.27 KB)
Trainable params: 2548 (9.95 KB)
Non-trainable params: 80 (320.00 Byte)
```

Figure 23 - Screenshot of the architecture for the model EEGNet in text format.

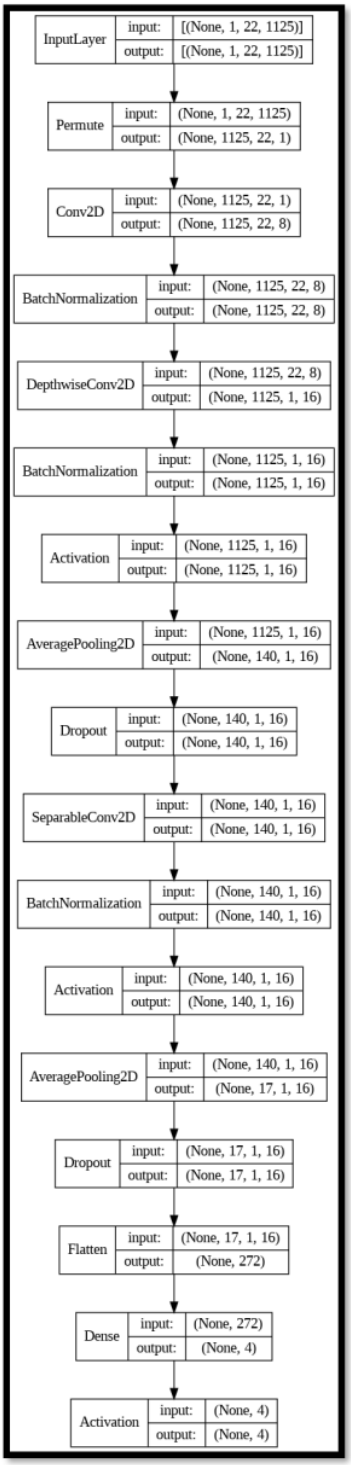


Figure 24 - Screenshot of the architecture for the model EEGNet in diagram format.

The input, convolutional, batch normalization, depth wise convolutional, activation, average pooling, dropout, separable convolutional, flatten, dense, and SoftMax layers are among the layers that make up the "EEGNet" model. The model has a total of 2628 parameters, out of which 2548 parameters are trainable, and the remaining 80 parameters are non-trainable. This data gives insights into the architecture including complexity of the model, helping in understanding its structure as well as behaviour during training and inference processes.

```
def EEGNet_classifier(n_classes, Chans=22, Samples=1125, F1=8, D=2, kernLength=64, dropout_eeg=0.25):
    input1 = Input(shape = (1,Chans, Samples))
    input2 = Permute((3,2,1))(input1)
    regRate=.25

    eegnet = EEGNet(input_layer=input2, F1=F1, kernLength=kernLength, D=D, Chans=Chans, dropout=dropout_eeg)
    eegnet = Flatten()(eegnet)
    dense = Dense(n_classes, name = 'dense',kernel_constraint = max_norm(regRate))(eegnet)
    softmax = Activation('softmax', name = 'softmax')(dense)

    return Model(inputs=input1, outputs=softmax)

def EEGNet(input_layer, F1=8, kernLength=64, D=2, Chans=22, dropout=0.25):

    F2= F1*D
    block1 = Conv2D(F1, (kernLength, 1), padding = 'same',data_format='channels_last',use_bias = False)(input_layer)
    block1 = BatchNormalization(axis = -1)(block1)
    block2 = DepthwiseConv2D((1, Chans), use_bias = False,
                             depth_multiplier = D,
                             data_format='channels_last',
                             depthwise_constraint = max_norm(1.))(block1)
    block2 = BatchNormalization(axis = -1)(block2)
    block2 = Activation('elu')(block2)
    block2 = AveragePooling2D((8,1),data_format='channels_last')(block2)
    block2 = Dropout(dropout)(block2)
    block3 = SeparableConv2D(F2, (16, 1),
                             data_format='channels_last',
                             use_bias = False, padding = 'same')(block2)
    block3 = BatchNormalization(axis = -1)(block3)
    block3 = Activation('elu')(block3)
    block3 = AveragePooling2D((8,1),data_format='channels_last')(block3)
    block3 = Dropout(dropout)(block3)
    return block3
```

Figure 25 - Code snippet for building the model EEGNet.

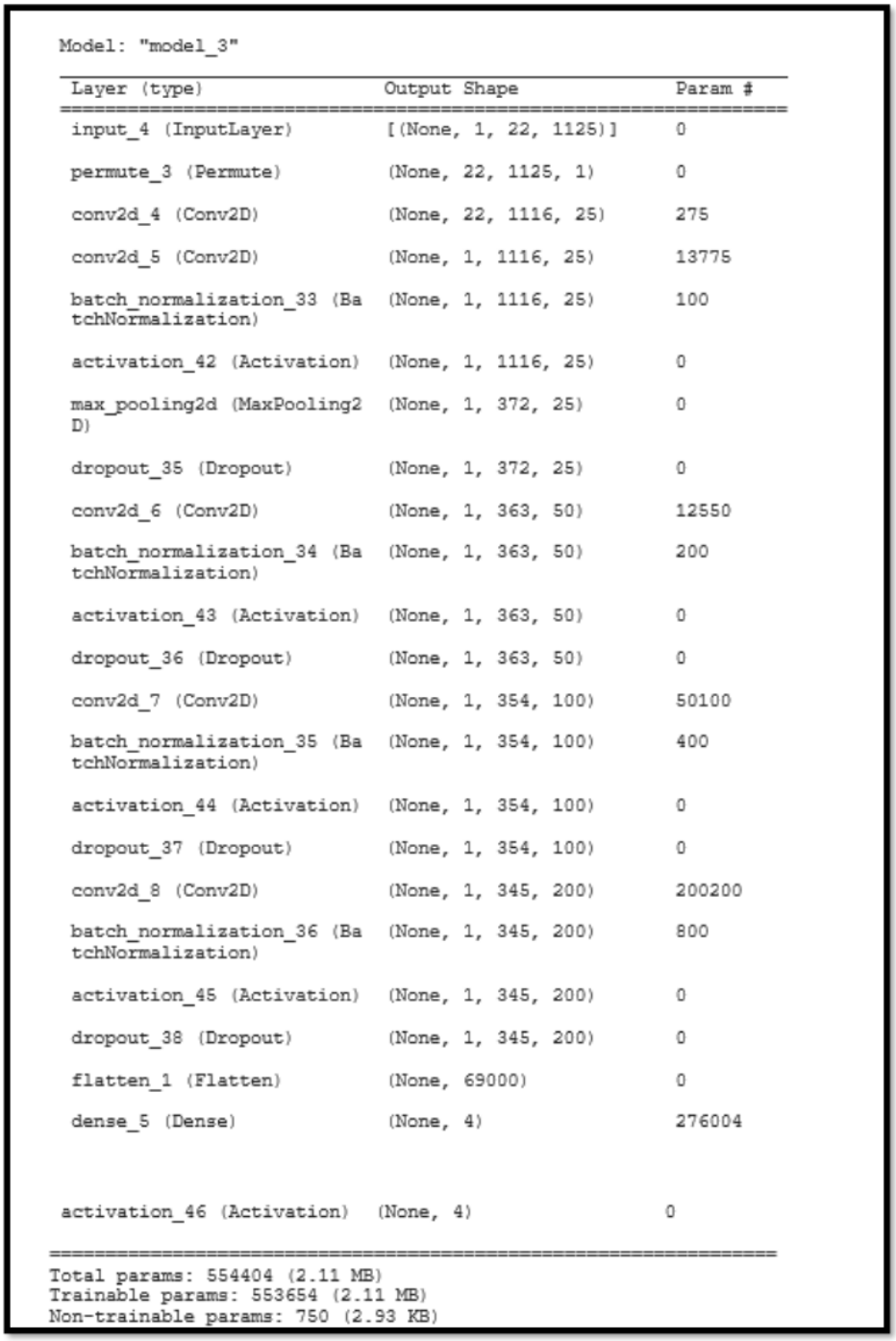
The EEGNet_classifier function creates a classification model for EEG signal analysis using the EEGNet architecture, as illustrated in the figure above. It takes parameters such as the number of classes, EEG channel count, sample length, and EEGNet-specific parameters for feature extraction. With convolutional and separable convolutional layers, batch normalization, ELU activation, average pooling, and dropout regularization for feature learning and overfitting prevention, the EEGNet function specifies the EEGNet architecture. The output is processed through a dense layer with softmax activation for classification. These functions encapsulate the essential steps for building a strong EEGNet-based classifier included for EEG signal classification tasks.

The model is trained using several parameters to optimize the training process of the EEGNet model. These parameters consist of 64 samples in a batch, training for 500 epochs with early stopping patience set to 100 epochs, including a learning rate of 0.001. The training is performed in a single session. Furthermore, the configuration allows visualization of learning curves during training and specifies that the model's output probabilities should be transformed into class probabilities utilizing softmax activation.

After the training is done, validation is carried out. For the EEGNet model, the validation performance, measured in accuracy percentage, varies across different subjects, ranging from 50.00% to 94.83% across nine subjects. 76.82% accuracy is the mean for all participants and seeds in the model. Furthermore, 11.3 minutes is the average training time for all seeds.

4.5.2 DeepConvNet:

An artificial neural network called the deep ConvNet model uses convolutions to identify local patterns in data. It is particularly good at taking raw input and extracting local, low-level data, and in subsequent layers, it gradually captures more global, high-level data. Multiple convolutional layers are a common component of DeepConvNet architectures, which allow for the recognition of higher-level visual elements from raw pictures, including edges, basic forms, and entire objects. This method has shown to be highly successful in a variety of applications, such as voice recognition and computer vision, frequently outperforming earlier state-of-the-art techniques. (Vernon J | Lawhern | Amelia J.Solon, 2018).



```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 1, 22, 1125)]     0

 permute_3 (Permute)         (None, 22, 1125, 1)       0

 conv2d_4 (Conv2D)           (None, 22, 1116, 25)      275

 conv2d_5 (Conv2D)           (None, 1, 1116, 25)       13775

 batch_normalization_33 (Ba  (None, 1, 1116, 25)       100
 tchNormalization)

 activation_42 (Activation)  (None, 1, 1116, 25)       0

 max_pooling2d (MaxPooling2  (None, 1, 372, 25)        0
 D)

 dropout_35 (Dropout)        (None, 1, 372, 25)        0

 conv2d_6 (Conv2D)           (None, 1, 363, 50)        12550

 batch_normalization_34 (Ba  (None, 1, 363, 50)        200
 tchNormalization)

 activation_43 (Activation)  (None, 1, 363, 50)        0

 dropout_36 (Dropout)        (None, 1, 363, 50)        0

 conv2d_7 (Conv2D)           (None, 1, 354, 100)       50100

 batch_normalization_35 (Ba  (None, 1, 354, 100)       400
 tchNormalization)

 activation_44 (Activation)  (None, 1, 354, 100)       0

 dropout_37 (Dropout)        (None, 1, 354, 100)       0

 conv2d_8 (Conv2D)           (None, 1, 345, 200)       200200

 batch_normalization_36 (Ba  (None, 1, 345, 200)       800
 tchNormalization)

 activation_45 (Activation)  (None, 1, 345, 200)       0

 dropout_38 (Dropout)        (None, 1, 345, 200)       0

 flatten_1 (Flatten)         (None, 69000)             0

 dense_5 (Dense)             (None, 4)                 276004


 activation_46 (Activation)  (None, 4)                 0

=================================================================
Total params: 554404 (2.11 MB)
Trainable params: 553654 (2.11 MB)
Non-trainable params: 750 (2.93 KB)
```

Figure 26 - Screenshot of the architecture for the model DeepConvNet in text format.

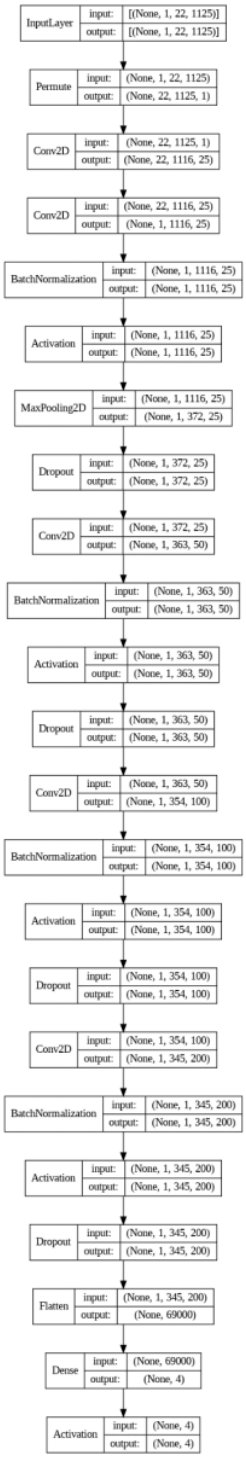


Figure 27 - Screenshot of the architecture for the model DeepConvNet in diagram format.

The DeepConvNet architecture consists of convolutional, batch normalization, activation, max pooling, dropout, and dense layers. It processes EEG data with 22 channels and 1125-time steps. The model consists multiple convolutional layers to extract hierarchical features, followed by batch normalization and activation layers for stabilization and non-linearity. Max pooling layers downsample feature maps, while dropout layers prevent overfitting. A dense output layer with softmax activation produces class probabilities. The model has 554,404 parameters, of which 553,654 are trainable.

```
def DeepConvNet(nb_classes, Chans = 64, Samples = 256,
                dropoutRate = 0.5):

    # start the model
    # input_main   = Input((Chans, Samples, 1))
    input_main   = Input((1, Chans, Samples))
    input_2 = Permute((2,3,1))(input_main)

    block1       = Conv2D(25, (1, 10),
                          input_shape=(Chans, Samples, 1),
                          kernel_constraint = max_norm(2., axis=(0,1,2)))(input_2)
    block1       = Conv2D(25, (Chans, 1),
                          kernel_constraint = max_norm(2., axis=(0,1,2)))(block1)
    block1       = BatchNormalization(epsilon=1e-05, momentum=0.9)(block1)
    block1       = Activation('elu')(block1)
    block1       = MaxPooling2D(pool_size=(1, 3), strides=(1, 3))(block1)
    block1       = Dropout(dropoutRate)(block1)

    block2       = Conv2D(50, (1, 10),
                          kernel_constraint = max_norm(2., axis=(0,1,2)))(block1)
    block2       = BatchNormalization(epsilon=1e-05, momentum=0.9)(block2)
    block2       = Activation('elu')(block2)
    block1       = MaxPooling2D(pool_size=(1, 3), strides=(1, 3))(block1)
    block2       = Dropout(dropoutRate)(block2)

    block3       = Conv2D(100, (1, 10),
                          kernel_constraint = max_norm(2., axis=(0,1,2)))(block2)
    block3       = BatchNormalization(epsilon=1e-05, momentum=0.9)(block3)
    block3       = Activation('elu')(block3)
    block1       = MaxPooling2D(pool_size=(1, 3), strides=(1, 3))(block1)
    block3       = Dropout(dropoutRate)(block3)

    block4       = Conv2D(200, (1, 10),
                          kernel_constraint = max_norm(2., axis=(0,1,2)))(block3)
    block4       = BatchNormalization(epsilon=1e-05, momentum=0.9)(block4)
    block4       = Activation('elu')(block4)

    block1       = MaxPooling2D(pool_size=(1, 3), strides=(1, 3))(block1)
    block4       = Dropout(dropoutRate)(block4)

    flatten      = Flatten()(block4)

    dense        = Dense(nb_classes, kernel_constraint = max_norm(0.5))(flatten)
    softmax      = Activation('softmax')(dense)

    return Model(inputs=input_main, outputs=softmax)
```

Figure 28 - Code snippet for building the model DeepConvNet

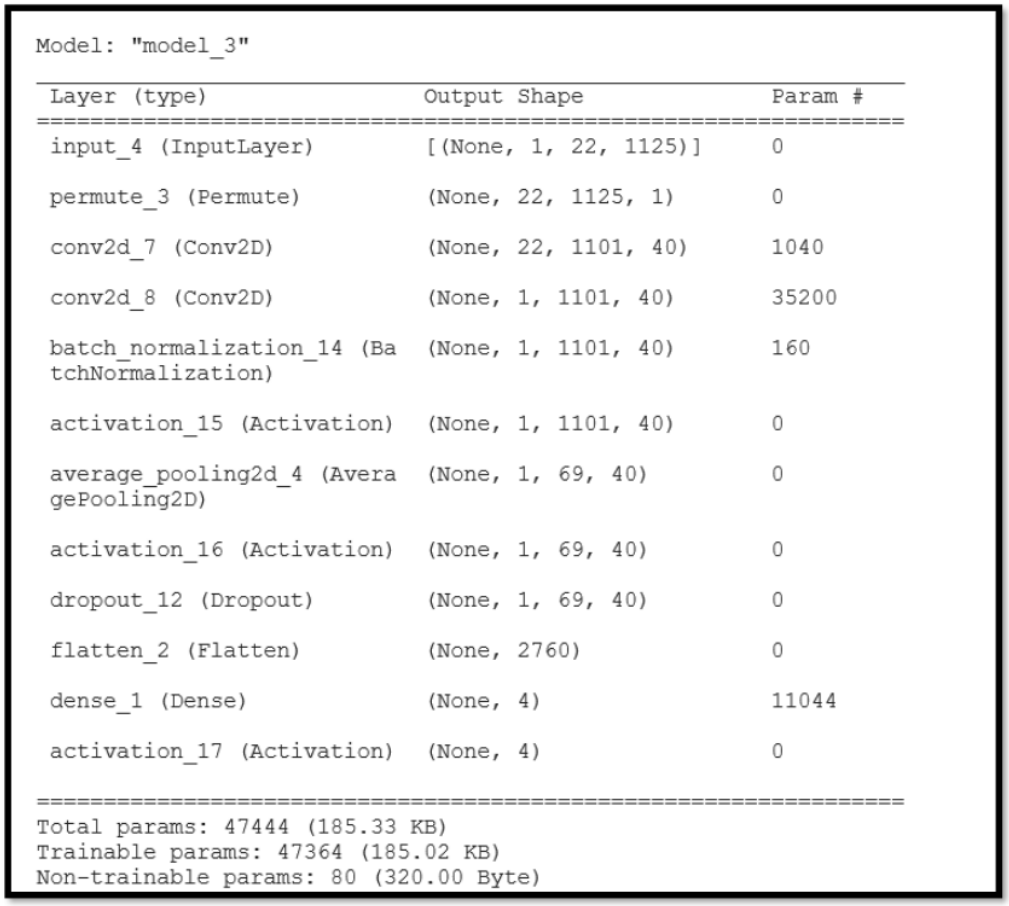
The DeepConvNet function builds a deep convolutional neural network (CNN) model optimized for EEG signal classification, as shown in the figure above. It consists multiple

convolutional blocks with increasing filter sizes, followed by batch normalization, ELU activation, max-pooling, and dropout layers for feature extraction as well as regularization. Max-norm constraints are applied to control weight magnitudes during training. The final dense layer with softmax activation allows multi-class classification, making the model appropriate for tasks requiring precise EEG signal analysis and classification across different classes.

After setting the training parameters for DeepConvNet model, which are same as EEGNet model, training is done. The model's validation performance varies depending on the subject; among nine subjects, the accuracies range from 18.97% to 79.31%. On average, the model achieves a validation accuracy of 46.17% across all subjects and seeds. Furthermore, the average training time across all seeds is 19.2 minutes.

4.5.3 ShallowConvNet:

The Filter Bank Common Spatial Patterns (FBCSP) pipeline, which was created especially for decoding band power characteristics from raw EEG data, serves as a precedent for the shallow ConvNet architecture. It is like the changes observed in FBCSP; it includes a logarithmic activation function, mean pooling, squaring nonlinearity, spatial filtering, and temporal convolution. compared to FBCSP, the shallow ConvNet combines every computational step into a single network, making it possible to optimize every step at once. Furthermore, it has been shown that incorporating numerous pooling areas within a single trial improves classification since it enables the ShallowConvNet to accurately record the temporal structure of band power variations throughout the trial. (Robin Tibor | Jost Tobias | Lukas, 2017).



