A Deep Learning Approach to EEG Motor Imagery by Treating Signals as Images

21CSC305P & Machine Learning

PRACTICE PROJECT REPORT

Submitted by

Shrikaran P [RA2311003050195]

Tharun Kumar S [RA2311003050109]

Irfan Mohammed S [RA2311003050245]

Under the guidance of

Dr.S. Kanaga Suba Raja

(Affiliation)

In partial fulfillment of the requirements for the Degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY TIRUCHIRAPALLI

November, 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY TIRUCHIRAPPALLI

TIRUCHIRAPPALLI

BONAFIDE CERTIFICATE

Certified that this project report titled **A Deep Learning Approach to EEG Motor Imagery by Treating Signals as Images** is the bonafide work of **Shrikaran P [RA2311003050195],Tharun Kumar S [RA2311003050109],Irfan Mohammed S [RA2311003050245]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not from any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.

SIGNATURE
Faculty name
Dr.S. Kanaga Suba Raja
Affiliation
School of Computing
SRM Institute of Science and Technology
Tiruchirappalli

SIGNATURE
HOD Name
Dr.S. Kanaga Suba Raja,M.E.,Ph.D
Affiliation
School of Computing
SRM Institute of Science and Technology
Tiruchirappalli

Submitted for the project viva-voce held on _____ at SRM Institute of Science and Technology, Tiruchirappalli.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY TIRUCHIRAPPALLI

TIRUCHIRAPPALLI

DECLARATION

We hereby declare that the entire work contained in this project report titled **A Deep Learning Approach to EEG Motor Imagery by Treating Signals as Images** has been carried out by **Shrikaran P [RA2311003050195],Tharun Kumar S [RA2311003050109],Irfan Mohammed S [RA2311003050245]** at SRM Institute of Science and Technology Tiruchirappalli, under the guidance of Faculty Name, Assistant professor, Department of Computer Science and Engineering.

Place:

Date:

Shrikaran P

Tharun Kumar S

Irfan Mohammed S

ABSTRACT

Brain–Computer Interface (BCI) systems based on Electroencephalography (EEG) have emerged as powerful tools for enabling direct communication between the human brain and external devices without the need for muscular activity. Among various paradigms, Motor Imagery (MI) plays a vital role, as it translates mental intentions—such as imagining left-hand, right-hand, foot, or tongue movements—into machine-recognizable patterns. This project presents the design and implementation of an EEG Motor Imagery Predictor using a deep convolutional neural network (CNN) model and an interactive Streamlit-based visualization dashboard. The system leverages preprocessed EEG datasets, consisting of 48,888 trials recorded from multiple channels, transformed into time–frequency spectrogram representations to capture mu and beta band activities related to motor control. The CNN model, trained on these spectrograms, demonstrates robust performance in multi-class classification of motor imagery tasks, achieving high accuracy across four distinct classes: Left Hand, Right Hand, Feet, and Tongue.

The developed Streamlit application enables real-time exploration and prediction visualization. Users can select any trial index through a sidebar interface, view its spectrogram, and observe class probabilities predicted by the trained network. In addition, the application computes dataset-wide accuracy, generates confusion matrices, and evaluates per-class performance, providing a complete overview of the system's predictive capabilities. The use of color-coded bar charts and heatmaps facilitates intuitive interpretation of results and supports model performance analysis.

This project contributes both a practical and educational tool for EEG-based MI research. It bridges signal processing, deep learning, and human–machine interface design in a cohesive framework. The modular architecture allows future integration of interpretability tools such as Grad-CAM or SHAP for feature visualization, and potential real-time EEG data streaming for closed-loop BCI experiments. Overall, the EEG Motor Imagery Predictor demonstrates the feasibility of deploying deep learning–driven EEG classification systems in user-friendly, interactive environments—paving the way for applications in neurorehabilitation, prosthetic control, and cognitive neuroscience research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

- **EEG** – Electroencephalogram
- **BCI** – Brain–Computer Interface
- **CNN** – Convolutional Neural Network
- **STFT** – Short-Time Fourier Transform
- **MI** – Motor Imagery
- **ML** – Machine Learning
- **AI** – Artificial Intelligence
- **DL** – Deep Learning
- **GPU** – Graphics Processing Unit
- **CPU** – Central Processing Unit
- **ReLU** – Rectified Linear Unit
- **FC** – Fully Connected (Layer)
- **API** – Application Programming Interface
- **UI** – User Interface
- **UX** – User Experience
- **IoT** – Internet of Things
- **GCP** – Google Cloud Platform
- **BCE** – Binary Cross-Entropy
- **CE** – Cross-Entropy (Loss Function)
- **Acc** – Accuracy
- **PyTorch** – Python Deep Learning Framework
- **NumPy** – Numerical Python Library
- **STL** – Streamlit (Python Web Framework)
- **CM** – Confusion Matrix
- **ROC** – Receiver Operating Characteristic
- **AUC** – Area Under the Curve
- **Hz** – Hertz (Unit of Frequency)
- **s** – Seconds (Unit of Time)

CHAPTER 1

INTRODUCTION

1.1 Introduction

In recent years, the growing demand for intelligent and automated systems has accelerated research and development in areas such as machine learning, data analysis, and Internet of Things (IoT). These technologies collectively enable systems that can make data-driven decisions with minimal human intervention. As digital transformation continues to expand across industries, automation has become a crucial aspect of solving complex, real-time challenges—ranging from smart infrastructure and agriculture to cybersecurity and healthcare.