```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 1, 22, 1125)]     0

 permute_3 (Permute)         (None, 22, 1125, 1)       0

 conv2d_7 (Conv2D)           (None, 22, 1101, 40)      1040

 conv2d_8 (Conv2D)           (None, 1, 1101, 40)       35200

 batch_normalization_14 (Ba  (None, 1, 1101, 40)       160
 tchNormalization)

 activation_15 (Activation)  (None, 1, 1101, 40)       0

 average_pooling2d_4 (Avera  (None, 1, 69, 40)         0
 gePooling2D)

 activation_16 (Activation)  (None, 1, 69, 40)         0

 dropout_12 (Dropout)        (None, 1, 69, 40)         0

 flatten_2 (Flatten)         (None, 2760)              0

 dense_1 (Dense)             (None, 4)                 11044

 activation_17 (Activation)  (None, 4)                 0

=================================================================
Total params: 47444 (185.33 KB)
Trainable params: 47364 (185.02 KB)
Non-trainable params: 80 (320.00 Byte)
```

Figure 29 - Screenshot of the architecture for the model ShallowConvNet in text format.

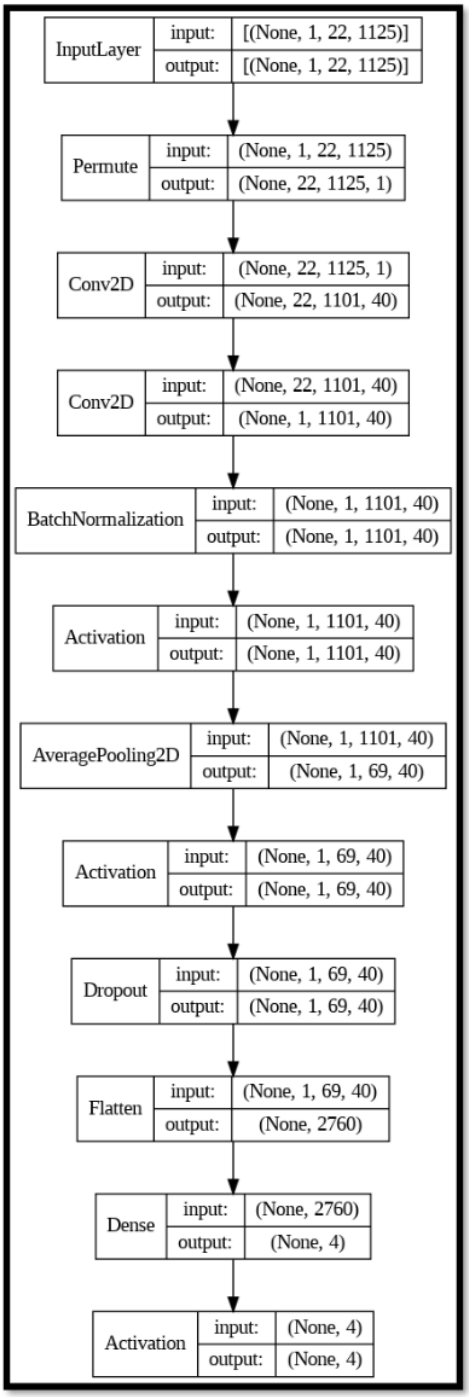


Figure 30 - Screenshot of the architecture for the model ShallowConvNet in diagram format.

The architecture of the ShallowConvNet model, denoted as "model_3," is designed for processing EEG data as shown in the above figure. It is made up of two convolutional layers that work together to extract important characteristics from the input data. To ensure stability during training, a batch normalization layer is incorporated. Activation layers incorporate non-linearities to enhance the model's expressive power. Moreover, an average pooling layer downsamples the feature maps, lowering computational complexity. Dropout regularization is employed to prevent overfitting, while a flatten layer reshapes the output for compatibility with the subsequent dense layer, which makes the final classification. Lastly, this model configuration, designed for EEG data with 22 channels and 1125 time steps, contains a total of 47,444 parameters, with 47,364 being trainable.

```
338
339 #%% need these for ShallowConvNet
340 def square(x):
341     return K.square(x)
342
343 def log(x):
344     return K.log(K.clip(x, min_value = 1e-7, max_value = 10000))
345
346
347 def ShallowConvNet(nb_classes, Chans = 64, Samples = 128, dropoutRate = 0.5):
348
349     # start the model
350     # input_main   = Input((Chans, Samples, 1))
351     input_main   = Input((1, Chans, Samples))
352     input_2 = Permute((2,3,1))(input_main)
353
354     block1       = Conv2D(40, (1, 25),
355                           input_shape=(Chans, Samples, 1),
356                           kernel_constraint = max_norm(2., axis=(0,1,2)))(input_2)
357     block1       = Conv2D(40, (Chans, 1), use_bias=False,
358                           kernel_constraint = max_norm(2., axis=(0,1,2)))(block1)
359     block1       = BatchNormalization(epsilon=1e-05, momentum=0.9)(block1)
360     block1       = Activation(square)(block1)
361     block1       = AveragePooling2D(pool_size=(1, 75), strides=(1, 15))(block1)
362     block1       = Activation(log)(block1)
363     block1       = Dropout(dropoutRate)(block1)
364     flatten      = Flatten()(block1)
365     dense        = Dense(nb_classes, kernel_constraint = max_norm(0.5))(flatten)
366     softmax      = Activation('softmax')(dense)
367
368     return Model(inputs=input_main, outputs=softmax)
```

Figure 31 - Code snippet for building the model ShallowConvNet.

The ShallowConvNet function builds a shallow convolutional neural network (CNN) model for EEG signal classification as shown in the above figure. Convolutional layers with certain filter sizes are used, and batch normalization is then used for stability. Activation functions that improve feature representation are logarithmic and square. Max-norm constraints regulate convolutional kernel magnitudes during training. Average pooling and dropout layers aid in downsampling and regularization. The model ends with a dense layer using softmax activation

for multi-class classification, making it suitable for precise EEG signal analysis across different classes.

After the model is built using the above architecture, with the same training parameters as previous models, Shallow Convolutional Network is trained. The model's validation performance varies depending on the subject; for seed 1, accuracies range from 50.00% to 91.38%. On average, the model achieves a validation accuracy of 72.99% across all subjects and seeds. The training process across all seeds collectively requires approximately 20.7 minutes.

4.5.4 EEG-TCNet:

With just a few trainable parameters, the novel temporal convolutional network (TCN) EEG-TCNET demonstrates impressive accuracy. EEG-TCNet's architecture places a strong emphasis on capturing the temporal dependencies observed in EEG data, which are crucial for understanding the brain's dynamic patterns across time. (Thorir Mar | Michael |Wang, 2020).

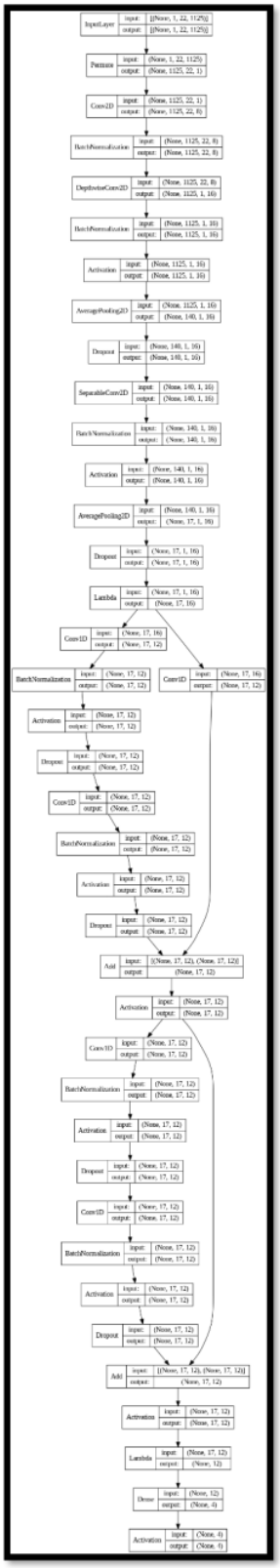

InputLayer
input: [(None, 1, 22, 1125)]
output: [(None, 1, 22, 1125)]
Permute
input: (None, 1, 22, 1125)
output: (None, 1125, 22, 1)
Conv2D
input: (None, 1125, 22, 1)
output: (None, 1125, 22, 8)
BatchNormalization
input: (None, 1125, 22, 8)
output: (None, 1125, 22, 8)
DepthwiseConv2D
input: (None, 1125, 22, 8)
output: (None, 1125, 1, 16)
BatchNormalization
input: (None, 1125, 1, 16)
output: (None, 1125, 1, 16)
Activation
input: (None, 1125, 1, 16)
output: (None, 1125, 1, 16)
AveragePooling2D
input: (None, 1125, 1, 16)
output: (None, 140, 1, 16)
Dropout
input: (None, 140, 1, 16)
output: (None, 140, 1, 16)
SeparableConv2D
input: (None, 140, 1, 16)
output: (None, 140, 1, 16)
BatchNormalization
input: (None, 140, 1, 16)
output: (None, 140, 1, 16)
Activation
input: (None, 140, 1, 16)
output: (None, 140, 1, 16)
AveragePooling2D
input: (None, 140, 1, 16)
output: (None, 17, 1, 16)
Dropout
input: (None, 17, 1, 16)
output: (None, 17, 1, 16)
Lambda
input: (None, 17, 1, 16)
output: (None, 17, 16)
Conv1D
input: (None, 17, 16)
output: (None, 17, 12)
BatchNormalization
input: (None, 17, 12)
output: (None, 17, 12)
Conv1D
input: (None, 17, 16)
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Dropout
input: (None, 17, 12)
output: (None, 17, 12)
Conv1D
input: (None, 17, 12)
output: (None, 17, 12)
BatchNormalization
input: (None, 17, 12)
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Dropout
input: (None, 17, 12)
output: (None, 17, 12)
Add
input: [(None, 17, 12), (None, 17, 12)]
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Conv1D
input: (None, 17, 12)
output: (None, 17, 12)
BatchNormalization
input: (None, 17, 12)
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Dropout
input: (None, 17, 12)
output: (None, 17, 12)
Conv1D
input: (None, 17, 12)
output: (None, 17, 12)
BatchNormalization
input: (None, 17, 12)
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Dropout
input: (None, 17, 12)
output: (None, 17, 12)
Add
input: [(None, 17, 12), (None, 17, 12)]
output: (None, 17, 12)
Activation
input: (None, 17, 12)
output: (None, 17, 12)
Lambda
input: (None, 17, 12)
output: (None, 12)
Dense
input: (None, 12)
output: (None, 4)
Activation
input: (None, 4)
output: (None, 4)

Figure 32 - Screenshot of the architecture for the model EEG-TCNet in diagram format.

Convolutional, batch normalization, activation, pooling, dropout, add, and dense layers are among the 29 layers that make up the EEG TCNet model. These layers are optimized to process EEG signals for classification, with the model designed to capture temporal dependencies and extract discriminative features effectively.

```
def TCN_block_(input_layer,input_dimension,depth,kernel_size,filters, dropout,
               weightDecay = 0.009, maxNorm = 0.6, activation='relu'):
    block = Conv1D(filters, kernel_size=kernel_size, dilation_rate=1, activation='linear',
                   kernel_regularizer=L2(weightDecay),
                   kernel_constraint = max_norm(maxNorm, axis=[0,1]),

                   padding = 'causal',kernel_initializer='he_uniform')(input_layer)
    block = BatchNormalization()(block)
    block = Activation(activation)(block)
    block = Dropout(dropout)(block)
    block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=1,activation='linear',
                   kernel_regularizer=L2(weightDecay),
                   kernel_constraint = max_norm(maxNorm, axis=[0,1]),

                   padding = 'causal',kernel_initializer='he_uniform')(block)
    block = BatchNormalization()(block)
    block = Activation(activation)(block)
    block = Dropout(dropout)(block)
    if(input_dimension != filters):
        conv = Conv1D(filters,kernel_size=1,
                   kernel_regularizer=L2(weightDecay),
                   kernel_constraint = max_norm(maxNorm, axis=[0,1]),

                   padding='same')(input_layer)
        added = Add()([block,conv])
    else:
        added = Add()([block,input_layer])
    out = Activation(activation)(added)

    for i in range(depth-1):
        block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=2**(i+1),activation='linear',
                   kernel_regularizer=L2(weightDecay),
                   kernel_constraint = max_norm(maxNorm, axis=[0,1]),

                  padding = 'causal',kernel_initializer='he_uniform')(out)
        block = BatchNormalization()(block)
        block = Activation(activation)(block)
        block = Dropout(dropout)(block)
        block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=2**(i+1),activation='linear',
                   kernel_regularizer=L2(weightDecay),
                   kernel_constraint = max_norm(maxNorm, axis=[0,1]),

                   padding = 'causal',kernel_initializer='he_uniform')(block)
        block = BatchNormalization()(block)
        block = Activation(activation)(block)
        block = Dropout(dropout)(block)
        added = Add()([block, out])
        out = Activation(activation)(added)

    return out
def EEGTCNet(n_classes, Chans=22, Samples=1125, layers=2, kernel_s=4, filt=12, dropout=0.3, activation='elu', F1=8, D=2, kernLength=32, dropout_eeg=0.2):

    input1 = Input(shape = (1,Chans, Samples))
    input2 = Permute((3,2,1))(input1)
    regRate=.25
    numFilters = F1
    F2= numFilters*D

    EEGNet_sep = EEGNet(input_layer=input2,F1=F1,kernLength=kernLength,D=D,Chans=Chans,dropout=dropout_eeg)
    block2 = Lambda(lambda x: x[:,:,-1,:])(EEGNet_sep)
    outs = TCN_block(input_layer=block2,input_dimension=F2,depth=layers,kernel_size=kernel_s,filters=filt,dropout=dropout,activation=activation)
    out = Lambda(lambda x: x[:,-1,:])(outs)
    dense       = Dense(n_classes, name = 'dense',kernel_constraint = max_norm(regRate))(out)
    softmax     = Activation('softmax', name = 'softmax')(dense)

    return Model(inputs=input1,outputs=softmax)
```

Figure 33 - Code snippet for building the model EEG-TCNet.

The code shown in the above figure defines two key functions for building a Temporal Convolutional Network (TCN) block and an EEGTCNet model for EEG signal classification. Convolutional layers, batch normalization, activation functions, dropout regularization, and residual connections are all added to a TCN block by using the TCN_block function. It takes input dimensions, depth (number of TCN layers), kernel size, number of filters, dropout rate, weight decay, max norm constraint, and activation function as parameters. The EEGTCNet function builds a deep learning model for EEG signal classification using TCN blocks and EEGNet layers. It sets up input layers, reshapes data, applies EEGNet for feature extraction, passes the output through TCN blocks, and adds a dense layer with softmax activation for classification.

The EEG-TCNet model is trained using the same training parameters as previous models after it has been constructed using the above architecture. The validation performance of the model across different subjects shows varying accuracies, ranging from 56.90% to 93.10% for individual subjects, with an average accuracy of 72.61%. Despite fluctuations, the model demonstrates overall effectiveness in classifying EEG signals. The training time for all seeds combined is 12.7 minutes, representing effective training of the model.

4.5.5 ATCNet:

The ATCNet model is an advanced deep-learning framework designed especially for motor imagery classification using EEG data. The temporal convolutional (TC) block, the attention (AT) block, and the convolutional (CV) block are its three main parts.

4.5.5.1 Convolutional Block (CV):

Three different convolutional layers—temporal, channel depth wise, and spatial convolutions—are used by the convolutional block to extract essential spatiotemporal properties from the motor imagery EEG input. Through this process, the raw EEG signal for motor imagery is converted into a condensed temporal sequence that has a more detailed representation. The only way the CV block differs from the EEGNet design is that it uses 2-D convolution rather than separable convolution like the EEGNet does.

```python
def Conv_block_(input_layer, F1=4, kernLength=64, poolSize=8, D=2, in_chans=22,
                weightDecay = 0.009, maxNorm = 0.6, dropout=0.25):

    F2= F1*D
    block1 = Conv2D(F1, (kernLength, 1), padding = 'same', data_format='channels_last',
                    kernel_regularizer=L2(weightDecay),

                    # In a Conv2D layer with data_format="channels_last", the weight tensor has shape
                    # (rows, cols, input_depth, output_depth), set axis to [0, 1, 2] to constrain
                    # the weights of each filter tensor of size (rows, cols, input_depth).
                    kernel_constraint = max_norm(maxNorm, axis=[0,1,2]),
                    use_bias = False)(input_layer)
    block1 = BatchNormalization(axis = -1)(block1)  # bn_axis = -1 if data_format() == 'channels_last' else 1

    block2 = DepthwiseConv2D((1, in_chans),
                             depth_multiplier = D,
                             data_format='channels_last',
                             depthwise_regularizer=L2(weightDecay),
                             depthwise_constraint  = max_norm(maxNorm, axis=[0,1,2]),
                             use_bias = False)(block1)
    block2 = BatchNormalization(axis = -1)(block2)
    block2 = Activation('elu')(block2)
    block2 = AveragePooling2D((8,1),data_format='channels_last')(block2)
    block2 = Dropout(dropout)(block2)

    block3 = Conv2D(F2, (16, 1),
                            data_format='channels_last',
                            kernel_regularizer=L2(weightDecay),
                            kernel_constraint = max_norm(maxNorm, axis=[0,1,2]),
                            use_bias = False, padding = 'same')(block2)
    block3 = BatchNormalization(axis = -1)(block3)
    block3 = Activation('elu')(block3)

    block3 = AveragePooling2D((poolSize,1),data_format='channels_last')(block3)
    block3 = Dropout(dropout)(block3)
    return block3
```

Figure 34 - Code Snippet for Convolutional Block in ATCNet model

The Conv_block_ function is designed as a feature extraction block for neural network architectures like ATCNet as shown in the above figure. It begins with a 2D convolutional layer (Conv2D) using a specified kernel size (kernLength) and filter count (F1), followed by batch normalization (BatchNormalization) and ELU activation. Subsequently, a depthwise convolutional layer (DepthwiseConv2D) operates on the output, applying separate convolutions per input channel (in_chans) to capture spatial patterns efficiently. To support feature learning and avoid overfitting, ELU activation, average pooling, and dropout regularization are used after batch normalization. A second 2D convolutional layer further extracts features with a larger kernel size and more filters (F2). The resulting features undergo downsampling via average pooling and dropout before being returned as the output of the convolutional block. This thorough block architecture includes all of the essential operations required for efficient regularization and feature extraction in deep learning models for applications like signal processing and classification.

Convolutional-Based Sliding Window (SW):

In order to improve data as well as decoding accuracy, the research uses a sliding window approach to divide temporal sequences into multiple windows of information. However, this method has a computational cost because it requires the input data to be sent through the deep learning model n times.

4.5.5.2 Attention Block (AT):

The attention block uses multi-head self-attention for highlighting important aspects in the motor imagery EEG data to make sure the model can focus on the data that is the most important point in the temporal sequence. The attention block can be contributed by three ways. They are Multi Head Self Attention Block (MHSA), SE block (Squeeze and excitation) and CBAM (Convolutional Block Attention Module).

4.5.5.3 Temporal Convolutional Block (TC):

High-level temporal characteristics are extracted from the temporal sequence of MI-EEG data with the help of the TC block. A fully connected (FC) layer that uses a SoftMax classifier receives the complex temporal patterns that are captured by temporal convolutional layers.

```
#%% Temporal convolutional (TC) block used in the ATCNet model
def TCN_block(input_layer,input_dimension,depth,kernel_size,filters,dropout,activation='relu'):

    block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=1,activation='linear',
                   padding = 'causal',kernel_initializer='he_uniform')(input_layer)
    block = BatchNormalization()(block)
    block = Activation(activation)(block)
    block = Dropout(dropout)(block)
    block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=1,activation='linear',
                   padding = 'causal',kernel_initializer='he_uniform')(block)
    block = BatchNormalization()(block)
    block = Activation(activation)(block)
    block = Dropout(dropout)(block)
    if(input_dimension != filters):
        conv = Conv1D(filters,kernel_size=1,padding='same')(input_layer)
        added = Add()([block,conv])
    else:
        added = Add()([block,input_layer])
    out = Activation(activation)(added)

    for i in range(depth-1):
        block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=2**(i+1),activation='linear',
                       padding = 'causal',kernel_initializer='he_uniform')(out)
        block = BatchNormalization()(block)
        block = Activation(activation)(block)
        block = Dropout(dropout)(block)
        block = Conv1D(filters,kernel_size=kernel_size,dilation_rate=2**(i+1),activation='linear',
                       padding = 'causal',kernel_initializer='he_uniform')(block)
        block = BatchNormalization()(block)
        block = Activation(activation)(block)
        block = Dropout(dropout)(block)
        added = Add()([block, out])
        out = Activation(activation)(added)

    return out
```

Figure 35 - Code snippet for the ATCNet model in temporal convolutional block

The TCN_block function implements a Temporal Convolutional Network (TCN) block utilized in the ATCNet model, as shown in the figure above. It consists of a series of convolutional layers with varying dilation rates to capture different temporal contexts in the input data. Batch normalization ensures stable training by normalizing the activations, followed by an activation function (defaulted to ReLU) for introducing non-linearity. Dropout regularization helps prevent overfitting during training. Residual connections are used by adding the input layer to the output of convolutional layers, aiding in gradient flow and model convergence. For temporal feature extraction and learning in deep learning architectures, especially for tasks involving time-series analysis and signal processing, this modular TCN block design is essential.

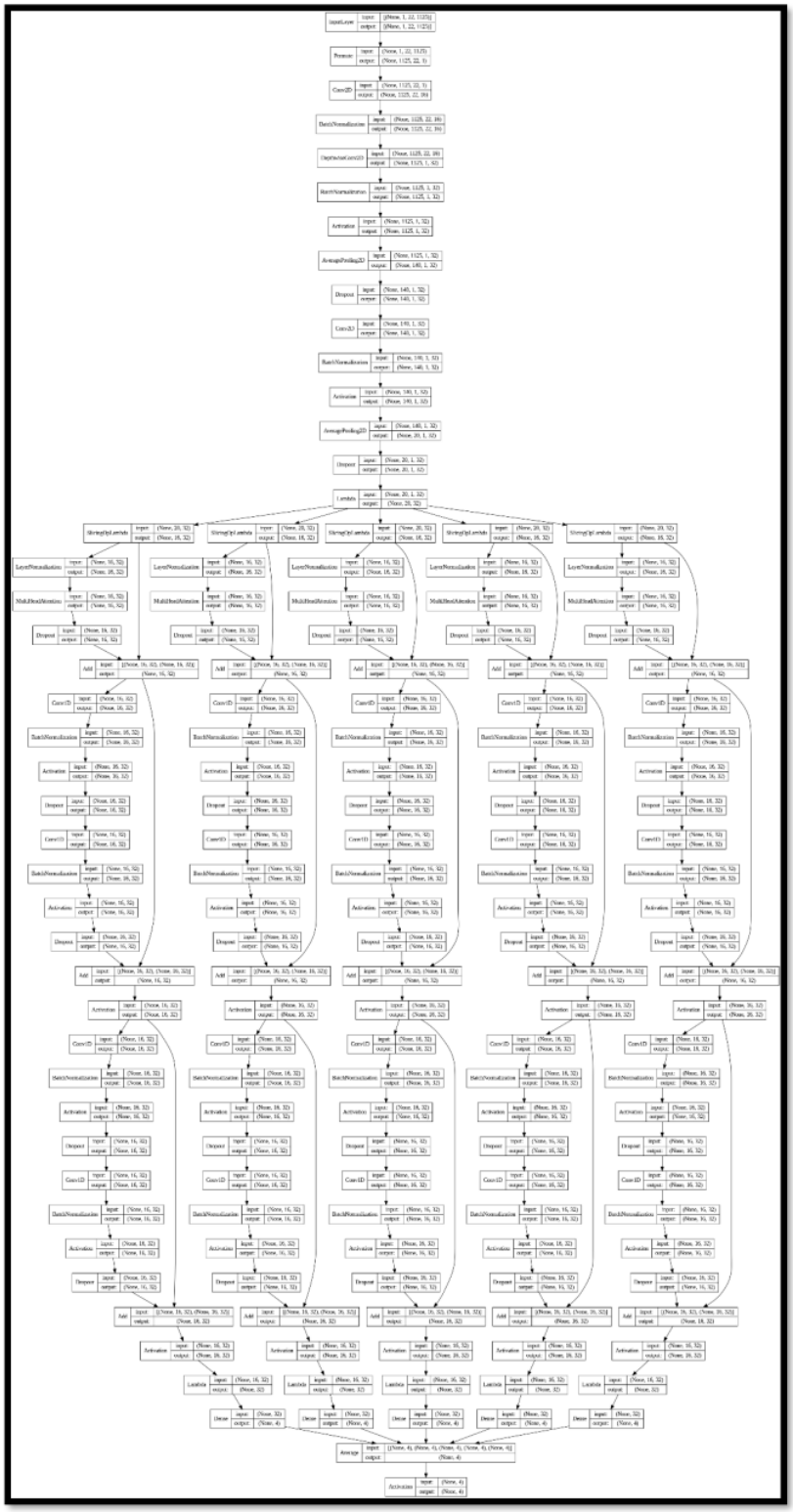
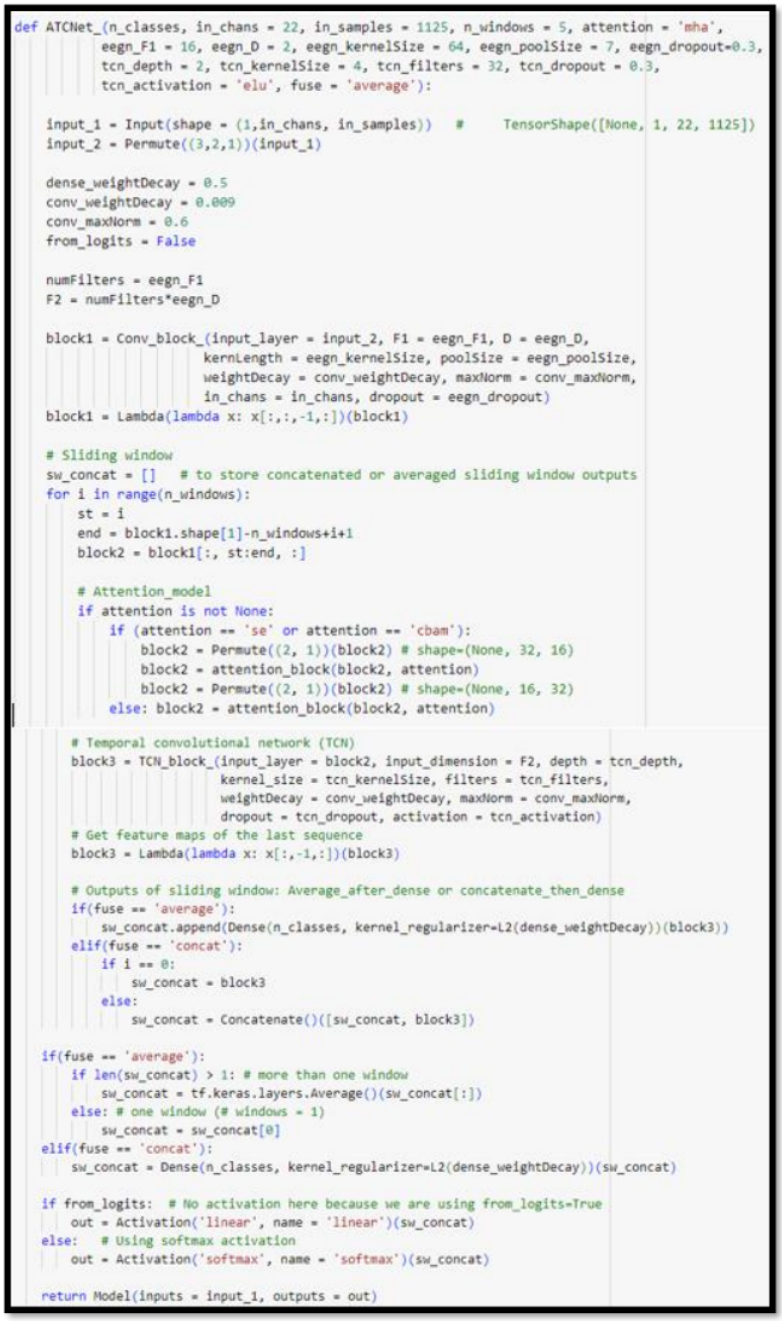


Figure 36 - Screenshot of the architecture for the model ATCNet in diagram format.



```
def ATCNet_(n_classes, in_chans = 22, in_samples = 1125, n_windows = 5, attention = 'mha',
            eegn_F1 = 16, eegn_D = 2, eegn_kernelSize = 64, eegn_poolSize = 7, eegn_dropout=0.3,
            tcn_depth = 2, tcn_kernelSize = 4, tcn_filters = 32, tcn_dropout = 0.3,
            tcn_activation = 'elu', fuse = 'average'):

    input_1 = Input(shape = (1,in_chans, in_samples))   #     TensorShape([None, 1, 22, 1125])
    input_2 = Permute((3,2,1))(input_1)

    dense_weightDecay = 0.5
    conv_weightDecay = 0.009
    conv_maxNorm = 0.6
    from_logits = False

    numFilters = eegn_F1
    F2 = numFilters*eegn_D

    block1 = Conv_block_(input_layer = input_2, F1 = eegn_F1, D = eegn_D,
                         kernLength = eegn_kernelSize, poolSize = eegn_poolSize,
                         weightDecay = conv_weightDecay, maxNorm = conv_maxNorm,
                         in_chans = in_chans, dropout = eegn_dropout)
    block1 = Lambda(lambda x: x[:,:,-1,:])(block1)

    # Sliding window
    sw_concat = []   # to store concatenated or averaged sliding window outputs
    for i in range(n_windows):
        st = i
        end = block1.shape[1]-n_windows+i+1
        block2 = block1[:, st:end, :]

        # Attention_model
        if attention is not None:
            if (attention == 'se' or attention == 'cbam'):
                block2 = Permute((2, 1))(block2) # shape=(None, 32, 16)
                block2 = attention_block(block2, attention)
                block2 = Permute((2, 1))(block2) # shape=(None, 16, 32)
            else: block2 = attention_block(block2, attention)

        # Temporal convolutional network (TCN)
        block3 = TCN_block_(input_layer = block2, input_dimension = F2, depth = tcn_depth,
                            kernel_size = tcn_kernelSize, filters = tcn_filters,
                            weightDecay = conv_weightDecay, maxNorm = conv_maxNorm,
                            dropout = tcn_dropout, activation = tcn_activation)
        # Get feature maps of the last sequence
        block3 = Lambda(lambda x: x[:,-1,:])(block3)

        # Outputs of sliding window: Average_after_dense or concatenate_then_dense
        if(fuse == 'average'):
            sw_concat.append(Dense(n_classes, kernel_regularizer=L2(dense_weightDecay))(block3))
        elif(fuse == 'concat'):
            if i == 0:
                sw_concat = block3
            else:
                sw_concat = Concatenate()([sw_concat, block3])

    if(fuse == 'average'):
        if len(sw_concat) > 1: # more than one window
            sw_concat = tf.keras.layers.Average()(sw_concat[:])
        else: # one window (# windows = 1)
            sw_concat = sw_concat[0]
    elif(fuse == 'concat'):
        sw_concat = Dense(n_classes, kernel_regularizer=L2(dense_weightDecay))(sw_concat)

    if from_logits:  # No activation here because we are using from_logits=True
        out = Activation('linear', name = 'linear')(sw_concat)
    else:   # Using softmax activation
        out = Activation('softmax', name = 'softmax')(sw_concat)