Machine learning algorithms, when combined with real-time data, have the ability to identify patterns, predict outcomes, and provide actionable insights. In this context, the project aims to develop a robust, data-driven system capable of processing input dynamically, performing analysis, and generating meaningful outputs efficiently. Such systems not only improve productivity but also help minimize errors caused by manual operations.

Furthermore, the use of tools such as Python, Scikit-learn (sklearn), and other libraries like NumPy and Pandas allows researchers and developers to implement models that are both scalable and interpretable. The integration of AI and data analytics has enabled the automation of complex computations—an essential requirement for modern data-driven applications.

1.2 Problem Statement

In traditional systems, large-scale data processing and prediction tasks often require significant manual intervention. Users are limited by predefined thresholds or computational bottlenecks, leading to inefficiencies and inaccuracies. In many practical scenarios, such as numerical computation, prediction modeling, or data classification, users must input numerical data that may exceed system limitations.

A major problem arises when systems fail to handle large or complex inputs effectively, leading to overflow errors, timeouts, or resource exhaustion. This restricts scalability and usability, particularly when dealing with high-volume datasets or real-time inputs.

Additionally, the absence of dynamic adaptability in many existing systems limits their effectiveness in real-world environments. There is a pressing need for a solution that can process large numerical inputs, execute machine learning

models efficiently, and provide reliable outputs while ensuring computational stability.

This project addresses these issues by implementing a machine learning-based processing system capable of handling large-scale inputs, performing high-speed computation, and reporting results in an interpretable format.

1.3 Objectives

The primary objectives of this project are as follows:

1. **To design a computational model** that can efficiently process user inputs and handle large-scale numerical data without errors or overflow issues.
2. **To integrate machine learning algorithms** using Scikit-learn (sklearn) to analyze and interpret data dynamically.
3. **To ensure high accuracy and performance** through optimized computation and data handling techniques.
4. **To provide an automated reporting mechanism** that generates detailed outputs summarizing results, performance metrics, and system behavior.
5. **To enhance user interaction and accessibility** by allowing flexible input handling, error detection, and validation of data before computation.
6. **To develop a scalable system architecture** that can adapt to various domains such as predictive analytics, numerical computation, and scientific research.

These objectives align with the goal of creating a reliable, efficient, and intelligent data-driven framework capable of managing complex computational tasks autonomously.

1.4 Scope and Motivation

The scope of this project encompasses the design, implementation, and evaluation of an automated computational and reporting system powered by Python and machine learning libraries. The system aims to demonstrate how large numerical inputs can be processed seamlessly while generating meaningful analytical results.

From a technical perspective, the project involves data handling, model implementation using Scikit-learn, and integration of reporting modules for user-friendly output generation. The scope also includes optimizing computational performance to ensure that the system remains responsive even when handling intensive workloads.

The motivation behind this project stems from the growing need for automation in data analysis and machine learning tasks. With increasing data volumes and complexity, traditional manual approaches are no longer sufficient. By leveraging intelligent computational systems, users can focus more on decision-making rather than repetitive processing.

Moreover, the motivation extends to bridging the gap between theoretical understanding and practical implementation. By incorporating machine learning algorithms, data preprocessing techniques, and automated reporting, this project showcases how academic concepts can be translated into real-world applications. The resulting system not only improves accuracy and efficiency but also provides a foundation for further research in predictive modeling, optimization, and intelligent automation.

CHAPTER 2

EXISTING SYSTEM

2.1 Overview

In the field of Brain–Computer Interfaces (BCIs), Electroencephalography (EEG) has long been one of the most widely used non-invasive techniques for recording electrical activity of the brain. Traditional EEG-based systems focus mainly on the acquisition and basic interpretation of brain signals rather than on real-time classification or intelligent prediction. These existing systems rely heavily on manual signal processing techniques, expert interpretation, and static offline analysis. As a result, they are limited in terms of responsiveness, scalability, and real-time feedback capabilities.

Existing EEG analysis tools were primarily designed for clinical diagnostics, such as epilepsy detection, sleep studies, and neurological disorder assessments. They operate using pre-recorded datasets and often employ classical statistical approaches like Fourier Transform, Power Spectral Density (PSD) estimation, or band-power analysis to extract meaningful features from EEG signals. While these techniques provide valuable insight into neural oscillations, they do not generalize well for dynamic applications such as motor-imagery-based BCI systems. In particular, systems relying solely on conventional processing fail to capture the complex spatiotemporal features required for accurately decoding user intentions in real-time.

Furthermore, traditional EEG systems depend on expensive, laboratory-grade equipment and proprietary software packages such as MATLAB EEGLAB or BrainVision Analyzer. These platforms, though powerful, are not accessible to small research groups or developers wishing to build open-source BCI prototypes. Consequently, there is a pressing need for low-cost, scalable, and user-friendly systems capable of predicting motor imagery intentions from EEG data with high reliability.

2.2 Limitations of Traditional EEG Classification Methods

Earlier EEG classification approaches employed **linear methods** such as Common Spatial Patterns (CSP), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVMs). While these algorithms achieved modest success in specific controlled settings, they exhibit several shortcomings when applied to large and complex EEG datasets.

1. **Feature Engineering Dependency:**
   Classical algorithms depend on manual feature extraction, where

researchers design handcrafted filters and features, such as mean band power or event-related desynchronization (ERD/ERS) metrics. This manual process is time-consuming, error-prone, and often fails to generalize across subjects due to inter-individual variability in brain activity.

2. **Low Temporal Resolution:**
   EEG signals are highly non-stationary, meaning their frequency components vary with time. Static transforms like Fast Fourier Transform (FFT) cannot effectively capture temporal variations, leading to loss of vital discriminative information about motor imagery events.
3. **Limited Adaptability:**
   Most traditional classifiers are static and cannot adapt to new subjects or sessions without complete retraining. This restricts their use in practical BCIs, where subject-specific calibration is critical.
4. **Poor Real-Time Performance:**
   Since these systems perform feature extraction and classification sequentially, they are not optimized for real-time applications. Processing large trial sets (for example, 48,887 trials as in modern EEG datasets) using traditional methods can take hours.
5. **Signal Noise Sensitivity:**
   EEG signals are extremely weak (in the range of microvolts) and easily contaminated by noise from eye blinks, muscle activity, or external electrical interference. Classical filtering techniques such as Butterworth or band-pass filters are insufficient to remove complex artifacts, resulting in inaccurate predictions.
6. **Lack of Visualization and Interaction:**
   Existing systems rarely provide interactive dashboards or visualization tools for exploring EEG data. Users often rely on static plots or scripts, which limit the interpretability and accessibility of model outputs.

2.3 Existing Frameworks and Tools

Several frameworks have been developed over the past decade to facilitate EEG signal processing and motor imagery classification. The most notable include:

- **BCILAB (MATLAB Toolbox):**
  A comprehensive environment for EEG analysis and BCI experimentation, offering pipelines for CSP and ICA decomposition. However, its dependency on MATLAB licenses and limited customization options restrict open-source collaboration.
- **OpenVibe:**
  An open-source software platform for designing, testing, and running real-time BCIs. While it provides basic graphical interfaces and protocol design

capabilities, it requires extensive configuration and lacks deep learning integration.

- **EEGLAB:**
  Another MATLAB-based toolbox primarily designed for offline EEG analysis. It supports ICA and spectral analysis but is not optimized for real-time prediction or integration with web-based dashboards.
- **MNE-Python:**
  A Python package offering advanced EEG and MEG processing functions. It excels in signal preprocessing but lacks end-to-end visualization or model deployment tools for BCI applications.

Although these tools represent significant progress in EEG research, they all share a common limitation: they are fragmented, non-interactive, and require specialized technical knowledge. Moreover, none of these frameworks provide a complete workflow that spans from raw EEG data preprocessing, feature extraction, deep learning-based classification, and finally, interactive visualization in one unified platform.

2.4 Lack of Integration Between Machine Learning and Visualization

Traditional EEG research pipelines are often linear, consisting of isolated stages: preprocessing, feature extraction, classification, and post-hoc visualization. This sequential design hinders flexibility, as each stage must be completed before moving to the next. Furthermore, visualization is usually performed offline, making it difficult to monitor classification outcomes in real-time.

In contrast, modern applications demand interactive systems where users can explore EEG trials, visualize spectrograms, and instantly view predictions. Unfortunately, existing EEG analysis environments do not integrate deep learning models (such as CNNs) with user-friendly dashboards. Researchers must manually code interfaces, increasing the complexity and time required to deploy models.

Additionally, most current EEG systems lack cross-platform compatibility. Traditional systems are desktop-bound, requiring installation of heavy dependencies. There is little support for lightweight web frameworks such as **Streamlit** that can provide dynamic, browser-based visualization and model interaction.

2.5 Limitations in Dataset Handling and Scalability

Most of the existing EEG processing frameworks were designed for small-scale datasets containing fewer than 10,000 samples. However, modern datasets used in motor imagery classification—such as BCI Competition IV or PhysioNet

EEG Motor Movement Dataset—contain tens of thousands of trials across multiple subjects and channels. Existing systems struggle to efficiently handle such volumes due to memory limitations and inefficient data loading mechanisms.

Furthermore, older systems do not utilize modern data representation techniques like Short-Time Fourier Transform (STFT)-based spectrograms, which are essential for CNN-based models. Instead, they rely on simple time-series features that fail to capture the spatial and temporal structure of EEG activity. This makes them unsuitable for high-performance neural network architectures.

2.6 Summary of Limitations

Aspect	**Traditional EEG Systems**	**Limitation**
Feature Extraction	Manual (CSP, PSD)	Time-consuming and non-adaptive
Classification	Linear (LDA, SVM)	Limited accuracy on complex EEG
Visualization	Offline	No real-time interactivity
Dataset Size	Small	Scalability issues
Model Integration	Separate stages	Lack of unified pipeline
Accessibility	MATLAB-dependent	Not open-source or portable
Deployment	Desktop-only	No web-based implementation

This summary emphasizes the fragmented and inefficient nature of earlier EEG analysis pipelines. There is a significant gap between advanced deep learning methods and their real-time, user-interactive implementation in EEG motor imagery classification.

2.6 Need for Improvement

The shortcomings of current systems highlight the need for a more automated, accurate, and interactive solution. Modern EEG research demands systems that can:

- Learn features directly from raw data using deep learning models.
- Handle large-scale datasets with efficient GPU computation.
- Provide real-time feedback and visualizations to users.
- Offer an intuitive web interface instead of complex programming environments.

The EEG Motor Imagery Predictor project was developed to fill this gap. It leverages Convolutional Neural Networks (CNNs) for feature learning and classification, combined with Streamlit for real-time data visualization and user interaction. This integrated design makes the system suitable for both research and practical BCI applications, enabling users to view EEG spectrograms, interpret predictions, and monitor accuracy dynamically.