    return Model(inputs = input_1, outputs = out)
```

Figure 37 - Code snippet for ATCNet model blocks

The code shown in the above figure defines an Attention-based Temporal Convolutional Network (ATCNet) model for EEG signal classification. The ATCNet_ function takes various

parameters including the number of classes, input channels, samples, number of sliding windows, attention mechanism type (e.g., 'mha' for multi-head attention), EEGNet parameters (F1, D, kernel size, pool size, dropout), TCN parameters (depth, kernel size, filters, dropout, activation), and fusion method ('average' or 'concat') for combining window outputs. The function first configures input layers and applies convolutional blocks and EEGNet layers for feature extraction. The EEG data is subsequently processed using sliding windows, with the ability to add attention mechanisms, and is finally sent via TCN blocks. It applies a dense layer for classification after averaging or concatenating the outputs from several windows, depending on the fusion method used. The final output is passed through either a softmax activation function or returned as logits based on the from_logits parameter. his model architecture uses temporal convolution, feature fusion, and attention methods to provide reliable EEG signal classification.

The validation results of the ATCNet model reveal its strong performance across diverse subjects, with an impressive average accuracy of 88.12%. Notably, it excelled in subjects 3, 7, 8, and 9, showcasing accuracy rates ranging from 93.10% to 98.28%. The model showed constant competence across all participants, even with slight variations. Additionally, its efficient training time of 33.7 minutes highlights its rapid adaptability to new datasets. All things considered, these results highlight the adaptability and proficiency of the ATCNet model in managing complex data analysis and classification tasks.

4.6 Summary:

The chapter describes a structured methodology for accurately classifying human motion patterns based on EEG signals. It begins with data collection from reliable sources like Kaggle, followed by rigorous pre-processing to assure data reliability by eliminating noise and outliers. Deep learning architectures such as EEGNet, DeepConvNet, ShallowConvNet, ATCNet, and EEGTCNet are then employed to build models capable of extracting relevant features from EEG signals. These models are trained using prepared datasets to establish associations between input signals and corresponding movements, facilitating efficient detection and classification of human motion movements using EEG signals.

CHAPTER 5: RESULT AND DISCUSSION

5.1 Introduction:

The models will be evaluated throughout this phase, and results will be produced. The models are compared using metrics for performance, such as kappa score and accuracy, along with an analysis of the confusion matrix.

Confusion matrix: A performance metric called a confusion matrix compares predictions against true labels to provide an overview of the classification outcomes of a model. (Anuganti Suresh, 2020)

		Prediction Results	
		Positive (PP)	Negative (PN)
Actual Observations	Positive (P)	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

• True Positives (TP) occur when the actual value and the predicted value are both Positive.

• True Negatives (TN) occur when the actual value and the predicted value are both Negative.

• False Positives (FP), also known as Type 1 errors, happen when the actual value is Negative, but the prediction is Positive.

• False Negatives (FN), also known as Type 2 errors, occur when the actual value is Positive, but the prediction is Negative.

Accuracy: A machine learning performance parameter called accuracy is used to evaluate the efficacy of a classification model. The ratio of accurately predicted instances to all instances in the dataset is used to calculate it. Accuracy can be evaluated after the calculation of confusion matrix. The formula for accuracy is:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Kappa score: Another performance metric that takes chance agreement probability into consideration is the kappa score, which evaluates the agreement among model predictions and actual labels. It has a range of -1 to 1, where 1 denotes total agreement and 0 denotes agreement

at the chance level. A low score indicates below-average performance. The formula for a kappa score is:

63

kappa = (observed accuracy - expected accuracy) / (1 - expected accuracy) where expected accuracy is the percentage of cases properly classified by chance alone, and observed accuracy is the percentage of cases with matching model predictions and real labels.

In addition to model evaluation, this phase encompasses model deployment, a pivotal step where the model exhibiting the most promising performance metrics is chosen for implementation. Model deployment involves creating a user-friendly interface, typically in the form of a simple web page. This interface allows users to input feature values, triggering the model to generate the corresponding output class. By streamlining this process into an accessible web application, we ensure the seamless integration of our model into practical use cases, facilitating efficient decision-making and enhancing accessibility to its predictive capabilities.

5.2 Model Evaluations and Discussions:

5.2.1 EEGNet:

5.2.1.1 Accuracy for EEGNet:

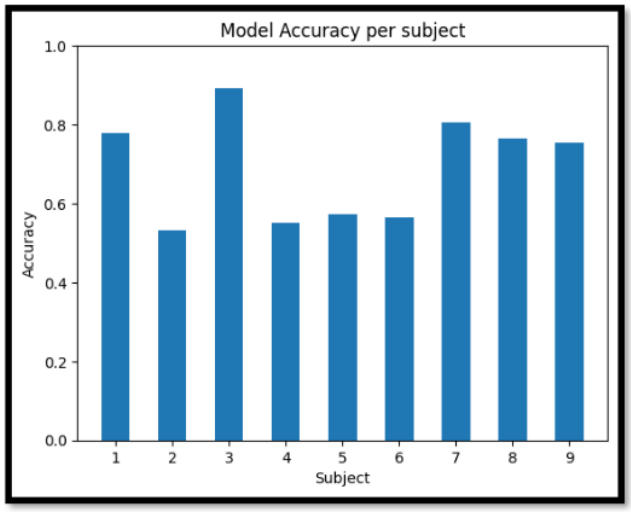


Figure 38 - Accuracy plot for EEGNet.

During the evaluation of trial 1, the EEGNet model showed different accuracy scores across 9 subjects. Significantly, Subject 2 had the lowest accuracy at 53.12% and Subject 3 the best at 89.24%. The remaining subjects achieved accuracies ranging from 55.21% to 80.56%. On average across all subjects, the EEGNet model achieved 69.06% accuracy. These results demonstrate the varying performance of the model when applied to different individuals in the dataset.

5.2.1.2 K-Score for EEGNet:

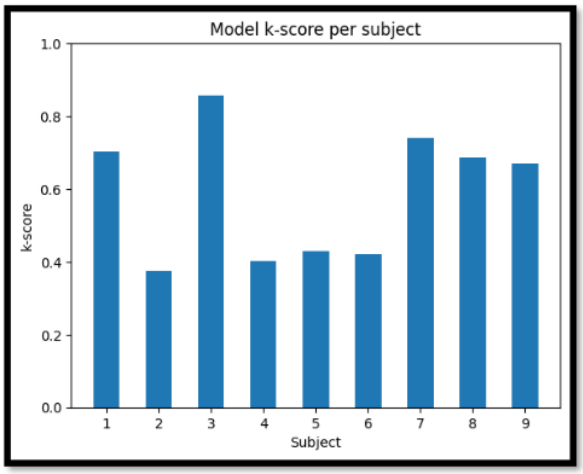


Figure 39 – K-Score Plot for EEGNet

The K-score plot as shown in the above figure gives a deeper understanding of how well the EEGNet model is really performing beyond just observing at accuracy figures. Each point on the plot indicates how well the model's predictions match up with the actual labels for different subjects. For instance, Subject 3 exhibits a high K-score of 0.856, representing that the model's predictions align well with reality for that individual. However, Subject 2 has a lower K-score of 0.375, indicating that the model's predictions are not as consistent for that individual.

Overall, the figure shows that, despite the EEGNet model's average accuracy of 69.06% across all subjects, the K-score enables us to observe the variation in performance within individuals. Furthermore, the model's quick processing time of 0.87 milliseconds per trial displays that it's efficient enough for real-time applications or managing massive amounts of EEG data, which adds to its practical usefulness beyond just accuracy numbers.

5.2.1.3 Confusion Matrix for EEGNet:

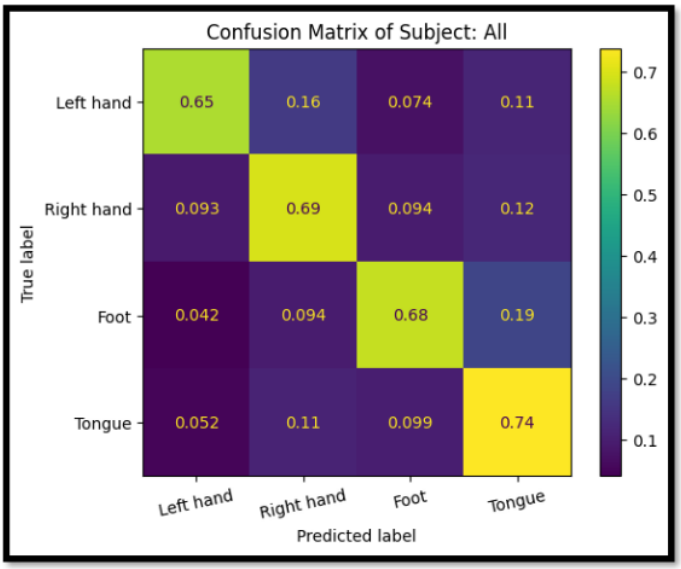


Figure 40 - Confusion matrix for EEGNet

On a dataset with four different classes—left hand, right hand, foot, and tongue movements—the model was evaluated. It achieved the highest accuracy in classifying Tongue movements (74%) but struggled the most with Left hand movements (65%) as shown in the above figure. Right-hand movements were classified with 69% accuracy, while Foot movements attained 68% accuracy. The confusion matrix analysis revealed specific misclassifications: the model confused Left hand movements with Right hand movements 16% of the time and Foot movements with Tongue movements 19% of the time. While the EEGNet model generally performed reasonably well in classifying EEG signals across these classes, there are evident areas for improvement, particularly in enhancing accuracy for Left-hand movement classification. Understanding these misclassifications can help future model refinements to enhance overall accuracy and reliability, especially in tasks requiring accurate movement classification.

5.2.2 DeepConvNet:

5.2.2.1 Accuracy for DeepConvNet:

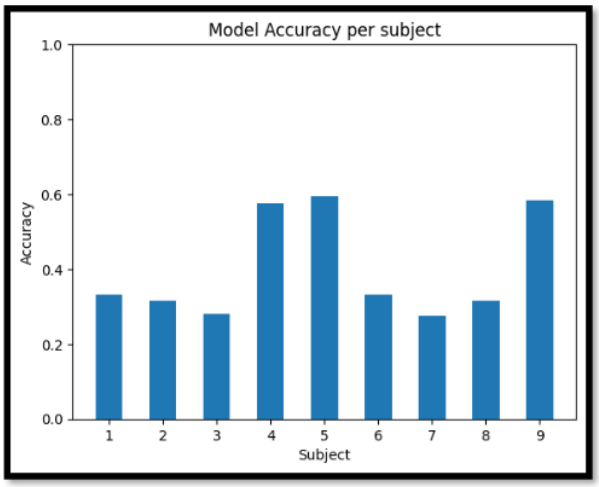


Figure 41 - Accuracy plot for DeepConvNet.

The DeepConvNet model displayed varied accuracy results across different subjects in the evaluation as shown in the above figure. Some subjects, like sub_4 and sub_5, achieved relatively high accuracies around 57.64% and 59.38%, while others such as sub_3 and sub_7 struggled with lower accuracies of 28.12% and 27.43%, respectively. On average, the model obtained an accuracy of 40.08% across all subjects, indicating its overall performance on this dataset. These results show that while the DeepConvNet model can identify EEG data quite well, there is still opportunity for development to ensure more consistent and accurate results for all subjects.

These accuracy findings highlight the need for more exploration into factors affecting the model's performance. These factors may involve the diversity of EEG signals among subjects, the complexity of tasks during data collection, and potential aspects of noise or artifacts in EEG recordings. By addressing these areas, researchers can enhance the DeepConvNet model, aiming to improve its accuracy and reliability for EEG data classification tasks.

5.2.2.2 K-Score for DeepConvNet:

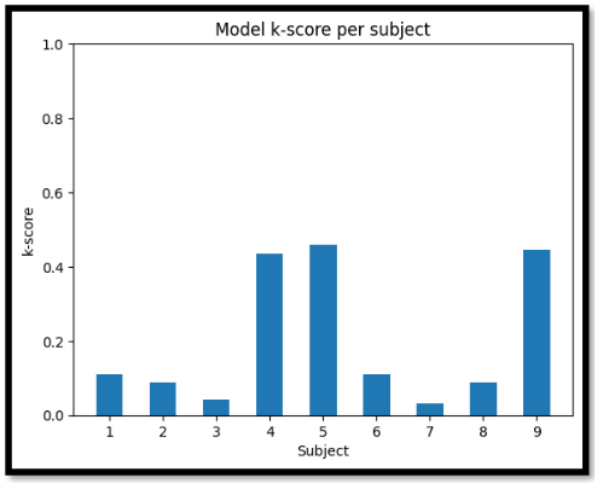


Figure 42 - K-Score plot for DeepConvNet.

The K-score results from the DeepConvNet model depicted in the above figure provide insights into how well the model's predictions match with actual labels, going beyond basic accuracy metrics. Across various subjects, K-scores range widely, from a low of 0.042 for sub_3 to a high of 0.458 for sub_5. A low K-score, like sub_3's, indicates limited agreement between predicted and actual labels, hinting at difficulties in accurately classifying EEG data for that specific subject. Subsequently, a high K-score, as seen in sub_5's case, indicates strong agreement between predicted and actual labels, demonstrating the model's ability to accurately represent underlying EEG signal patterns.

The average K-score across all subjects at 0.201 summarizes the DeepConvNet model's whole performance in label agreement. This average points to regions where the model's predictions might not be consistent or reliable across several EEG datasets, suggesting moderate degrees of agreement. In addition, the model exhibits efficiency with an inference time of 0.61 milliseconds per trial, representing quick processing appropriate for real-time applications or managing large-scale data sets.

5.2.2.3 Confusion Matrix for DeepConvNet:

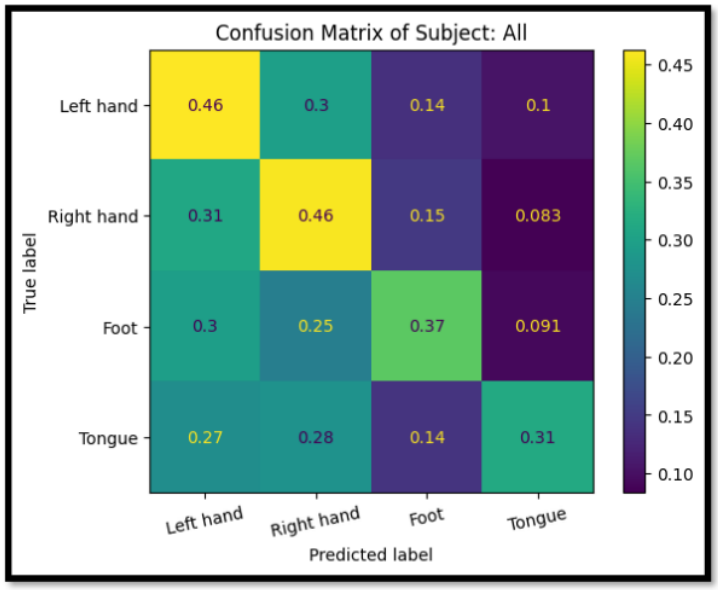


Figure 43 - Confusion Matrix for DeepConvNet

71 The confusion matrix serves as a useful tool for comprehending the model's performance across various areas. In the top right corner, which represents total accuracy, researcher observe that the model gets an accuracy of 45% in correctly classifying objects. It is particularly good at identifying between the left and right hands, with 46% accuracy in each case. However, its performance declines when identifying feet, with an accuracy of 37%, and it struggles the most with tongue classification, obtaining only 31% accuracy. The model frequently misclassifies left hands, right hands, and feet with other objects, highlighting confusion rates of 54%, 54%, and 63%, respectively. Tongues face significant misclassifications, being mistaken for other objects 27-28% of the time.

These results suggest areas for improvement in model training or feature representation to improve classification accuracy as well as lower misclassification rates across various categories.

5.2.3 ShallowConvNet:

5.2.3.1 Accuracy for ShallowConvNet:

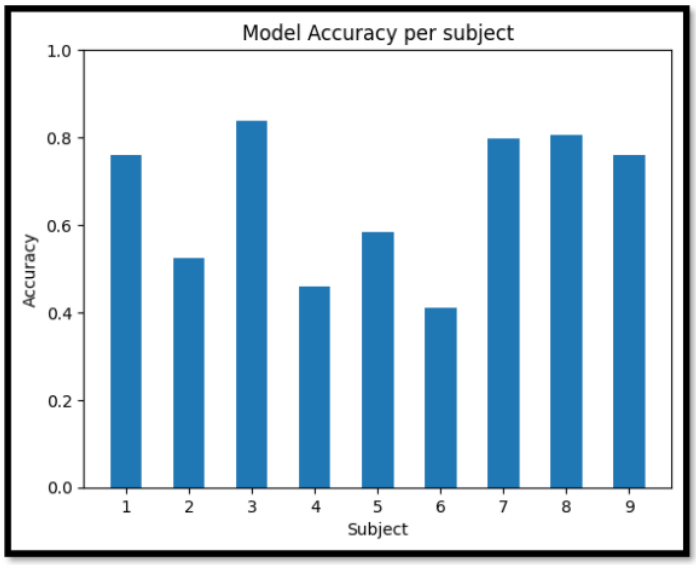


Figure 44 – Accuracy plot for ShallowConvNet

The ShallowConvNet model exhibits varying performance levels across different subjects, indicating its sensitivity to unique EEG signals and brain activity patterns. Notably, subjects such as sub_1 and sub_7 achieve high accuracies, representing successful capture and classification of unique EEG features. However, sub_6 struggles with lower accuracy, likely due to factors like noise or variability in brain activity. The average accuracy across all subjects, at 65.97%, represents overall decent performance but warrants further investigation into subject-specific variations to optimize model performance consistently across different scenarios. Even though the ShallowConvNet model displays promise for EEG data classification, ongoing improvements are necessary for reliable and consistent performance across diverse subject groups and experimental conditions.

5.2.3.2 K-Score for ShallowConvNet:

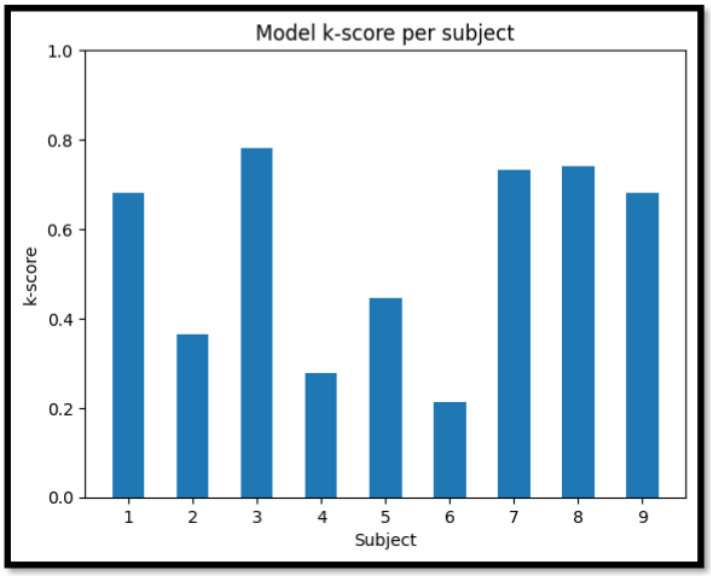


Figure 45 – K-Score Plot for ShallowConvNet

The K-score results for the ShallowConvNet model as shown in the figure reveal differing levels of agreement between predicted as well as actual labels across various subjects, with an average score of 0.546. Sub_3 is one subject that shows good agreement (K-score of 0.782), whereas sub_6 is one subject that shows lesser agreement (K-score of 0.213). This suggests room for improvement in the model's consistency and predictability in making predictions. Despite this unpredictability, the model's efficiency is notable, with a fast inference time of 0.86 milliseconds per trial, making it appropriate for real-time applications and large-scale data analysis tasks.

5.2.3.3 Confusion Matrix for ShallowConvNet:

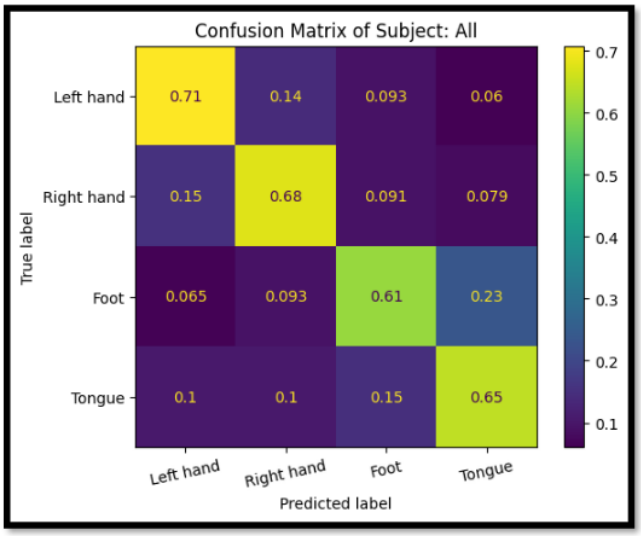


Figure 46 - Confusion Matrix for ShallowConvNet

Analysing the confusion matrix, as shown in the figure above, gives critical insights into how well the model performs across different object classes. The top right corner value, showing overall accuracy, indicates that the model correctly identifies objects 70% of the time. This suggests that it assigns the correct label to the object with a 70% accuracy rate. The model performs well in differentiating between the left and right hands, with 71% and 68% accuracy, respectively. However, it faces challenges in classifying feet (61% accuracy) and struggles the most with tongues (65% accuracy), representing difficulty in accurately identifying signals related to these objects compared to hand signals.

Further examination of the confusion matrix reveals instances of misclassification between classes. For instance, it occasionally mislabels left hands as right hands about 14% of the time and vice versa 15% of the time. Tongues are consistently misclassified with other objects approximately 10% to 15% of the time, indicating difficulty in differentiating tongue-related signals from hand or foot signals.

When comparing the ShallowConvNet and DeepConvNet models, researchers find that the first one performs significantly better—its total accuracy is 70% as opposed to 45%. Even with this improvement, there are still issues, particularly with accurately recognizing tongues and feet. This represents that further refinements in the model architecture or training approach may be necessary to achieve higher accuracy and reliability across all object classes.

5.2.4 EEG -TCNet:

5.2.4.1 Accuracy of EEGTC-Net:

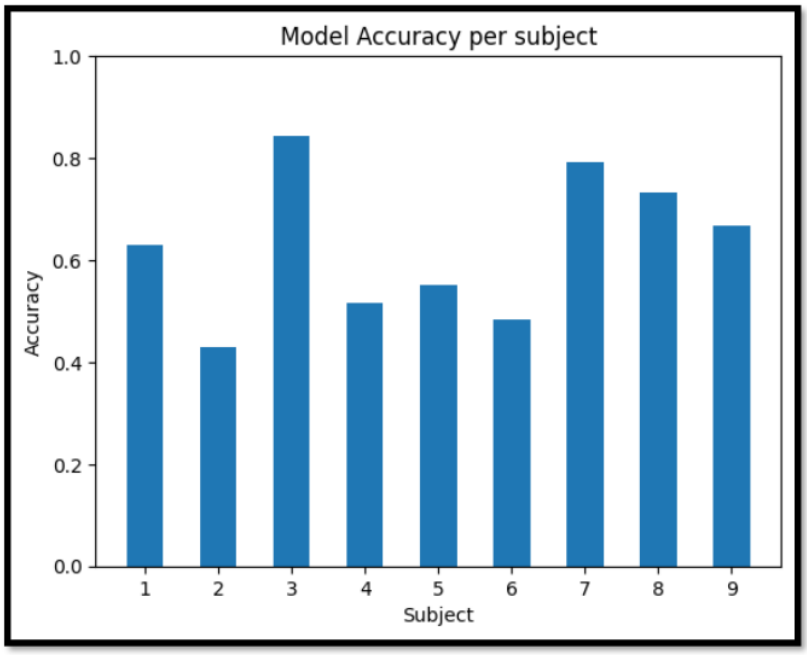


Figure 47 - Accuracy plot for EEG-TCNet

The EEG-TCNet model demonstrates diverse performance across subjects, with accuracies ranging from 43.06% to 84.38%. As shown in the above figure, subjects like sub_3 achieve high accuracies, indicating effective capture and classification of relevant EEG features. Alternatively, sub_2 has reduced accuracy, indicating difficulties in discerning certain patterns of brain activity.

The total average accuracy for all topics is 62.73%, which is an adequate level of performance. But further analysis is needed to understand variations in accuracy between subjects and optimize the model accordingly. While the EEG-TCNet model holds promise for EEG data classification, there is need for improvement to assure consistency including robustness across different subject populations and experimental situations.

5.2.4.2 K-Score of EEGTC-Net:

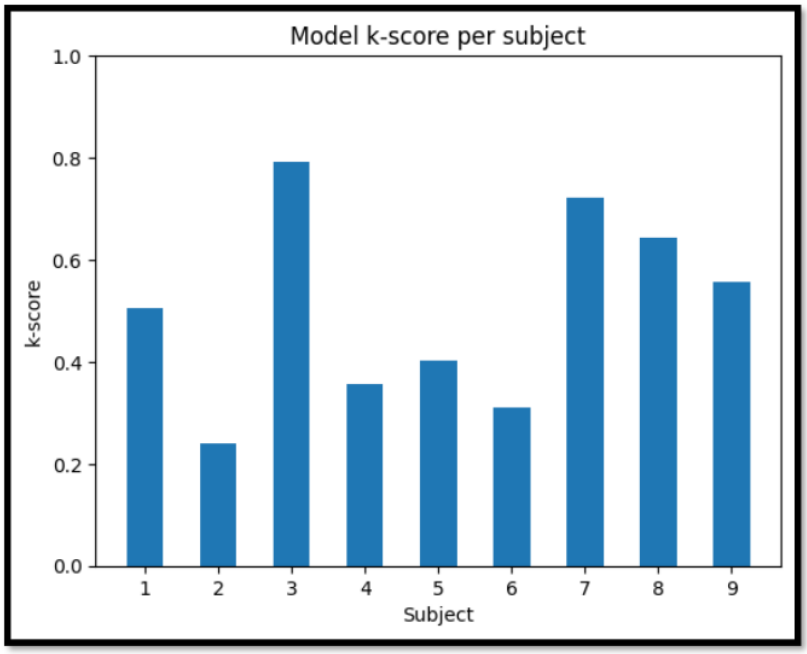


Figure 48 - K-Score Plot for EEGTCNet model.

The K-score results for the EEG-TCNet model show a reasonable level of agreement between predicted and actual labels among all subjects, with an average K-score of 0.503, as shown in the above figure. This shows that the model's predictions often match the ground truth labels rather well. However, there is potential for improvement to enhance the consistency and reliability of the model's predictions.

K-scores vary between participants; they range from 0.241 to 0.792 when evaluated more closely. A lower K-score, as shown in subjects like sub_2, indicates less agreement between predicted and actual labels, showcasing difficulties in accurately classifying EEG data for those subjects. Conversely, higher K-scores, such as in sub_3, shows stronger agreement and more reliable predictions. Furthermore, it's worth noting the model's efficiency, with an inference time of 0.60 milliseconds per trial.

5.2.4.3 Confusion Matrix for EEG-TCNet:

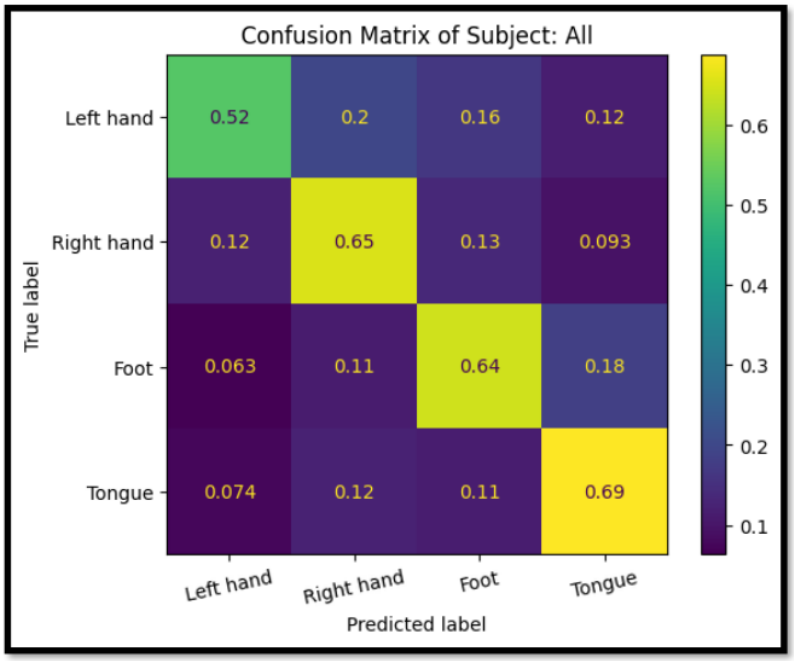


Figure 49 - Confusion Matrix of EEG-TCNet

The EEG-TCNet model performs differently when it comes to classifying various movement types using EEG inputs. It shows the highest accuracy in classifying tongue movements at 69%, while its accuracy is lowest for left-hand movements at 52%. The accuracy rates for movements with the right hand and foot are 65% and 64%, specifically, which is the middle of it. The model often confuses left-hand movements with right-hand movements (20% of the time) and foot movements with left-hand movements (18% of the time), including with some confusion between right hand and foot movements (13% of the time). The confusion matrix shows these misclassifications and highlights areas for model improvement, particularly in accurately classifying left-hand movements. Overall, while the model shows promise in classifying EEG signals into the four specified classes, there is potential for improvement in model's classification accuracy, especially for specific movement types.

5.2.5 ATCNet:

5.2.5.1 Accuracy for ATCNet:

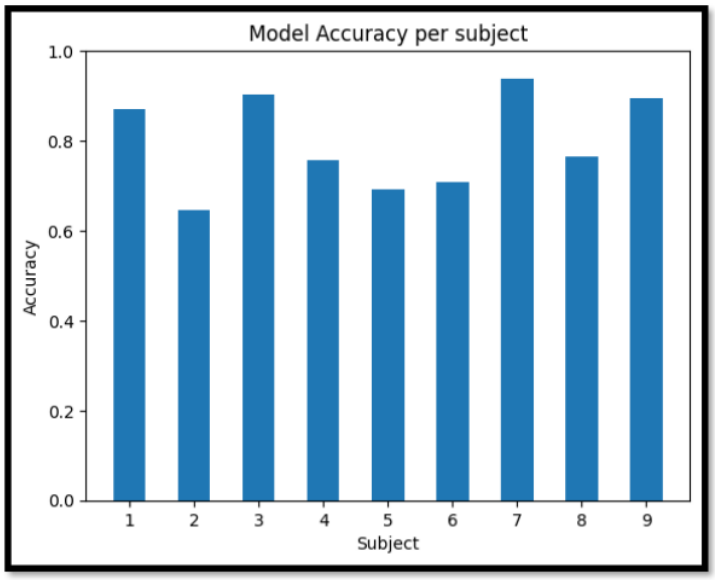


Figure 50 - Accuracy plot for ATCNet

The ATCNet model shows strong accuracy results across all subjects, with accuracies ranging from 64.58% for sub_2 to an incredible 93.75% for sub_7. Interestingly, subjects like sub_3 and sub_7 show high accuracies of 90.28% and 93.75%, respectively, representing their EEG signal features are well-observed and distinguished by the model. while subjects with lower accuracies, such as sub_2, still perform relatively well compared to other models, showcasing the model's flexibility to different data qualities and subject characteristics as depicted in the above figure.

Moreover, the average accuracy of 79.71% across subjects highlights the ATCNet model's consistent and reliable performance in EEG data classification. This reliability positions the model favourably for practical applications in brain-computer interface systems or clinical settings. Overall, these results demonstrate the ATCNet model's effectiveness in accurately interpreting complicated EEG patterns, which is important for developing neurotechnology as well as learning brain functions.

5.2.5.2 K-Score of ATCNet:

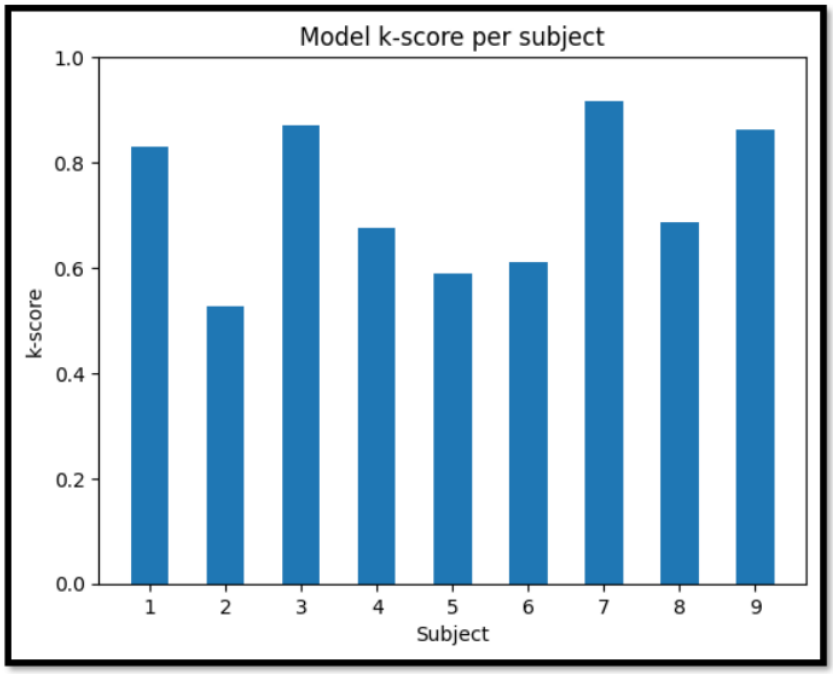


Figure 51 - K-Score plot of ATCNet

The K-score results for the ATCNet model, as shown in the above figure, highlight its strong agreement between predicted and actual labels across subjects, with an average K-score of 0.729. This high average score indicates the model's reliability in accurately capturing complex EEG patterns. Subjects differ in their degree of agreement; for example, sub_2's K-score is 0.528, whereas sub_7's is 0.917. Lower K-scores like sub_2's suggest reasonable consistency in predictions, although with less precision, while higher K-scores like sub_7's indicate strong agreement and dependability.

5.2.5.3 Confusion Matrix for ATCNet:

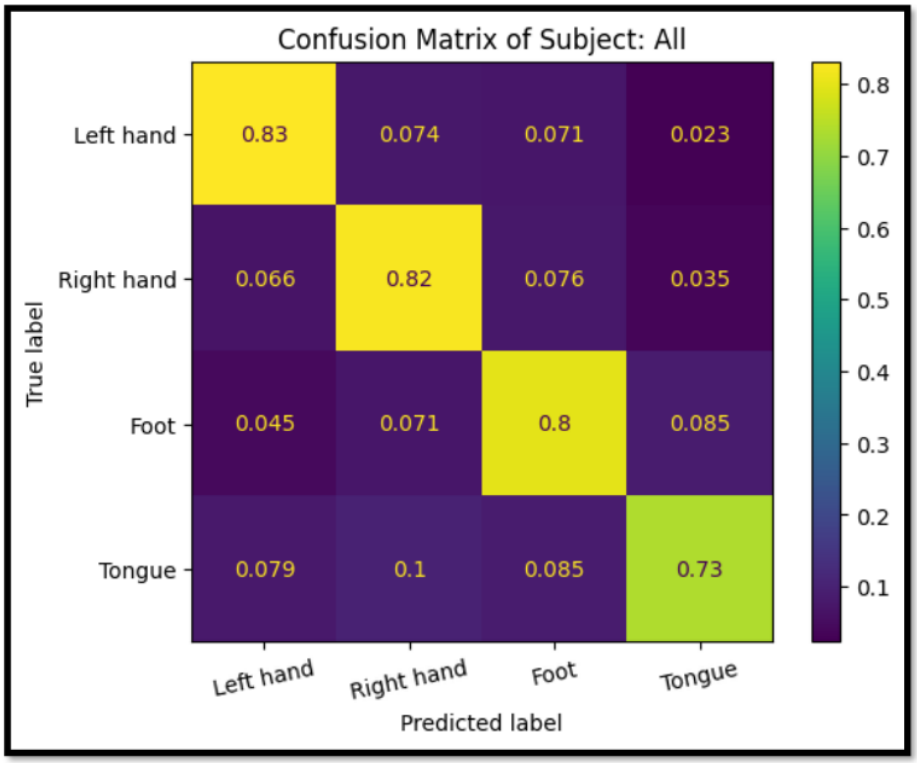


Figure 52 - Confusion Matrix for ATCNet

The confusion matrix analysis clearly shows a concise overview of the model's performance. The top right corner score of 0.8 indicates an overall accuracy rate of 80% in correctly classifying objects. The model does remarkably well in identifying left hands (83%) and right hands (82%), but it does somewhat worse in identifying feet (80%) and more quite insufficiently in distinguishing tongues (73%). Instances of confusion include left hands (17%), right hands (18%), and feet (20%) being occasionally misclassified with other objects, while tongues face consistent misclassifications ranging between 7.9% and 10%. Despite these challenges, the model's efficiency shows through with an average inference time of 0.89 milliseconds per trial, making it appropriate for real-time applications.

5.3 Model Comparison:

Model name	Validation accuracy (%)	Training time (min)	Testing accuracy (%)	Testing K-score
EEGNet	79.31	11.4	69.06	0.587
DeepConvNet	46.17	19.2	40.08	0.201
ShallowConvNet	72.99	20.7	65.97	0.546
EEG-TCNet	72.61	12.7	62.73	0.503
ATCNet	88.12	33.7	79.71	0.729

ATCNet model has outperformed the other models with the average subject accuracy of 79.71% and K-score value of 0.729. DeepConvNet model has the least average testing accuracy of 40.08% and K-score value of 0.201. So, the developer concludes that ATCNet model is the best model for this efficient detection of human motion movements having an accuracy of 80%.

5.4 Model Deployment:

In this step, the developer leverages the trained model—specifically the ATCNet Model—within a practical application. The goal is to create a straightforward webpage that predicts the intended motor imagery task of a user. This means that when a user interacts with the webpage, the ATCNet Model processes their input to determine the motor imagery task they are thinking about.

Essentially, the developer integrates the model into the webpage's backend logic so that it can process user data and provide accurate predictions. This allows users to experience real-time predictions based on their inputs, showcasing the practical application of the trained model.

5.4.1 Code snippet for model deployment:

```
import os

import streamlit as st
import base_utilites as bu

from models import ATCNet_

from helper import (
    set_background ,
    get_data ,
    plot_pie)
```

Figure 53 - Code snippet for model deployment part1

The code snippet shown above is a web application built using streamlit, designed for the purpose of predicting the intended motor imagery task of a user based on EEG signals, using the ATCNet Model. The environment setup for the Streamlit application includes importing essential libraries as well as modules to support the functionality. The 'os' module facilitates system operations, streamlit (imported as 'st') is utilised for creating the interactive app interface, and base_utilities provide utility functions essential for several tasks like data preprocessing or file handling. Custom modules such as models contain machine learning models required for predictions, while helper (.py file) contains project-specific functionalities like data loading or model evaluation. Together, these components make a strong framework for developing a Streamlit web application with streamlined data handling, machine learning capabilities, and interactive user interfaces customized to meet project requirements.

```
def prediction() :

    set_background(bu.format_path(
        '''
        Assets/
            Images/
                background/
                    prediction.jpg
        '''
    ))

    st.write(open(bu.format_path(
        '''
        Assets/
            Texts/
                prediction.txt
        '''
    )).read())

    subject = st.text_input('Enter the Subject name')
    sample = st.text_input('Enter the sample name')

    paths = os.listdir(bu.format_path(
        '''
        Assets/
            MAT_Files
        '''))
    paths = [
        bu.format_path(
            '''
            Assets/
                MAT_Files/
            '''
        ) + path
        for path
        in paths]

    if st.button('Predict') :

        error = 0

        if sample == '' or subject == '' :

            st.error('Please enter the values properly' , icon = '🚨')
            error += 1

        if error == 0 :

            if sample < 0 or sample > 287 :

                st.error('Please enter the Sample number in the given ranges', icon = '🚨')
                error += 1

            if error == 0 :

                with st.spinner('Just a minute, Getting your results !!') :

                    data_path = paths[int(subject) - 1]
                    x_train = get_data(data_path)
                    model = ATCNet_(
                        n_classes = 4 ,
                        in_chans = 22 ,
                        in_samples = 1125 ,
                        n_windows = 5 ,
                        attention = 'mha' ,
                        eegn_F1 = 16 ,
                        eegn_D = 2 ,
                        eegn_kernelSize = 64 ,
                        eegn_poolSize = 7 ,
                        eegn_dropout = 0.3 ,
                        tcn_depth = 2 ,
                        tcn_kernelSize = 4 ,
                        tcn_filters = 32 ,
                        tcn_dropout = 0.3 ,
                        tcn_activation='elu')

                    model.load_weights(bu.format_path(
                        f'''
                        run-1/
                            subject-{subject}.h5
                        '''))
                    label = [
                        'right_hand' ,
                        'left_hand' ,
                        'foot' ,
                        'tongue']

                    pred = model.predict(x_train)

                labels = ['Right hand', 'Left hand', 'Foot', 'Tongue']
                sizes = pred[sample]
                plot_pie(labels , sizes)
                labels = pred.argmax(axis = -1)
                st.write(f"The predicted value is : {label[labels.tolist()[sample]]}")
```

Figure 54 - Code Snippet for model deployment part 2

The prediction() function is a crucial part of a Streamlit application created for EEG data analysis as well as prediction. It starts by setting the background image and presenting introductory text to guide the user. Users are asked to provide the subject name and sample name within the numbers provided, which are then used to find the corresponding EEG data files. Error handling is developed to assure valid inputs, such as checking for empty fields or values outside the specified ranges, and appropriate error messages are displayed using Streamlit's st.error() function. Once the user provides valid inputs and clicks the "Predict" button, the function loads the pre-trained ATCNet model, which is made especially for classifying EEG data. The model is configured with several parameters such as the number of classes, input channels, sample size, and network architecture. It then loads the weights specific to the selected subject for prediction. The EEG data is processed and fed into the model for inference, resulting in predictions for the input sample. These predictions are displayed using a pie chart to show the distribution of probabilities across different classes, giving a clear understanding of the model's confidence in each prediction. Furthermore, the predicted class label is shown as text, providing a straightforward interpretation of the model's output to the user.

```
114  def home() :
115
116      set_background(bu.format_path(
117          '''
118          Assets/
119              Images/
120                  Background/
121                      home.jpg
122          '''
123      ))
124
125      st.write(open(bu.format_path(
126          '''
127          Assets/
128              Texts/
129                  home.txt
130          '''
131      )).read())
132
```

Figure 55 - Code Snippet for model deployment part 3

The above code snippet is portion of a larger application that most likely make use of the Streamlit framework for building interactive web applications with Python. In this snippet, the home() function is defined to manage the content appeared on the home page of the web application. There are two primary features in it. First, it sets the background image of the home page using the set_background() function and a specified image path. This function helps in customizing the visual appearance of the page, making it more engaging for users. Second, it reads and displays the content of a text file (home.txt) stored in the 'Assets/Texts/' directory. The content of this text file is dynamically shown on the home page using Streamlit's st.write() function, making it simple to control and update the text that is shown to users.

```python
def about() :

    col_1 , col_2 = st.columns(2)

    col_1.image('Assets/Images/Logos/De_Montfort.png')
    col_2.image('Assets/Images/Logos/APU.jpg')

    style = """
    <style>
    p {
    text-align: center; /* Center align the text */
    font-size: 20px;  /* Set the font size to 24 pixels */
    }
    </style>
    """

    st.markdown(style, unsafe_allow_html=True)  # Apply the CSS style

    st.write("<p align='center'><b>Final Year Project</b></p>", unsafe_allow_html=True)

    col_1 , _ , _ , _ , _ = st.columns(5)
    col_1.image('Assets/Images/Logos/Good_Health.png')

    st.write("<p align='center'><b>Efficient Detection of Human Motion Movements through</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>EEG signals using Deep learning</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>By</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>Venkta Krishna Chaitanya Bysani</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>TP062476</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>APD3F2308CSDA</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'>A report submitted in partial fulfilment of the requiremenets for the degree of</p>", unsafe_allow_html=True)
    st.write("<p align='center'>B.Sc. (Hons) Computer Scienece Specialism in Data Analaytics</p>", unsafe_allow_html=True)
    st.write("<p align='center'>at Asia Pacific University of Technology and Innovation", unsafe_allow_html=True)
    st.write("<p align='center'></p>", unsafe_allow_html=True)
    st.write("<p align='center'></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>Supervised by Dr. Mukil Alagirisamy</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>2nd Marker : Assoc Porf. Dr. Raja Rajeswari</b></p>", unsafe_allow_html=True)
    st.write("<p align='center'><b>2024</b></p>", unsafe_allow_html=True)
```

Figure 56 - Code Snippet for model deployment part 4

The about() function sets up a structured display of project details and contributors using Streamlit as shown in the above code snippet. It arranges content into two columns (col_1 and col_2) to display logo images related to the project. CSS styling is applied through an HTML style block (style) to center-align text and adjust font size. Markdown is used to format text and headings (<p> and <b>) for clarity and emphasis, presenting project title, author details, project ID, degree information, university name, and supervisor names, all aligned center horizontally for visual consistency. This feature makes a visually appealing and informative layout for the "About" page, making it easy for users to understand key project information and contributors' responsibilities.