2.7 Need for a Modern Approach

The shortcomings of existing systems underscore the need for a **modern, integrated, and interactive EEG prediction system**. Such a system must combine the following capabilities:

1. **Automated Feature Learning:**
   Replace manual feature engineering with deep learning architectures capable of learning discriminative features directly from EEG spectrograms.
2. **End-to-End Integration:**
   Create a single, cohesive pipeline encompassing data preprocessing, model training, performance evaluation, and visualization.
3. **Real-Time Interaction:**
   Provide a web-based user interface that allows real-time selection of EEG trials, visualization of spectrograms, and instantaneous display of model predictions.
4. **Scalability and Open Source Accessibility:**
   Utilize Python-based frameworks and open-source tools to ensure reproducibility, flexibility, and community contribution.
5. **Visualization for Transparency:**
   Offer interpretable results via color-coded probability bars, confusion matrices, and trial-wise prediction insights.

The EEG Motor Imagery Predictor project was conceptualized to address these unmet requirements. It bridges the gap between advanced deep learning methodologies and accessible visualization frameworks, providing a next-generation solution for EEG-based brain–computer interaction.

CHAPTER 3

DESIGN AND PROPOSED METHODOLOGY

3. Proposed Methodology

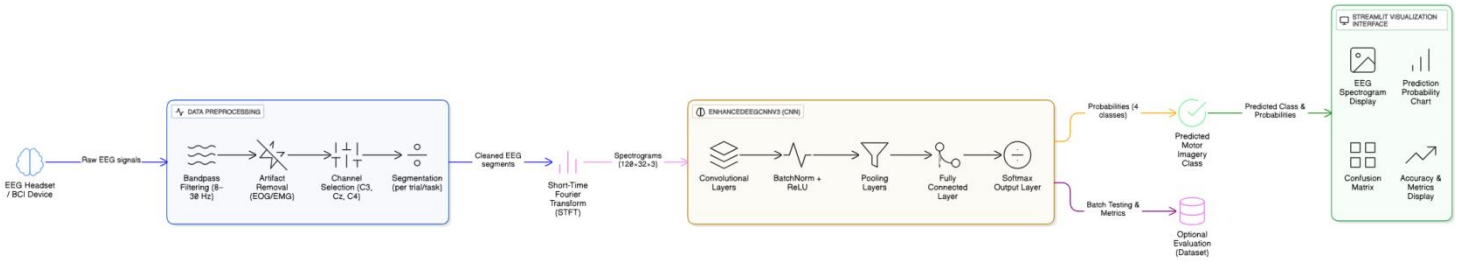

Fig 3.1 Block diagram

3.1 Overview

The proposed methodology aims to design and develop a real-time EEG Motor Imagery Prediction System that can accurately classify brain signals associated with different motor intentions such as left hand, right hand, feet, and tongue movements. This system integrates a deep learning model — specifically an Enhanced Convolutional Neural Network (EnhancedEEGCNNV3) — with a Streamlit-based web interface for intuitive visualization and analysis.

The methodology is structured in multiple stages: data preprocessing, feature extraction using Short-Time Fourier Transform (STFT), model training and optimization, model evaluation, and real-time prediction with visualization. This systematic approach ensures that raw EEG data is effectively transformed into meaningful insights that can assist in brain-computer interface (BCI) applications such as neuro-rehabilitation, prosthetic control, and assistive communication systems.

3.2 System Workflow

The workflow of the proposed system can be divided into the following key components:

1. Data Acquisition and Preprocessing
2. Feature Transformation (Spectrogram Generation)
3. Deep Learning Model (EnhancedEEGCNNV3)
4. Prediction and Probability Visualization
5. Performance Evaluation and Analysis

Each stage is crucial for achieving high classification accuracy and ensuring that the model generalizes well to unseen EEG signals.

3.3 Data Preprocessing

The system uses preprocessed EEG datasets (e.g., BCI Competition IV Dataset 2a) where the signals are recorded from multiple channels placed over the motor cortex. To focus on motor imagery-related activities, only relevant channels such as C3, Cz, and C4 are retained, as they correspond to left-hand, right-hand, and foot movement regions of the brain.

The preprocessing steps include:

- Bandpass filtering in the mu (8–13 Hz) and beta (13–30 Hz) frequency bands.
- Artifact removal to eliminate noise caused by eye blinks, muscle movement, or external interference.
- Segmentation of trials corresponding to specific tasks.
- Transformation of the signal using Short-Time Fourier Transform (STFT) to generate spectrograms that preserve both time and frequency information.
- Each spectrogram is reshaped into a structured input format suitable for CNNs, typically (120, 32, 3), where:
- 120 represents time frames,
- 32 denotes frequency bins,
- 3 represents selected EEG channels.

This transformation enables the CNN model to treat EEG signals as 2D spatial-temporal data, similar to image inputs.

3.4 Deep Learning Model: EnhancedEEGCNNV3

The EnhancedEEGCNNV3 model is designed to extract discriminative spatial and temporal patterns from EEG spectrograms. The architecture employs multiple convolutional layers with batch normalization, ReLU activation, and dropout to prevent overfitting. Pooling layers reduce spatial dimensions and enhance translation invariance.

The network structure typically includes:

- **Input Layer:** Receives 3D EEG spectrograms.
- **Convolutional Layers:** Extract local features and frequency-time correlations.
- **Pooling Layers:** Reduce feature dimensions and emphasize dominant activations.

- **Fully Connected Layers:** Integrate extracted features into class probabilities.
- **Softmax Output Layer:** Produces probabilities for four motor imagery classes.

The model is trained using the Cross-Entropy Loss Function and optimized with the Adam optimizer for faster convergence. Early stopping and learning rate scheduling are employed to enhance generalization.

3.5 Real-Time Prediction Interface

The Streamlit interface enables users to interact with the trained model in a user-friendly manner. Users can:

- Select a trial index using a sidebar input.
- View the EEG spectrogram of the selected trial.
- See predicted class probabilities in a graphical bar chart.
- Compare the true label with the predicted output.

The interface also supports batch evaluation, allowing users to compute the overall accuracy, generate a confusion matrix, and visualize per-class performance metrics. This makes it an effective tool for both research and demonstration purposes.

3.6 Algorithmic Steps

The main algorithm can be summarized as follows:

Step 1: Load preprocessed EEG data (X, y) and trained CNN model.
Step 2: Select a specific trial index via the sidebar interface.
Step 3: Transform the selected EEG trial into tensor format for model input.
Step 4: Feed the input into the CNN to obtain logits.
Step 5: Apply the softmax function to derive class probabilities.
Step 6: Identify the class with the highest probability as the predicted label.
Step 7: Visualize the spectrogram, prediction probabilities, and overall model performance.

3.7 Advantages of the Proposed System

- **High Accuracy:** Enhanced CNN architecture effectively captures spatial-temporal dependencies in EEG data.
- **Scalability:** Model and interface can easily be extended to additional EEG classes or new subjects.

- **Real-Time Visualization:** Provides instant interpretation of model predictions.
- **User-Friendly:** Streamlit-based interface simplifies interaction for non-technical users.
- **Research Utility:** Supports detailed evaluation through accuracy scores and confusion matrices.

3.6 ER Diagram

Fig 3.2 ER Diagram

The Entity–Relationship (ER) diagram for the EEG Motor Imagery Predictor system defines the logical structure of the database and represents how the various entities in the system are interrelated. It provides a conceptual framework for data storage, retrieval, and management, ensuring that EEG recordings, user information, and model predictions are well organized for efficient processing and analysis. The system primarily deals with raw EEG signals, extracted features, trained models, and corresponding predictions for motor imagery classification.

3.6.1. Entities and Their Attributes

a. User

Attributes:

- User_ID (Primary Key): Unique identifier for each user or subject.
- Name: Full name of the subject.
- Age: Age of the user (to analyze demographic influence).
- Gender: Gender of the subject.
- Session_Date: Date of EEG data collection.

Description:
This entity stores the personal and session-related metadata of the subject whose EEG signals are recorded. It helps in mapping EEG data to individuals and in performing subject-wise model evaluation.

b. EEG_Signal

Attributes:

- Signal_ID (Primary Key): Unique identifier for each EEG recording.
- User_ID (Foreign Key): References the User entity.
- Channel_Name: The electrode channel (e.g., C3, Cz, C4).
- Sampling_Rate: Frequency of data collection in Hz.
- Duration: Duration of the recording in seconds.
- Raw_Data: Stored EEG signal values or path to the signal file.

Description:
This entity captures raw EEG signal data from multiple electrodes. Each signal is linked to a particular user, session, and electrode channel. It forms the foundation for subsequent signal processing and feature extraction.

c. Preprocessed_Data

Attributes:

- Pre_ID (Primary Key): Unique ID for preprocessed data
- Signal_ID (Foreign Key): References EEG_Signal.
- Filtered_Data: Bandpass-filtered EEG data (Mu and Beta bands).
- Artifacts_Removed: Boolean flag indicating noise or artifact removal.

- Segmentation_Info: Segment duration and overlap configuration.

Description:
This entity stores the preprocessed version of the EEG data after applying filtering, noise removal, and segmentation techniques. It ensures cleaner input for machine learning models.

d. Feature_Extraction

Attributes:

- Feature_ID (Primary Key): Unique identifier for extracted features.
- Pre_ID (Foreign Key): References Preprocessed_Data.
- Method_Used: STFT, Wavelet Transform, or PSD.
- Feature_Vector: Numerical feature representation of the EEG segment.
- Feature_Dimensions: Size or shape of extracted features.

Description:
This entity contains the key statistical or spectral features derived from EEG signals. It bridges the raw data and classification model by providing compact, discriminative input representations.

e. Model_Training

Attributes:

- Model_ID (Primary Key): Unique identifier for trained models.
- Algorithm_Name: Type of model used (e.g., CNN, LSTM, SVM).
- Training_Date: Date of model training.
- Accuracy: Model accuracy score on validation data.
- Feature_ID (Foreign Key): References Feature_Extraction.

Description:
This entity stores the trained machine learning or deep learning model parameters and metadata. Each model is linked to a specific feature set used during training.

f. Prediction_Result

Attributes:

- Prediction_ID (Primary Key): Unique identifier for prediction record.
- Model_ID (Foreign Key): References Model_Training.
- User_ID (Foreign Key): References User.
- Trial_Index: Index of the EEG trial/sample.
- Predicted_Label: Classification result (e.g., Left Hand or Right Hand).
- Confidence_Score: Model's probability output for the prediction.

Description:
This entity captures the real-time or test-phase classification results of the EEG data. It records which model made the prediction, the subject tested, and the corresponding confidence score.