```
171
172  options = st.sidebar.selectbox(
173      'Navigator' ,
174      options = [
175          'Home' ,
176          'Prediction' ,
177          'About'
178      ]
179  )
180
181  if options == 'Home' : home()
182  elif options == 'Prediction' : prediction()
183  elif options == 'About' : about()
```

Figure 57 - Code snippet for model deployment part 5

This code segment sets up a sidebar navigation menu in a Streamlit web application using the st.sidebar.selectbox() function, displaying users with options such as 'Home', 'Prediction', and 'About'. Based on the option selected by the user, the subsequent if-elif-else conditional statements determine which function to execute: selecting 'Home' triggers the home() function to display the home page content, 'Prediction' triggers the prediction() function to manage prediction-related tasks, and 'About' triggers the about() function to show project information. This structure efficiently directs users to the relevant sections of the web application based on their selection in the sidebar menu, providing a seamless navigation experience.

5.4.2 Website Interface:

Figure 58 - Website interface part 1

The above image shows the project's user interface, and it offers three main options to the user: "Home," "About," and "Prediction." Users can click on these options to access different sections of the website, including general information, project details, and prediction functionalities.

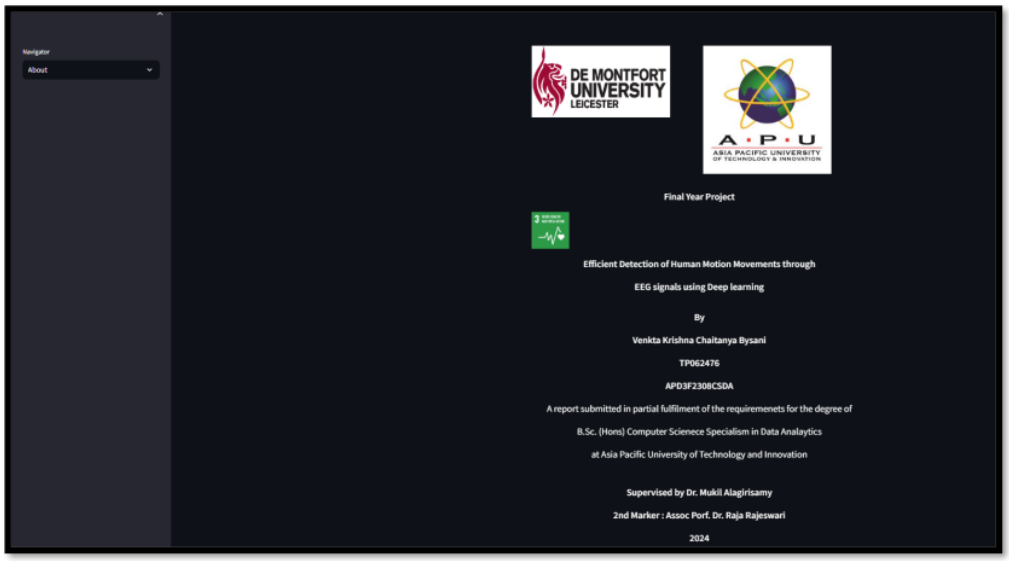


Figure 59 - Website Interface part 2

The "About" page features logos of the universities involved, details about the project author including their name and ID, information about the course name and intake code, as well as details about the project supervisor and second marker.

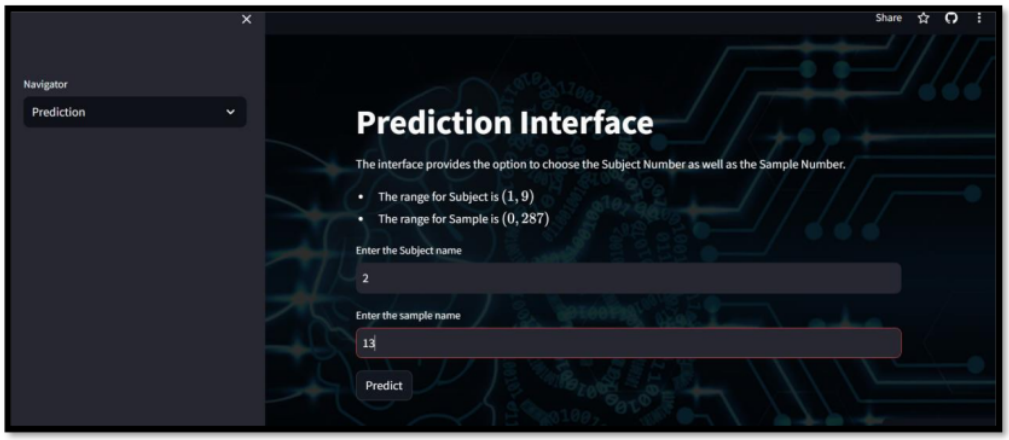


Figure 60 - Website Interface part-3

The "Prediction" page is where users can interact with the machine learning model deployed in the web application as shown in the above figure. It presents a form or interface where users can input specific values related to the prediction task. The results or predictions are then displayed to the user as shown in the below image, providing insights or classifications based on the input data and the model's analysis.

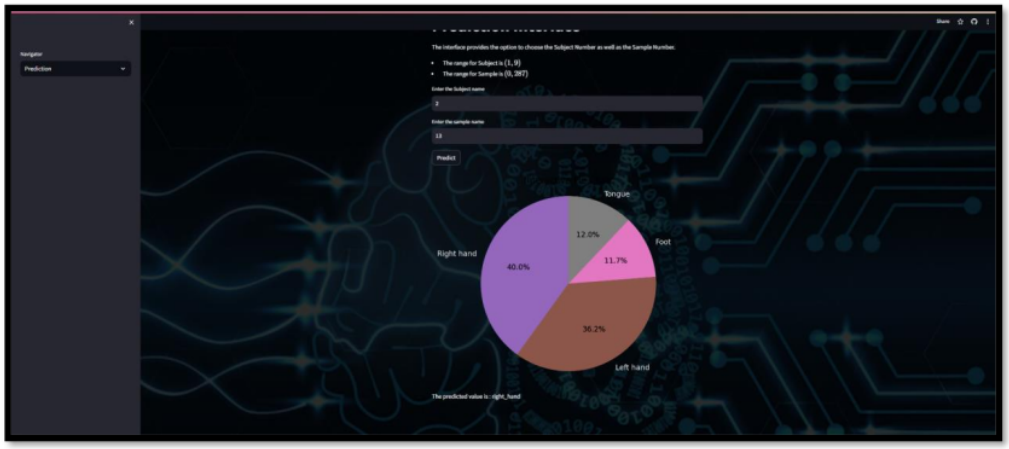


Figure 61 - Website Interface part 4

After entering input and selecting "Predict," the website applies the implemented machine learning model to the data. Subsequently, it shows the prediction results, often in the form of a pie chart as shown in the above image. This visual output helps users easily understand and interpret the model's predictions based on the input provided.

5.5 Summary:

In Chapter 5, Developer conducted comprehensive model evaluation, discussion, comparison, and deployment, culminating in the selection of the ATCNet model as the best performer. Achieving a best accuracy of around 80%, the ATCNet model emerged as the most reliable for classifying EEG data in motor imagery tasks. By taking advantage of its superior performance, Developer deployed the ATCNet model into a simple webpage interface. This streamlined approach not only showcases othe best-performing model but also provides users with a practical and accessible tool for EEG data classification tasks.

CHAPTER 6: CONCLUSION

6.1 Critical Evaluation:

6.1.1 Overall Project Achievement:

The first achievement of the overall project achievement would be predicting the intended motor imagery task of a user. The project has two essential phases, each contributing significantly to its outcome. Initially, a thorough literature study, technological research, and methodological selection for the research development were carried out. This groundwork was necessary for efficiently detecting human motion movements through EEG signals using deep learning techniques, following the structured CRISP-DM framework meticulously. This method allowed for a systematic exploration of the data analysis process, from precise data collection to the subsequent deployment of models. Moving into the second phase, an enhanced literature review and a detailed Phase 1 report set the stage for thorough model evaluations across five different deep learning models such as ATCNet, EEGNet, EEG-TCNet, DeepConvNet, and ShallowConvNet. The ATCNet model performed the best out of all of these, obtaining an impressive accuracy rate of 80%. Detailed analyses covering accuracy, K-score, and confusion matrices provided precise insights into model robustness and performance metrics. The strategic deployment of the top performing ATCNet model, as explained in Chapter 5, further underscored the project's success. These benchmarks not only mark the project's completion but also signify the invaluable insights gained into human motion movements through deep learning and EEG signal analysis. The user-friendly website interface, available at https://tp062476venkatakrishnafyp.streamlit.app/,now allows users to seamlessly predict motor imagery tasks based on their inputs, showcasing the project's tangible and useful outcomes.

6.1.2 Contribution of the project towards Community/Industries:

The project "Efficient Detection of Human Motion Movements through EEG Signals using deep learning" is a innovative initiative with significant implications, especially in the healthcare domain for patients with conditions such as paralysis or limited mobility. These individuals frequently encounter difficulties in effectively communicating their needs or actions due to physical limitations. Through the revolutionary integration of EEG signals as well as creative deep learning methodologies, this project introduces a transformative solution. It allows the real-time detection and interpretation of human motion intentions directly from brain signals, showcasing advanced leap forward in assistive technology.

By using EEG signals, which represent brain activity linked with motor intentions, and utilizing advanced deep learning algorithms, the project provides a reliable method to decode these signals into actionable commands or movements. This discovery has the potential to reshape caregiving and medical support paradigms for individuals with limited mobility, providing them newfound independence and enhancing their overall quality of life.

The project's influence extends beyond technical developments; it addresses an essential societal need by enabling caregivers and healthcare providers to better understand and respond to the needs of paralyzed individuals promptly and accurately. This not only improves patient care standards but also fosters a sense of dignity and autonomy among the affected population. The project's success ultimately demonstrates the transformational power of technology in tackling pressing healthcare issues and emphasizes the significance of multidisciplinary collaboration in bringing about significant societal change.

6.1.3 Strength of the project:

The project excels in several important areas, showcasing its creative as well as significant contributions. Its standout feature is the merging of EEG signals and deep learning algorithms, a innovative approach that sets it apart from conventional methods used to detect human motion movements effectively. This creative integration presents new opportunities for linked disciplines of study and development in addition to showcasing highly developed technological skills. The project has demonstrated remarkable accuracy rates, especially with the ATCNet model achieving an accuracy of 80%, showcasing its reliability and effectiveness in accurately predicting motor imagery tasks. For practical uses, such as helping people in the healthcare industry who are paralyzed or have restricted movement, this high accuracy rate is essential. The successful deployment of the best model into a user-friendly web interface improves accessibility and making the project's results available to a wider range of stakeholders, including healthcare providers. By encouraging cooperation across academic fields and promoting simplicity of use, this user-friendly plan enhances the project's strengths and possible social consequences.

6.2 Limitation:

While the project has achieved notable success, it is essential to acknowledge its limitations for comprehensive understanding and future improvements.

The project faces limitations primarily in its achieved accuracy of 80%, indicating potential improvements necessary for higher precision. This limitation is compounded by challenges arising from limited and diverse EEG datasets available for model training, impacting the generalization of the model across different scenarios. Furthermore, the interpretability and explainability of deep learning models—two essential components of healthcare applications—may be limited by their inherent complexity. Resource-intensives needs for training and deploying deep learning models also arise practical challenges. Ethical issues related to EEG data privacy, security, and biases need careful consideration. Additional validation across diverse demographic groups and conditions is essential to ensure broader applicability. Addressing these limitations involves improving user interface design, accessibility features, as well as model explainability for seamless integration and adoption in real-world healthcare environments.

6.3 Recommendation:

For future improvements, fine-tuning the model's hyperparameters and architectures using techniques like grid search or random search can enhance accuracy beyond the current 80%. Augmenting the dataset with techniques such as rotation, scaling, or noise addition can improve model generalization. Collaboration with domain experts like neuroscientists or medical professionals can provide insights for better feature selection and model refinement. Utilizing explainable AI methods will improve transparency in decision-making. For applications such as brain-computer interfaces, the system's responsiveness can be enhanced via real-time data processing capabilities. Ethical considerations such as data privacy, security, and model fairness should be prioritized for responsible deployments in real-world situations.

References:

Anuganti Suresh, (2020). What is confusion Matrix. Retrieved from

https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

Architecture, (2022). CNN Architecture Detailed explanation. Retrieved from

https://www.interviewbit.com/blog/cnn-architecture/

Britannica, (2023). About windows operating system. Retrieved from

https://www.britannica.com/technology/IBM-OS-2

Code Zero, (n. d).Image of the KDD. Retrieved from

https://codezero844163712.wordpress.com/2021/01/01/the-kdd-process-in-data-mining/

Coursera, (2023). Introduction to python. Retrieved from https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Codecademyteam, (n.d). what is an IDE? Retrieved from https://www.codecademy.com/article/what-is-an-ide

Daly & Wolpaw, 2018). BCI and its types. Retrieved from

https://www.researchgate.net/figure/Types-of-brain-computer-interfaces-and-their-uses-There-are-3-types-of-BCIs-They-can-be_fig1_335892770

Datagy, (2022). Introduction to scikit-Learn in python. Retrieved from

https://datagy.io/python-scikit-learn-introduction/

Divya Sreekumar, (2023). what is research methodology? Definition, types, and examples. Retrieved from

https://paperpal.com/blog/academic-writing-guides/what-is-research-methodology

Elana Zion, (2017). what are EEG signals. Retrieved from

https://brain.org.il/articles/faces-e.pdf

eLearn, (2023). Explanation of Pandas Library. Retrieved from

https://elearn.interviewgig.com/pandas-library/

Eleanor Thomas, (2023). Introduction to Keras. Retrieved from

https://www.dataquest.io/blog/tutorial-introduction-to-keras/

Elise Moreau, (2022). what is google Chrome Browser? Retrieved from https://www.lifewire.com/what-is-google-chrome-4687647

Enrique Corrales, (2023). pros and Cons of the visual studio. Retrieved from

https://www.developer.com/languages/microsoft-visual-studio-review/

Fazel-Rezai, M. R., Kumar, B. N., & Shenoy, P. (2018). A review of EEG-based BCI applications for neurorehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 26(10), (pp.1851-1865).

Fernando Munoz, (2018). Logo of the visual studio. Retrieved from

https://www.abd.es/2018/07/visual-studio-2019-comienza-a-mostrar-sus-caracteristicas/

Gege Zhan, Shugeng Chen & Yanyun Ji, (2022). EEG-Based Brain Network Analysis of Chronic Stroke Patients After BCI Rehabilitation Training. Retrieved from

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9271662/

Geeksforgeeks, (2023). what is machine learning and its types. Retrieved from

https://www.geeksforgeeks.org/what-is-machine-learning/

Geeksforgeeks, (2023). what is python? Retrieved from

https://www.geeksforgeeks.org/what-is-python/

Geeksforgeeks, (2023). PyCharm advantages and disadvantages. Retrieved from

https://www.geeksforgeeks.org/what-is-pycharm/

Geeksforgeeks, (2023). Introduction to matplotlib. Retrieved from

https://www.geeksforgeeks.org/python-introduction-matplotlib/

Geeksforgeeks, (2023). KDD Process, advantages and disadvantages. Retrieved from https://www.geeksforgeeks.org/kdd-process-in-data-mining/

Geeksforgeeks, (2023). Data Reduction in data mining. Retrieved from https://www.geeksforgeeks.org/data-reduction-in-data-mining/

Geeksforgeeks, (2023). Introduction to Convolutional neural network. Retrieved from https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Hamdi altaheri, Ghulam Muhammad & Mansour Alsulaiman.(2023). Physics Informed attention temporal Convolutional network for EEG based Motor imagery classification. IEEE Transactions on industrial informatics, vol.19, no.2, (pp.2249 – 2258).

Hao Zhu, Dylan Forenzo, & Bin He. (2022). On the Deep Learning Models for EEG-Based Brain-Computer Interface Using Motor Imagery. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.30 (pp.2283-2291).

Hongkong University of Science and technology. (N.A). Neurofeedback games using EEG based Brain computer interface technology. Retrieved from https://repository.hkust.edu.hk/ir/Record/1783.1-106838

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

IBM, (n. d). what is data mining? Retrieved from https://www.ibm.com/topics/data-mining

Imotions. (2021). What is EEG? Retrieved from https://imotions.com/blog/learning/research-fundamentals/what-is-eeg/

Intellipaat, (2023). Advantages & Disadvantages of python. Retrieved from https://intellipaat.com/blog/advantages-and-disadvantages-of-python/

iSixSigmastaff, (2022). Differences between Tangible and intangible benefits. Retrieved from https://www.isixsigma.com/methodology/tangible-vs-intangible-benefits-whats-the-difference/

Javatpoint, (2023). Pros and Cons of R Language. Retrieved from https://www.javatpoint.com/r-advantages-and-disadvantages

Javapoint, (n.d). Feature Selection in machine learning. Retrieved from https://www.javatpoint.com/feature-selection-techniques-in-machine-learning

Jiarong Wang, Luzheng Bi, Weijie Fei, (2023). EEG-Based Motor BCIs for Upper Limb Movement: Current Techniques and Future Insights. Retrieved from

https://ieeexplore.ieee.org/document/10309987

Jing Jin, Chang Liu, Ian Daly, &Yangyang Miao. (2020). Bispectrum-Based Channel Selection for Motor Imagery Based Brain-Computer Interfacing. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.28, no.10 (pp.2153 - 2163).

Joel Falconer, (2022). What is an IDE? How does it enable faster development? Retrieved from

https://www.sitepoint.com/what-is-an-ide/

John Spacey, (2023) What is target user? Retrieved from https://simplicable.com/design/target-user

Lari Giba, (2024). What is standardization? Retrieved from https://machinelearningcompass.com/dataset_optimization/standardization/

Lukasz Radzinski & Tomasz Kocejko, (2022). Deep learning approach on surface EEG based Brain Computer Interface. 15th International Conference on Human System Interaction (HSI), Melbourne, Australia (pp.1-6).

Mamunur Rashid, Norizam Sulaiman, Sabira Khatun, (2020). Current Status, Challenges, and Possible Solutions of EEG-Based Brain-Computer Interface: A Comprehensive Review. Retrieved from https://www.frontiersin.org/articles/10.3389/fnbot.2020.00025/full

Mary, (2022). Data Transformation. Retrieved from

https://www.techtarget.com/searchdatamanagement/definition/data-transformation

Michael X Cohen, (2017). Advantages of EEG signals. Retrieved from https://drive.google.com/file/d/11jGEfUkuLnTPTTUhtlriZ-iukv3Rm1JI/view

Microsoft, (2023). What is visual studio? Retrieved from

https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022

Mohammad Chebba, (2020). Knowledge discovery Data. Retrieved from

https://medium.com/analytics-vidhya/knowledge-discovery-data-kdd-a8b41509bff9

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). Foundations of Machine Learning, second edition. In *Google Books*. MIT Press. Retrieved from https://books.google.com.my/books?hl=en&lr=&id=dWB9DwAAQBAJ&oi=fnd&pg=PR5&dq=what+is+machine+learning&ots=AysQZSx3l2&sig=EgM2ilBo0Oa_9-KXZb7n1ivA9OY&redir_esc=y#v=onepage&q=what%20is%20machine%20learning&f=false

Muhammad Ahmad Mir, (2023). What are advantages and disadvantages of using visual studio? Retrieved from

https://medium.com/@ssc.ahmed.926748/what-are-the-advantages-and-disadvantages-of-using-visual-studio-code-or-atom-d3132bf1af85

Nick Hotz, (2023). CRISP Methodology Image. Retrieved from https://www.datascience-pm.com/crisp-dm-2/

Nick McCullum, (2020). The ultimate guide to the NumPy package for scientific computing in python. Retrieved from https://www.freecodecamp.org/news/the-ultimate-guide-to-the-numpy-scientific-computing-library-for-python/

NumPy, (n. d). what is NumPy? Retrieved from https://numpy.org/doc/stable/user/whatisnumpy.html

Packt, (n.d). Image of the ML types. Retrieved from https://subscription.packtpub.com/book/data/9781788398435/1/01lvl1sec9/discover-the-different-types-of-machine-learning

Patrick Hall, (2023). Clustering in Machine learning. Retrieved from https://www.techtarget.com/searchEnterpriseAI/definition/clustering-in-machine-learning

Pramod Gaur, Ram Bilas Pachori, Hui Wang & Girijesh Prasad. (2019). An automatic Subject Specific Intrinsic Mode Function Selection for enhancing Two-Class EEG based motor Imagery Brain Computer Interface. IEEE Sensors Journal, vol.19, no.16 (pp.6938-6947).

Pranay, (2016). Matplotlib with benroot. Retrieved from https://softwareengineeringdaily.com/2016/02/01/matplotlib-with-ben-root/

Programmiz, (n.d). Learn R language. Retrieved from https://www.programiz.com/r

Pulp Learning, (2021). Scikit-Learn python libraries for machine learning. Retrieved from https://pulplearning.altervista.org/librerie-python-per-il-machine-learning-scikit-learn/

Pypi, (n.d). Project Description of Scikit-Learn. Retrieved from https://pypi.org/project/scikit-learn/

Python, (n.d). History and features of python. Retrieved from https://www.python.org/doc/essays/blurb/

Qanelph, (2022). PyCharm logo update. Retrieved from

https://intellij-support.jetbrains.com/hc/en-us/community/posts/5909943756818-PyCharm-Logo-Update

Robin Tibor, Jost Tobias, Lukas, (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/hbm.23730

Rohit Sharma, (n.d). KDD Process in data mining. Retrieved from https://www.upgrad.com/blog/kdd-process-data-mining/

Sai Manikanth Thiyari, (2020). Types of matplotlib in python. Retrieved from https://dzone.com/articles/types-of-matplotlib-in-python

Selig, J. (2020, May 6). *What is machine learning? A definition - expert system*. Expert.ai. Retrieved from https://www.expert.ai/blog/machine-learning-definition/

Simran Kaur Arora, (2023). Features of TensorFlow and its image. Retrieved from https://hackr.io/blog/how-to-install-tensorflow

Simran Kaur Arora, (2023). What is PyCharm and its features. Retrieved from https://hackr.io/blog/what-is-pycharm

Softwarekeep, (n. d). Features and characteristics of visual studio. Retrieved from https://softwarekeep.com/help-center/what-is-microsoft-visual-studio-where-can-i-download-it

Soroosh Shahtalebi & Amir Asif. (2020). Siamese Neural Networks for EEG Based Brain computer Interfaces. Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Montreal, Canada. (pp.442-446).

TechTarget contributor, (2023). Brain computer interface. Retrieved from https://www.techtarget.com/whatis/definition/brain-computer-interface-BCI

Techvidvan, (n. d).Characteristics & features of R language. Retrieved from https://techvidvan.com/tutorials/r-features/

The Indian Wire staff, (2019). Image of the R language. Retrieved from https://www.theindianwire.com/programming/r-programming-language-best-resources-134662/

Thorir Mar, Michael Hersche, Xiyaying Wang, (2020). EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain-Machine Interfaces. Retrieved from https://arxiv.org/abs/2006.00622

Tianwei Shi, Ling Ren & Wenhua Cui. (2020). Feature Extraction of Brain–Computer Interface Electroencephalogram Based on Motor Imagery. IEEE Sensors Journal, vol.20, no.20 (pp.11787-11794).

Vernon J, Lawhern, Amelia J. Solon (2018). EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces. Retrieved from https://arxiv.org/abs/1611.08024

V. K. Benzy, A. P. Vinod, R. Subasree & Suvarna Alladi. (2020). Motor Imagery Hand Movement Direction Decoding Using Brain Computer Interface to Aid Stroke Recovery and Rehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.28, no.12 (pp.3051-3062).

W3 schools, (n.d). Characteristics and introduction of python. Retrieved from

https://www.w3schools.com/python/python_intro.asp

W3schools, (n.d). TensorFlow introduction. Retrieved from

https://www.w3schools.in/tensorflow/introduction

Will Hillier, (2023). What is data cleaning and why does it matter? Retrieved from

https://careerfoundry.com/en/blog/data-analytics/what-is-data-cleaning/

Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). Cbam: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

Xia Chen, Xiangbin Teng, Han Chen, (2023). Toward reliable signals decoding for electroencephalogram: A benchmark study to EEGNeX. Retrieved from https://arxiv.org/abs/2207.12369

Xin Deng, Boxian Zhang, Nian Yu & Ke Liu. (2021). Advanced TSGL-EEGNet for Motor Imagery EEG-Based Brain-Computer Interfaces. IEEE Access, vol.9 (pp.25118-25130).

Yudha Vijaya, (2021). CRISP Methodology for data science project. Retrieved from

https://towardsdatascience.com/crisp-dm-methodology-for-your-first-data-science-project-769f35e0346c

Appendices:

Appendix A: PPF – Title Registration Proposal

Project Proposal Form (PPF)

Office Record	Receipt
Date Received:	Student name: Venkata krishna
Received by whom:	Student number: TP062476
	Received by:
	Date:

DRAFT PROJECT PROPOSAL FORM

Proposal ID :

Supervisor :
1) Dr. Mukil Alagirisamy
2) Dr. Murugananthan Velayutham
3) Ms. Minnu Helen Joseph
4) Dr. Vazeerudeen Abdul Hamed
5) Mr. Raheem Mafas

Student Name : **VENKATA KRISHNA CHAITANYA BYSANI**

Student No : **TP062476**

Email Address : **tp062476@mail.apu.edu.my& krishnabysanil@gmail.com**

Programme Name : Bsc. (Hons) in computer science with specialism Data Analytics (CSDA)

Title of project : **"Efficient Detection of human motion movements through EEG signals using Deep learning".**

Please record which module(s) your topic is related to:

Python Programming (CT108-3-1-PYP)
Data Management (CT075-3-2-DTM)
Datamining and predictive modelling (CT119-3-2-DMPM)

1

Table of Contents

1. Introduction:

In today's world, there is a significant number of people facing neurological disabilities, rendering them dependent on others for basic tasks involving their hands and legs. This project, " Efficient Detection of human motion movements through EEG signals using Deep learning," has the capability of offering these people crucial support and enabling them to regain control over their life.

The scientific approach of choice for recording the electrical activity produced by the brain using electrodes embedded in the scalp is electroencephalography or EEG. Data may be obtained from identical scalp placements throughout all respondents because of electrodes attached in elastic caps like bathing caps, which allows for much quicker application. (Imotions,2021).

A relatively new method of developing contact between humans and machines, the brain-computer interface (BCI) converts brain activity into instructions to facilitate interaction and control. BCI is an interesting way for paralyzed people to communicate with the outside world since it can identify human intentions. The most popular invasive method for capturing activity in the brain is the electroencephalogram (EEG), which is used by many of the current BCI systems (Hongkong University of Science and technology, N.A). Motor imagery (MI) is one of the most popular BCI paradigms, in which people visualize moving their limbs to operate the system. MI-based BCIs have an extensive number of prospective applications, such as supportive technology for paralyzed patients, neurorehabilitation, and gaming (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023).

Deep learning for MI-based BCI control has drawn more attention in the past few years. Deep learning methods are now being used to create innovative and modern BCI systems since they have been proven to produce advanced results on a range of BCI datasets. In the past several years, deep learning, a class of machine learning algorithms, has transformed a few industries, including speech recognition, computer vision, and natural language processing. Because they can recognize complicated patterns in data and extract features that are challenging to identify using conventional techniques, deep learning algorithms are particularly well suited for BCI applications (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023).

2. Problem Statement:

Electroencephalography (EEG) signals are used in brain-computer interfaces (BCIs) to enable people to operate devices or communicate with their minds. However, a variety of problems restrict them from developing normally, such as:

1. **Limited performance as the number of mental tasks increases:**

   BCI performance degrades as the number of mental tasks that users can control increases. Due to the noise and complexity of EEG data, it can be challenging to differentiate between various mental states (Soroosh Shahtalebi & Amir Asif, 2020).

2. **Noise and redundant information in multi-channel EEG:**

   According to Jing Jin & Chang Liu (2020), it might be challenging to extract information about the user's mental state from EEG signals because of their high levels of noise and redundant information.

3. **Poor time-frequency localization as well as excessive non-stationarity of EEG signals:**

   EEG signals have poor time-frequency localization, and they are also highly non-stationary, which makes it difficult to monitor changes in the user's mental state over time (Pramod Gaur, Ram Bilas Pachori, 2019).

4. **Low signal-to-noise ratio (SNR) of EEG signals:**

   EEG signals have a low SNR, which makes them susceptible to interference from external sources such as muscle activity, and eye movements, along power lines. (Pramod Gaur, Ram Bilas Pachori, 2019).

5. **Feature extraction:**

   According to the authors (Fazel-Rezai, Kumar, & Shenoy, 2018), The process of extracting appropriate and reliable features from EEG data is challenging.

3. Project Aim and Objectives:

The aim of this project ("Efficient Detection of human motion movements through EEG signals using Deep learning ") is to develop a model using deep learning that can accurately predict the intended motor imagery task of a user. This could have a significant impact on people with disabilities, as it could enable them to control prosthetic devices or other assistive technologies using their thoughts.

The objectives of this project are:

1. To build a deep learning model that can recognize various motor imagery tasks from EEG data, such as left- and right-hand motions, and tongue and foot movements.
2. To Create a communication pathway for people with neurological disabilities in order that they can communicate properly and express their requirements and wishes.
3. To build effective motor imagery based BCIs to aid in neurorehabilitation, improving motor recovery as well as the quality of life for people with spinal cord injuries and cerebral paralysis.
4. To Develop a machine learning model for classifying motor imagery that achieves at least 80% accuracy on a held-out test set.

4. Literature Review:

Hamdi Altaheri, Ghulam Muhammad, and Mansour Alsulaiman used the BCI Competition IV-2a dataset in the research they conducted, which included 5184 trials carried out on nine patients using 22 EEG electrodes and three EOG electrodes. Each trial was connected to one of four motor imagery (MI) tasks, which included visualizing the tongue, right hand left hand, and both foot movements. For the classification of these MI tasks, the researchers used an Attention-based Temporal Convolutional Network (ATCNet), which consists of three main blocks: the Temporal Convolution Block (TC), designed to extract high-level temporal features, the Multihead Self-Attention Block (AT), that highlights important information within the temporal sequence, and the Convolution Block (CV), responsible for encoding raw MI-EEG signals into concise temporal sequences. The accuracy and Kappa score measures used in performance evaluation showed that using the BCI Competition IV-2a dataset, the ATCNet model was superior to the state-of-the-art methods. (Hamdi Altaheri | Ghulam Muhammad | Mansour Alsulaiman, 2023)

In the research Conducted by Jing Jin and Chang Liu aimed to build a model that could recognize EEG channels with improved temporal as well as spatial characteristics. Three different datasets—the BCI Competition IV dataset 1, the BCI Competition III dataset IVa, and the BCI Competition III dataset IIIa—were used to validate their methodology. They extracted features from EEG signals using bispectrum analysis, a method that is proficient at extracting non-linear and non-Gaussian information, including the Sum of Logarithmic Amplitudes (SLA) and the First Order Spectral Moment (FOSM). These features were subsequently utilized in the Bispectrum-based Channel Selection (BCS) procedure to identify the best channels for a BCI based on Motor Imagery (MI), or motor imagery. In order to extract suitable features from the selected channels, the Common Spatial Pattern (CSP) method was subsequently used. Using accuracy as the performance criteria, Support Vector Machine (SVM) and Linear Discriminatory Analysis (LDA) classifiers were used for task classification. When compared to conventional 3C-CSP approaches, the findings showed considerable performance improvements, with accuracy rates for datasets 1, 2, and 3 of 83.8%, 86.3%, and 77.8%, respectively. (Jing Jin | Chang Liu, 2020)

In research including both healthy volunteers and stroke patients, V.K. Benzy and A.P. Vinod used BCI based on EEG to interpret the directions of imagined hand movements. A motorized arm

support was given control instructions using the decoded left/right-hand movement imagining. The synchronization parameter Phase Locking Value (PLV), derived from EEG, was employed by the researchers to figure out the directionality of the MI task. The time bin relevant to the MI task was chosen using event-related desynchronization/synchronization (ERD/ERS) analysis on the Mu and Beta frequency bands of EEG. By choosing the most significant channel pairings that offered the greatest variation in PLV characteristics, the differences between the two orientations of MI tasks were discovered. In both healthy people (63.7% to 82.8%) and stroke patients (56.9% to 74.4%), the classification accuracy was shown to be enhanced by employing ERD/ERS analyses, according to the researchers. Additionally, a stroke patient feedback session was held, increasing the classification accuracy to 68.63%. According to their research, EEG-based BCI may be utilized to operate prosthetic devices along with other assistive technologies for both healthy individuals and stroke patients. (V.K. Benzy | A.P. Vinod, 2020)

Tianwei Shi, Ling Ren, and Wenhua Cui conducted a concentrated investigation in their research to solve two-class classification issues in motor imagery EEG. Their research focused on categorizing EEG signals associated with left-hand vs. right-hand, left-hand vs. foot, and right-hand vs. foot images. The most effective time-frequency filter type and filtering range was chosen for the study's preliminary processing using wavelet packet analysis. The filtered EEG data received additional refining in the feature extraction process utilizing the Common Spatial Pattern (CSP) and Adaptive Auto-regressive (AAR) methods. Band energy, sample entropy, and order accumulation were all investigated as potential distinguishing factors for the categorization of motor imagery. The study also performed a comparison of the classification results obtained using Bayesian, common space, and linear discrimination classifiers. Additionally, the training set data allowed for the co-space model's best spatial filter to be derived, which maximizes the variation among the two job categories. The classification performance showed a benefit from the improved distance criterion. his improved distance criterion has a positive effect on classification rate and proves that the proposed method can effectively extract the features of EEG signals during motor imagery. (Tianwei Shi | Ling Ren | Wenhua ,2020)

On the other research conducted by two famous authors, to discriminate between left- and right-hand motor imagery despite the dataset's four initial classifications, Lukasz Radzinski, and Tomasz Kocejko's research used the BCI Competition IV dataset 2a. To reduce higher-frequency noise and

eye movement artifacts, band-pass filtering was first used during preprocessing. Their model used a variety of methods, such as Deep and Shallow Convolutional Neural Networks (CNNs), the Common Spatial Pattern (CSP), the Filter Bank Common Spatial Pattern (FBCSP), and data augmentation approaches. With the use of spatial filters that increase the variance ratio between the two classes, CSP was able to maximize signal separation. Prior to spatial separation via CSP, FBCSP, an improved form of CSP, employed frequency filtering across multiple banks. The study also looked at CNNs for classifying Motor Imagery (MI), with Deep ConvNet focusing on deep learning and Shallow ConvNet emulating FBCSP's results. Frequency shifting was used for data augmentation, and the augmented set was included with the initial training data. A detailed investigation of methodologies involved comparisons, and Shallow Cropped Aug+CSP showed the greatest accuracy at 79.17%. (Lukasz Radzinski | Tomasz Kocejko, 2022)

On the other research conducted, Hao Zhu, Dylan Forenzo, and Bin He utilized two of the largest publicly available datasets, namely the MBT-42 and Med-62 datasets. In the MBT-42 dataset, subjects were tasked with performing left or right cursor movement tasks, while the Med-62 dataset involved 62 subjects participating in cursor movement control tasks of three types: left/right (LR) movement only, up/down (UD) movement only, and combined 2D movement (2D). The study entailed a comparative analysis of five deep learning models: EEGNet, Shallow ConvNet, Deep ConvNet, MB3D, and ParaAtt. EEGNet, a convolutional neural network, comprised three convolutional layers and one fully connected layer, aiming to encode various EEG feature extraction concepts, including optimal spatial filtering and filter-bank construction. Deep and Shallow ConvNet, two variations of convolutional neural networks, featured distinct architectural designs. Multi-Branch 3D CNN (MB3D) accepted CxT matrices as input, transforming the 2D input into a 3D tensor and subsequently utilizing 3D convolutional layers for generating predictions. Parallel Self-Attention Network (ParaAtt) incorporated attention modules, effectively capturing global relationships among input entries. The research encompassed both within-subject and cross-subject analyses, ultimately demonstrating that EEGNet outperformed other methods for both datasets (Hao Zhu | Dylan Forenzo | Bin He, 2022).

In the other study conducted by Xin Deng, Boxian Zhang, and Nian Yu used data from two open-access datasets: the BCI Competition IV 2a dataset from 2008, which had four motor imagery classes, and the BCI Competition III IIIa dataset, which featured the same classes but was recorded using 60 electrodes. They used the Filter-Bank Common Spatial Pattern (FBCSP), EEGNet, TSGL-EEGNet, Model Saving, and Ensemble Methods in their investigation, which combined all five of these techniques. While EEGNet used a convolutional architecture with feature selection using Temporal-constrained Sparse Group Lasso (TSGL), FBCSP added frequency domain features to the CSP technique to improve it. Model Saving techniques reduced training time, and Ensemble Methods, particularly stacking, improved the performance of the final model. With an average accuracy of 88.89% and an average kappa value of 0.8519, the TSGL-EEGNet stacking approach outperformed both FBCSP and EEGNet by 7.48% and 11.04%, respectively (Xin Deng | Boxian Zhang | Nian Yu, 2021)

5. Deliverables:

Developed Deep learning will be able to predict the motor imagery task that a user intends to perform.

The following are the steps which will be implemented to develop a model using deep learning:

1. **Data collection**:
   Collect a dataset of EEG signals from people engaged in various motor imagery tasks using kaggle. The dataset should be extensive and diverse enough to train and test reliable machine learning models.
2. **Data understanding**:
   Perform exploratory data analysis (EDA) to understand the characteristics of the dataset. This involves identifying and resolving missing values, outliers, and other data quality issues. It also entails visualizing the data to identify patterns and trends.
3. **Data preprocessing:**
   To preprocess the data for machine learning, it must be transformed into a machine-learnable format. This consists of feature engineering, scaling, and normalizing the data.
4. **Model training:**
   Train several machine learning models, including ATCNet, to classify the dataset's motor imagery tasks.
5. **Model evaluation:**
   On a held-out test set, evaluate how well the trained machine learning models performed. This involves utilizing measurements like accuracy and the confusion matrix.

6. References:

1. Fazel-Rezai, M. R., Kumar, B. N., & Shenoy, P. (2018). A review of EEG-based BCI applications for neurorehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 26(10), (pp.1851-1865).
2. Hamdi altaheri, Ghulam Muhammad & Mansour Alsulaiman.(2023). Physics Informed attention temporal Convolutional network for EEG based Motor imagery classification. IEEE Transactions on industrial informatics, vol.19, no.2, (pp.2249 – 2258).
3. Hao Zhu, Dylan Forenzo, & Bin He. (2022). On the Deep Learning Models for EEG-Based Brain-Computer Interface Using Motor Imagery. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.30 (pp.2283-2291).
4. Hongkong University of Science and technology. (N.A). Neurofeedback games using EEG based Brain computer interface technology. Retrieved from https://repository.hkust.edu.hk/ir/Record/1783.1-106838
5. Imotions. (2021). What is EEG? Retrieved from https://imotions.com/blog/learning/research-fundamentals/what-is-eeg/
6. Jing Jin, Chang Liu, Ian Daly, &Yangyang Miao. (2020). Bispectrum-Based Channel Selection for Motor Imagery Based Brain-Computer Interfacing. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.28, no.10 (pp.2153 - 2163).
7. Lukasz Radzinski & Tomasz Kocejko. (2022). Deep learning approach on surface EEG based Brain Computer Interface. 15th International Conference on Human System Interaction (HSI), Melbourne, Australia (pp.1-6).
8. Pramod Gaur, Ram Bilas Pachori, Hui Wang & Girijesh Prasad. (2019). An automatic Subject Specific Intrinsic Mode Function Selection for enhancing Two-Class EEG based motor Imagery Brain Computer Interface. IEEE Sensors Journal, vol.19, no.16 (pp.6938-6947).
9. Soroosh Shahtalebi & Amir Asif. (2020). Siamese Neural Networks for EEG Based Brain computer Interfaces. Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Montreal, Canada. (pp.442-446).

10. Tianwei Shi, Ling Ren & Wenhua Cui. (2020). Feature Extraction of Brain–Computer Interface Electroencephalogram Based on Motor Imagery. IEEE Sensors Journal, vol.20, no.20 (pp.11787-11794).
11. V. K. Benzy, A. P. Vinod, R. Subasree & Suvarna Alladi. (2020). Motor Imagery Hand Movement Direction Decoding Using Brain Computer Interface to Aid Stroke Recovery and Rehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol.28, no.12 (pp.3051-3062).
12. Xin Deng, Boxian Zhang, Nian Yu & Ke Liu. (2021). Advanced TSGL-EEGNet for Motor Imagery EEG-Based Brain-Computer Interfaces. IEEE Access, vol.9 (pp.25118-25130).

Appendix B: Ethics Forms (Fast Track)

Office Record	Receipt™ Fast-Track Ethical Approval
Date Received:	Student name: Venkata Krishna Chaitanya Bysani
Received by whom:	Student number: TP062476
	Received by:
	Date:

APU / APIIT FAST-TRACK ETHICAL APPROVAL FORM (STUDENTS)

Tick one box (level of study):

- ☐ POSTGRADUATE (PhD / MPhil / Masters)
- ☑ UNDERGRADUATE (Bachelors degree)
- ☐ FOUNDATION / DIPLOMA / Other categories

Tick one box (purpose of approval):

- ☑ Thesis / Dissertation / FYP project
- ☐ Module assignment
- ☐ Other: _____

Title of Programme on which enrolled – B.sc(Hons) in computer science with specialism Data Analytics

Tick one box: ☑ Full-Time Study or ☐ Part-Time Study

Title of project / assignment - "Efficient Detection of human motion movements through EEG signals using Deep learning".

Name of student researcher … Venkata Krishna Chaitanya Bysani… … … … … … … … … … … … … … …

Name of supervisor / lecturer…… Dr. Mukil Alagirisamy… … … … … … … … … … … … … … … … … …

Student Researchers- please note that certain professional organisations have ethical guidelines that you may need to consult when completing this form.

Supervisors/Module Lecturers - please seek guidance from the Chair of the APU Research Ethics Committee if you are uncertain about any ethical issue arising from this application.

		YES	NO	N/A
1	Will you describe the main procedures to participants in advance, so that they are informed about what to expect?			✓
2	Will you tell participants that their participation is voluntary?			✓
3	Will you obtain written consent for participation?			✓
4	If the research is observational, will you ask participants for their consent to being observed?			✓
5	Will you tell participants that they may withdraw from the research at any time and for any reason?			✓
6	With questionnaires and interviews will you give participants the option of omitting questions they do not want to answer?			✓
7	Will you tell participants that their data will be treated with full confidentiality and that, if published, it will not be identifiable as theirs?			✓
8	Will you give participants the opportunity to be debriefed i.e. to find out more about the study and its results?			✓

If you have ticked **No** to any of Q1-8 you should complete the full Ethics Approval Form.

		YES	NO	N/A
9	Will your project/assignment deliberately mislead participants in any way?		✓	
10	Is there any realistic risk of any participants experiencing either physical or psychological distress or discomfort?		✓	