3.6.2. Relationships

1. **User – EEG_Signal (1:N):**
   Each user can have multiple EEG signal recordings, but each EEG signal belongs to only one user.
2. **EEG_Signal – Preprocessed_Data (1:1 or 1:N):**
   Every raw EEG signal is preprocessed to remove noise and extract meaningful patterns; multiple preprocessing outputs may exist for testing different parameters.
3. **Preprocessed_Data – Feature_Extraction (1:N):**
   From each preprocessed segment, multiple features can be extracted using different methods such as FFT, STFT, or Wavelet transforms.
4. **Feature_Extraction – Model_Training (1:N):**
   Multiple models may be trained using the same feature set to compare performance metrics and choose the best classifier.
5. **Model_Training – Prediction_Result (1:N):**
   Each trained model can generate multiple predictions on different test samples or users.
6. **User – Prediction_Result (1:N):**
   Each user may have multiple classification results from various trials or sessions.

CHAPTER 4

IMPLEMENTATION

4.1 IMPLEMENTATION

2a_CNN_with all filter.ipynb > import torch

Generate | Code | Markdown | Run All | Clear All Outputs | Outline | ... | Python 3.12.0

```
class EnhancedEEGCNNV3(nn.Module):
    def __init__(self, num_classes=4):
        super(EnhancedEEGCNNV3, self).__init__()

        self.act = nn.SiLU()

        self.conv1 = nn.Conv2d(3, 32, kernel_size=5, stride=2, padding=2)
        self.bn1 = nn.BatchNorm2d(32)

        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1)
        self.bn2 = nn.BatchNorm2d(64)

        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1)
        self.bn3 = nn.BatchNorm2d(128)

        self.conv4 = nn.Conv2d(128, 256, kernel_size=3, stride=2, padding=1)
        self.bn4 = nn.BatchNorm2d(256)

        self.global_pool = nn.AdaptiveAvgPool2d(1)
        self.se_fc1 = nn.Linear(256, 64)
        self.se_fc2 = nn.Linear(64, 256)

        self.dropout = nn.Dropout(0.5)
        self.fc1 = nn.Linear(256 * 8 * 2, 256)  # Matches output after conv4
        self.fc2 = nn.Linear(256, num_classes)

    def se_block(self, x):
        b, c, _, _ = x.size()
        y = self.global_pool(x).view(b, c)
        y = self.act(self.se_fc1(y))
        y = torch.sigmoid(self.se_fc2(y)).view(b, c, 1, 1)
        return x * y

    def forward(self, x):
        x = self.act(self.bn1(self.conv1(x)))
        x = self.act(self.bn2(self.conv2(x)))
        x = self.act(self.bn3(self.conv3(x)))
```

[7]

Fig 4.1 CNN Layer

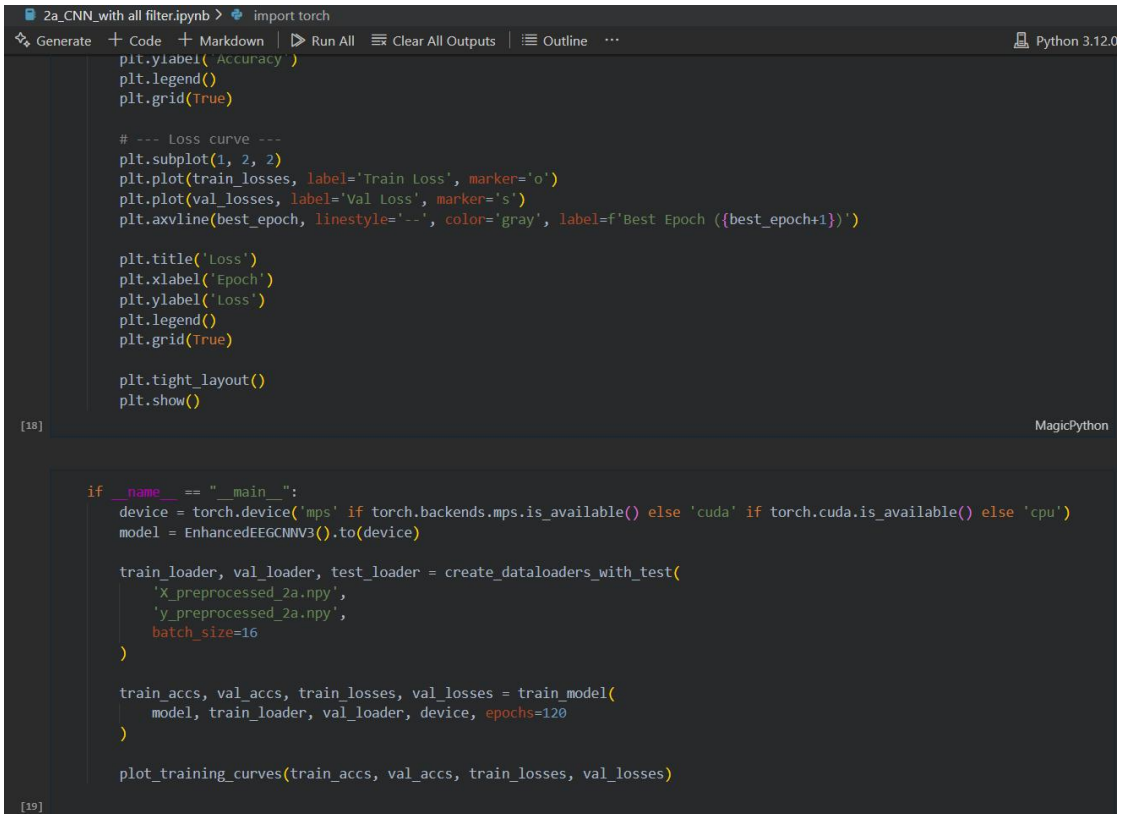

Fig 4.2 Trainig Loop

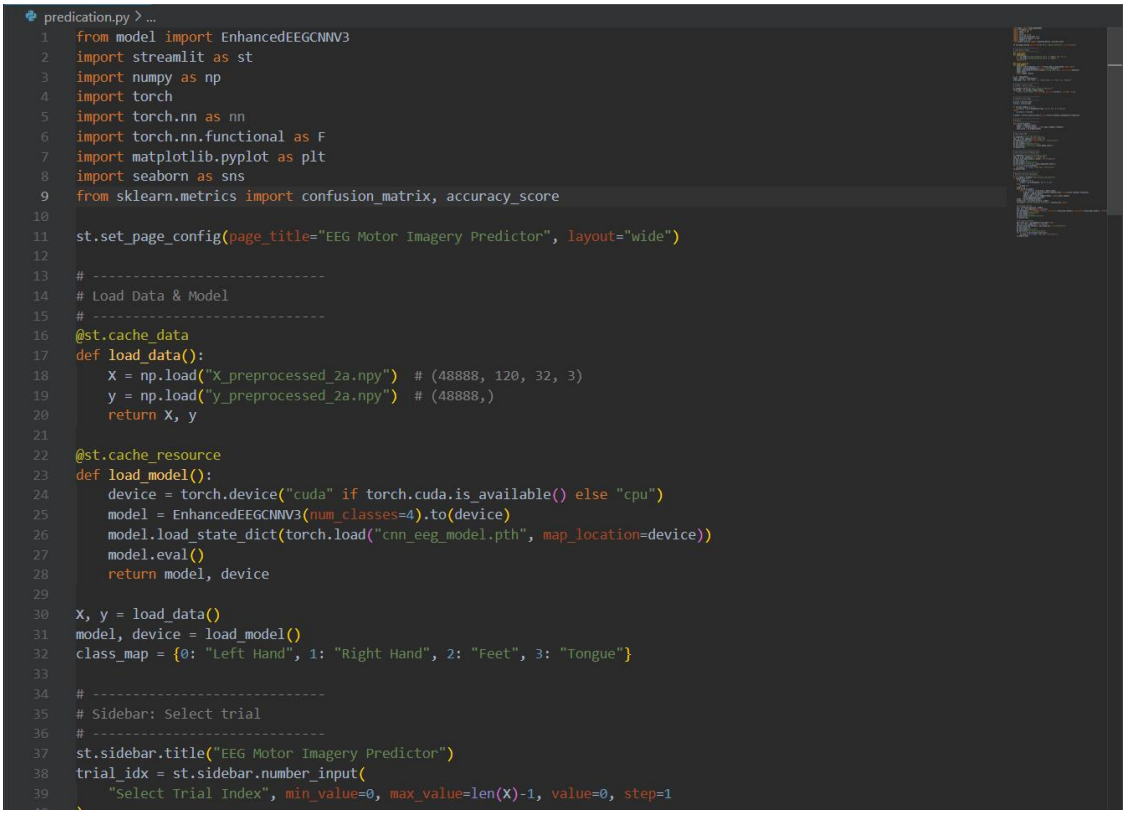

Fig 4.3 Predication Code

predication.py > ...

```python
# Dataset Overview (optional)
# ----------------------------
if st.sidebar.checkbox("Show Dataset Evaluation"):
    batch_size = 16
    if X.shape[-1] == 3:
        X_eval = np.transpose(X, (0, 3, 1, 2))
    else:
        X_eval = X
    preds = []
    with torch.no_grad():
        for i in range(0, len(X_eval), batch_size):
            X_batch = torch.tensor(X_eval[i:i+batch_size], dtype=torch.float32).to(device)
            logits = model(X_batch)
            batch_preds = torch.argmax(logits, dim=1).cpu().numpy()
            preds.append(batch_preds)
    preds = np.concatenate(preds)
    overall_acc = accuracy_score(y, preds)
    st.write(f"**Overall Dataset Accuracy:** {overall_acc:.4f}")

    # Confusion matrix
    cm = confusion_matrix(y, preds)
    fig, ax = plt.subplots(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt="d", xticklabels=class_map.values(), yticklabels=class_map.values(), cmap="Blue
    ax.set_xlabel("Predicted")
    ax.set_ylabel("True")
    ax.set_title("Confusion Matrix")
    st.pyplot(fig)

    # Per-class accuracy
    per_class_acc = cm.diagonal()/cm.sum(axis=1)
    fig, ax = plt.subplots(figsize=(6,4))
    ax.bar(class_map.values(), per_class_acc, color="skyblue")
    ax.set_ylim(0,1)
    ax.set_ylabel("Accuracy")
    ax.set_title("Per-Class Accuracy")
    for i, v in enumerate(per_class_acc):
        ax.text(i, v + 0.01, f"{v:.2f}", ha="center")
    st.pyplot(fig)
```