11	Is the nature of the research such that contentious or sensitive issues might be involved?		✓	

If you have ticked **Yes** to 9, 10 or 11 you should complete the full Ethics Approval Form. In relation to question 10 this should include details of what you will tell participants to do if they should experience any problems (e.g. who they can contact for help). You may also need to consider risk assessment issues.

			YES	NO	N/A
12	Does your project/assignment involve work with animals?			✓	
13	Do participants fall into any of the following special groups? **Note that you may also need to obtain satisfactory clearance from the relevant authorities**	Children (under 18 years of age)			✓
		People with communication or learning difficulties			
		Patients			
		People in custody			
		People who could be regarded as vulnerable			
		People engaged in illegal activities (eg drug taking)			
14	Does the project/assignment involve external funding or external collaboration where the funding body or external collaborative partner requires the University to provide evidence that the project/assignment had been subject to ethical scrutiny?			✓	

If you have ticked **Yes** to 12, 13 or 14 you should complete the full Ethics Approval Form. There is an obligation on student and supervisor to bring to the attention of the APU Research Ethics Committee any issues with ethical implications not clearly covered by the above checklist.

STUDENT RESEARCHER
Provide in the boxes below (plus any other appended details) information required in support of your application, THEN SIGN THE FORM.

	Please Tick Boxes
I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee.	✓
Give a brief description of participants and procedure (methods, tests used etc) in up to 150 words. This project is to monitor the Human Motion movements by using EEG Signals. This Project will use a dataset form Kaggle https://www.kaggle.com/datasets/kgreeshmalakshmi/eeg-based-bci-four-class-wheelchair-application.The dataset consists of EEG data from 9 subjects. Each dataset contains information related to 22 EEG channels, 3 EOG channels. It records four different motor imagery tasks, namely the imagination of movement of the left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4). Convolutional Neural Network (CNN) techniques will be utilized to train the models. Following this, the models will be assessed using a confusion matrix that encompasses precision, accuracy, recall, and other metrics. It's important to note that no surveys or interviews will be conducted as part of the project.	
I also confirm that: i) All key documents e.g. consent form, information sheet, questionnaire/interview are appended to this application.	
Or ii) Any key documents e.g. consent form, information sheet, questionnaire/interview schedules which need to be finalised following initial investigations will be submitted for approval by the project/assignment supervisor/module lecturer before they are used in primary data collection.	✓

E-signature… … … Vishnu Krishna Teja ;……Date…07/11/2023 … …
(Student Researcher)

Please note that any variation to that contained within this document that in any way affects ethical issues of the stated research requires the appending of new ethical details. New ethical consent may need to be sought.

The completed form (and any attachments) should be submitted for consideration by your Supervisor/Module Lecturer

SUPERVISOR/MODULE LECTURER
PLEASE CONFIRM THE FOLLOWING:

	Please Tick Box
I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee	✓
i) I have checked and approved the key documents required for this proposal (e.g. consent form, information sheet, questionnaire, interview schedule) Or	
ii) I have checked and approved draft documents required for this proposal which provide a basis for the preliminary investigations which will inform the main research study. I have informed the student researcher that finalised and additional documents (e.g. consent form, information sheet, questionnaire, interview schedule) must be submitted for approval by me before they are used for primary data collection.	✓

SUPERVISOR AND SECOND ACADEMIC SIGNATORY

STATEMENT OF ETHICAL APPROVAL (please delete as appropriate)

1) **THIS PROJECT/ASSIGNMENT HAS BEEN CONSIDERED USING AGREED APU/SU PROCEDURES AND IS NOW APPROVED**

2) **THIS PROJECT/ASSIGNMENT HAS BEEN APPROVED IN PRINCIPLE AS INVOLVING NO SIGNIFICANT ETHICAL IMPLICATIONS, BUT FINAL APPROVAL FOR DATA COLLECTION IS SUBJECT TO THE SUBMISSION OF KEY DOCUMENTS FOR APPROVAL BY SUPERVISOR (see Appendix A)**

E-signature... ... ... A. Mukti ... ... Date... 08/11/2023 ... ...
(Supervisor/Lecturer)

E-signature... ... ... Print Name... ... ... Date... ... ...
(Second Academic Signatory)

Office Record	Receipt re: Appendix A (Fast-Track Ethics Form)
Date Received: Received by whom:	Student name: Venkata krishna chaitanya Bysani Student number: TP062476 Received by: Date:

APPENDIX A
AUTHORISATION FOR USE OF KEY DOCUMENTS

Completion of Appendix A is required when for good reasons key documents are not available when a fast track application is approved by the supervisor/module lecturer and second academic signatory.

I have now checked and approved all the key documents associated with this proposal e.g. consent form, information sheet, questionnaire, interview schedule

Title of project/assignment... ... "Efficient Detection of human motion movements through EEG signals using Deep learning". ... ...

Name of student researcher ... ... Venkata Krishna Chaitanya Bysani... ... ...

Student ID: .....TP062476..... Intake: ... ...APD3F2308CS(DA)... ... ...

E-signature... ... ... Date... 08/11/2023 ... ...
(Supervisor/Lecturer)

Appendix C: Log Sheets

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory Session

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani Date: 11-10-2023 Meeting No: 1

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature: A. Mukil

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Introduced myself**
2. **Requested to have a look on PPF.**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor informed me to use Microsoft teams for any further discussion on FYP.**
2. **My supervisor suggested new FYP title and informed me to discuss with FYP manager on that.**
3. **My supervisor recommended me to write objectives, problem statements in detailed in PPF.**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **My supervisor told me to send the PPF with the new changes and FYP title then she will give feed back**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory Session

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani **Date: 07-11-2023** **Meeting No: 2**

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature:

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Approval on PPF**
2. **Request to review and sign on Fast track form.**
3. **Ask to check IR report of 3 chapters.**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor told me to send PPF later in teams.**
2. **My supervisor signed on Fast Track form**
3. **My supervisor recommended to improve more in chapter 3 for data preprocessing and write in detailed regarding the machine learning used for the project.**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **Do the changes as per the recommendations of supervisor and ask for review on IR.**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory Session

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani Date: 22-11-2023 Meeting No: 3

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature:

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Ask to check the final IR.**
2. **6 Models comparison is suitable or not**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor had look and asked me to send in team for final check and give feedback via teams**
2. **My supervisor told me to implement the models for comparison**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **Give final touch to the final IR Documentation after getting the feedback.**
2. **Complete and submit it into the moodle**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory | **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani Date: 19-02-2024 **Semester 2 Meeting No:** 1

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature:

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Seek an Review for Investigation Report (IR)**
2. **Ask next steps for the project.**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor informed me to start exploring dataset.**
2. **My supervisor suggested to learn the Deep learning models and implement in project.**
3. **My supervisor recommended me to start Data preprocessing and modelling.**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **EDA, Data preprocessing, Choosing and Developing Models**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory | **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani **Date:** 05-03-2024 **Semester 2 Meeting No: 2**

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature: A. Mukil

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Show the Current progress status**
2. **Request to review regarding the models and their performance.**
3. **Ask about the process of deployment**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor told me to add one more model.**
2. **My supervisor suggested me to deploy using Streamlit or Python Flask Server**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **Add DeepConvNet Model, Complete code for deployment**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory | **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Venkata Krishna Chaitanya Bysani **Date:** 22-04-2024 **Semester 2 Meeting No:** 3

Project title: Efficient Detection of human motion movements through EEG signals using deep learning.
Intake: APD3F2308CSDA

Supervisor's name: Dr. Mukil Alagirisamy

Supervisor's signature: A. Mukil

Items for discussion (noted by student before mandatory supervisory meeting):

1. **Request to do Signature on the forms.**
2. **Review of Completed project**

Record of discussion (noted by student during mandatory supervisory meeting):

1. **My supervisor had look on the documentation.**
2. **My supervisor told me to add final touches to the project and signed on the forms**

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **Give final touch to the final documentation and Make poster as well as record the demo video.**
2. **Complete and submit it into the moodle**

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Appendix D: Poster

Appendix E: Gantt Chart

Appendix F: Sample Code Implementation

FYP - Venkata Krishna Chaitanya Bysani - TP062476 - APD3F2308CSDA.docx

ORIGINALITY REPORT

16%	10%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	2%
2	v3r.esp.org Internet Source	1%
3	repository.essex.ac.uk Internet Source	1%
4	www.frontiersin.org Internet Source	1%
5	Submitted to Bridgepoint Education Student Paper	<1%
6	Hamdi Altaheri, Ghulam Muhammad, Mansour Alsulaiman. "Physics-inform attention temporal convolutional network for EEG-based motor imagery classification", IEEE Transactions on Industrial Informatics, 2022 Publication	<1%
7	Submitted to Ngee Ann Polytechnic Student Paper	<1%

8	explore.openaire.eu Internet Source	<1 %
9	open-innovation-projects.org Internet Source	<1 %
10	VK Benzy, A.P Vinod, R Subasree, Suvarna Alladi, K Raghavendra. "Motor Imagery Hand Movement Direction Decoding using Brain Computer Interface to Aid Stroke Recovery and Rehabilitation", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2020 Publication	<1 %
11	Submitted to Liverpool John Moores University Student Paper	<1 %
12	www.ncbi.nlm.nih.gov Internet Source	<1 %
13	www.coursehero.com Internet Source	<1 %
14	www.journaltocs.ac.uk Internet Source	<1 %
15	Submitted to Melbourne Institute of Technology Student Paper	<1 %
16	Submitted to The Robert Gordon University Student Paper	<1 %

17	www.ijraset.com Internet Source	<1 %
18	www.researchgate.net Internet Source	<1 %
19	Submitted to University of Greenwich Student Paper	<1 %
20	Submitted to University of Sheffield Student Paper	<1 %
21	Xin Deng, Boxian Zhang, Nian Yu, Ke Liu, Kaiwei Sun. "Advanced TSGL-EEGNet for Motor Imagery EEG-Based Brain-Computer Interfaces", IEEE Access, 2021 Publication	<1 %
22	Tianwei Shi, Ling Ren, Wenhua Cui. "Feature Extraction of Brain-computer Interface Electroencephalogram Based on Motor Imagery", IEEE Sensors Journal, 2019 Publication	<1 %
23	medium.com Internet Source	<1 %
24	Hao Zhu, Dylan Forenzo, Bin He. "On The Deep Learning Models for EEG-based Brain-Computer Interface Using Motor Imagery", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2022 Publication	<1 %

25	Submitted to SASTRA University Student Paper	<1%
26	ntnuopen.ntnu.no Internet Source	<1%
27	www.glerl.noaa.gov Internet Source	<1%
28	www.mdpi.com Internet Source	<1%
29	Submitted to University of Southampton Student Paper	<1%
30	c.coek.info Internet Source	<1%
31	dokumen.pub Internet Source	<1%
32	"Innovations in Electrical and Electronic Engineering", Springer Science and Business Media LLC, 2024 Publication	<1%
33	Shujhat Khan, Tipu Aziz. "Transcending the brain: is there a cost to hacking the nervous system?", Brain Communications, 2019 Publication	<1%
34	pure.urosario.edu.co Internet Source	<1%

35	"Neural Information Processing", Springer Science and Business Media LLC, 2017 Publication	<1 %
36	Submitted to BITS, Pilani-Dubai Student Paper	<1 %
37	Submitted to Liberty University Student Paper	<1 %
38	"Proceedings of International Ethical Hacking Conference 2018", Springer Science and Business Media LLC, 2019 Publication	<1 %
39	Bindu Bala, Sunny Behal. "AI techniques for IoT-based DDoS attack detection: Taxonomies, comprehensive review and research challenges", Computer Science Review, 2024 Publication	<1 %
40	iarjset.com Internet Source	<1 %
41	pure.tue.nl Internet Source	<1 %
42	link.springer.com Internet Source	<1 %
43	Submitted to Canadian University of Dubai Student Paper	<1 %

44	Submitted to St. Joseph By The Sea High School Student Paper	<1 %
45	core.ac.uk Internet Source	<1 %
46	www.hindawi.com Internet Source	<1 %
47	umpir.ump.edu.my Internet Source	<1 %
48	Lukasz Radzinski, Tomasz Kocejko. "Deep learning approach on surface EEG based Brain Computer Interface", 2022 15th International Conference on Human System Interaction (HSI), 2022 Publication	<1 %
49	Submitted to University of Houston Clear Lake Student Paper	<1 %
50	ebin.pub Internet Source	<1 %
51	repositorio.unal.edu.co Internet Source	<1 %
52	Submitted to European University of Lefke Student Paper	<1 %
53	Submitted to University of Surrey Student Paper	

		<1 %
54	eprints.ums.edu.my Internet Source	<1 %
55	new.esp.org Internet Source	<1 %
56	vital.seals.ac.za Internet Source	<1 %
57	"Advancement of Machine Intelligence in Interactive Medical Image Analysis", Springer Science and Business Media LLC, 2020 Publication	<1 %
58	M. Anna Latha, E. Sathish, Florence Gnana Poovathy. "Computational Approach to guide Mind Controlled Robotic Arm using BCI – A Review", 2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII), 2021 Publication	<1 %
59	f1000research.com Internet Source	<1 %
60	Submitted to HELP UNIVERSITY Student Paper	<1 %
61	Submitted to University of Central Florida Student Paper	<1 %

62	Xiaojian Liao, Yuli Wu, Zi Wang, Deheng Wang, Hongmiao Zhang. "A Convolutional Spiking Neural Network with Adaptive Coding for Motor Imagery Classification", Neurocomputing, 2023 Publication	<1 %
63	Submitted to Trinity College Dublin Student Paper	<1 %
64	Submitted to University of Westminster Student Paper	<1 %
65	Submitted to Westcliff University Student Paper	<1 %
66	Sándor Beniczky, Donald L. Schomer. "Electroencephalography: basic biophysical and technological aspects important for clinical applications", Epileptic Disorders, 2021 Publication	<1 %
67	Submitted to University of North Texas Student Paper	<1 %
68	opus.lib.uts.edu.au Internet Source	<1 %
69	www.ajbasweb.com Internet Source	<1 %
70	www.pubfacts.com Internet Source	<1 %

71	Submitted to Arts, Sciences & Technology University In Lebanon Student Paper	<1 %
72	Hongyi Zhi, Zhuliang Yu, Tianyou Yu, Zhenghui Gu, Jian Yang. "A Multi-Domain Convolutional Neural Network for EEG-Based Motor Imagery Decoding", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2023 Publication	<1 %
73	Mohammad Rasoul Khalilpour Darzi, Seyed Taghi Akhavan Niaki, Majid Khedmati. "Binary classification of imbalanced datasets: The case of CoIL challenge 2000", Expert Systems with Applications, 2019 Publication	<1 %
74	Tianjun Liu, Deling Yang. "A Densely Connected Multi-Branch 3D Convolutional Neural Network for Motor Imagery EEG Decoding", Brain Sciences, 2021 Publication	<1 %
75	Submitted to University of Nottingham Student Paper	<1 %
76	cse.anits.edu.in Internet Source	<1 %
77	www.doe.k12.de.us Internet Source	<1 %

78	Syed Umar Amin, Hamdi Altaheri, Ghulam Muhammad, Mansour Alsulaiman, Abdul Wadood. "Attention-Inception and Long Short-Term Memory-based Electroencephalography Classification for Motor Imagery Tasks in Rehabilitation", IEEE Transactions on Industrial Informatics, 2021 Publication	<1 %
79	dataanalysis.conferenceseries.com Internet Source	<1 %
80	docs.oracle.com Internet Source	<1 %
81	era.library.ualberta.ca Internet Source	<1 %
82	oaji.net Internet Source	<1 %
83	ouci.dntb.gov.ua Internet Source	<1 %
84	pure.qub.ac.uk Internet Source	<1 %
85	scholarbank.nus.edu.sg Internet Source	<1 %
86	www.mitpressjournals.org Internet Source	<1 %

87	"Intelligent Computing", Springer Science and Business Media LLC, 2023 Publication	<1 %
88	Abbas Salami, Javier Andreu-Perez, Helge Gillmeister. "EEG-ITNet: An Explainable Inception Temporal Convolutional Network for motor imagery classification", IEEE Access, 2022 Publication	<1 %
89	John E. Richards. "Localizing the development of covert attention in infants with scalp event-related potentials.", Developmental Psychology, 2000 Publication	<1 %
90	Muhammed Enes OZELBAS, Emine Elif Tülay, Serhat Ozekes. "Improving Cross-Subject Classification Performance of Motor Imagery Signals: A Data Augmentation-focused Deep Learning Framework", Machine Learning: Science and Technology, 2024 Publication	<1 %
91	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1 %
92	Yuxuan Wei, Jie Li, Hongfei Ji, Lingjing Jin, Lingyu Liu, Zhongfei Bai, Chen Ye. "A Semi-Supervised Progressive Learning Algorithm for Brain Computer Interface", IEEE	<1 %

	Transactions on Neural Systems and Rehabilitation Engineering, 2022 Publication	
93	deepai.org Internet Source	<1 %
94	kiiky.com Internet Source	<1 %
95	vital.seals.ac.za:8080 Internet Source	<1 %
96	vocal.media Internet Source	<1 %
97	www.breakthrough-hrs.co.za Internet Source	<1 %
98	Haodong Deng, Mengfan Li, Jundi Li, Miaomiao Guo, Guizhi Xu. "A robust multi-branch multi-attention-mechanism EEGNet for motor imagery BCI decoding", Journal of Neuroscience Methods, 2024 Publication	<1 %
99	Jiahui Ying, Qingguo Wei, Xichen Zhou. "Riemannian geometry-based transfer learning for reducing training time in c-VEP BCIs", Scientific Reports, 2022 Publication	<1 %
100	dspace.univ-tiaret.dz Internet Source	<1 %

101	eprajournals.com Internet Source	<1 %
102	eprints.qut.edu.au Internet Source	<1 %
103	export.arxiv.org Internet Source	<1 %
104	iieta.org Internet Source	<1 %
105	jeas.springeropen.com Internet Source	<1 %
106	machine-tests.practicetestgeeks.com Internet Source	<1 %
107	qubixity.net Internet Source	<1 %
108	riunet.upv.es Internet Source	<1 %
109	rua.ua.es Internet Source	<1 %
110	westminsterresearch.westminster.ac.uk Internet Source	<1 %
111	www.ijritcc.org Internet Source	<1 %
112	www.piclub.or.jp Internet Source	<1 %

113	Brunner, C.. "Spatial filtering and selection of optimized components in four class motor imagery EEG data using independent components analysis", Pattern Recognition Letters, 20070601 Publication	<1 %
114	Chiranjevulu Divvala, Madhusudhan Mishra. "Deep Learning-Based Attention Mechanism for Automatic Drowsiness Detection Using EEG Signal", IEEE Sensors Letters, 2024 Publication	<1 %
115	Lecture Notes in Computer Science, 2015. Publication	<1 %
116	"HCI International 2022 - Late Breaking Papers. Multimodality in Advanced Interaction Environments", Springer Science and Business Media LLC, 2022 Publication	<1 %
117	Jiyao Liu, Huifu Li. "REEG-BTCNet: A Novel Framework for EEG-based Motor Imagery Classification", 2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2022 Publication	<1 %
118	S. Aghaei, Amirhossein, Mohammad Shahin Mahanta, and Konstantinos Plataniotis. "Separable Common Spatio-Spectral Patterns for Motor Imagery BCI Systems", IEEE	<1 %

Transactions on Biomedical Engineering, 2015.
Publication

119 V. K. Benzy, A. P. Vinod, R. Subasree, Suvarna Alladi, K. Raghavendra. "Motor Imagery Hand Movement Direction Decoding Using Brain Computer Interface to Aid Stroke Recovery and Rehabilitation", IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2020 <1 %
Publication

120 Xinghe Xie, Liyan Chen, Shujia Qin, Fusheng Zha, Xinggang Fan. "Bidirectional feature pyramid attention-based temporal convolutional network model for motor imagery electroencephalogram classification", Frontiers in Neurorobotics, 2024 <1 %
Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

FYP - Venkata Krishna Chaitanya Bysani - TP062476 - APD3F2308CSDA.docx

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96

PAGE 97

PAGE 98

PAGE 99

PAGE 100

PAGE 101

PAGE 102

PAGE 103

PAGE 104

PAGE 105

PAGE 106

PAGE 107

PAGE 108

PAGE 109

PAGE 110

PAGE 111

PAGE 112

PAGE 113

PAGE 114

PAGE 115

PAGE 116

PAGE 117

PAGE 118

PAGE 119

PAGE 120

PAGE 121

PAGE 122

PAGE 123

PAGE 124

PAGE 125

PAGE 126

PAGE 127

PAGE 128

PAGE 129

PAGE 130

PAGE 131

PAGE 132

PAGE 133

PAGE 134

PAGE 135

PAGE 136

PAGE 137

PAGE 138

PAGE 139

PAGE 140