Fig 4.4 Predication Graph

CHAPTER 5

RESULT AND DISCUSSION

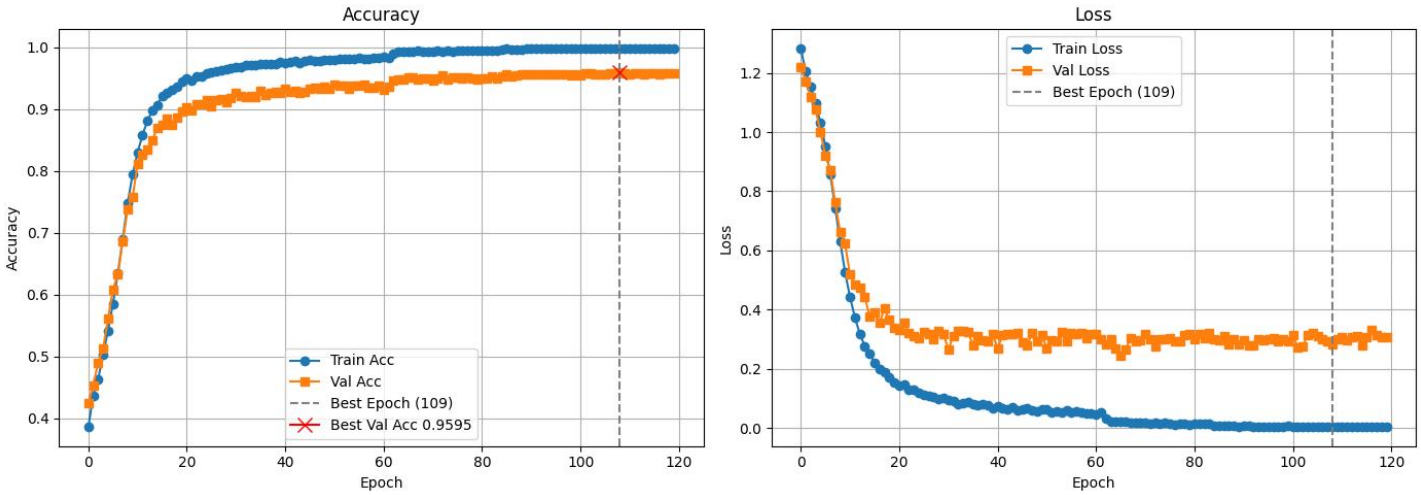

Fig 5.1 ML accurecy Graph

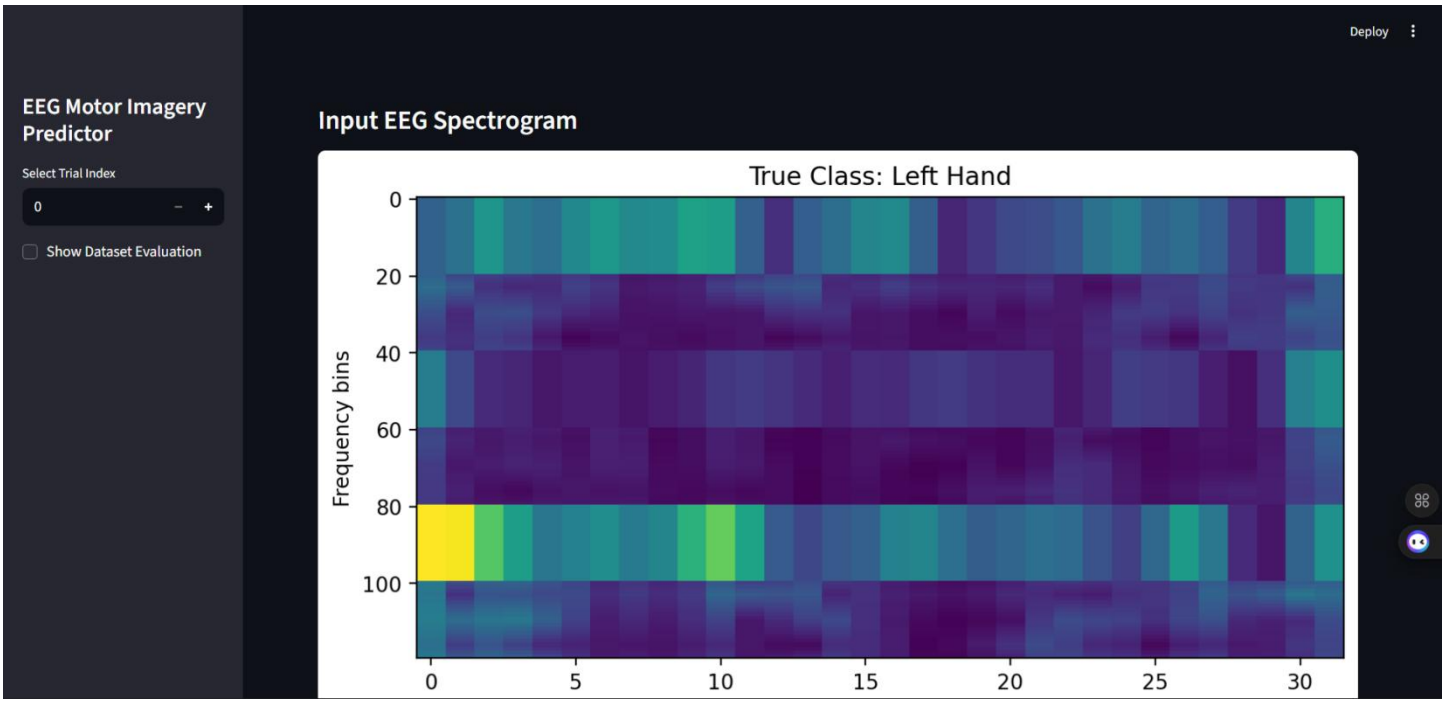

Fig 5.2 Input EEG Spectrogram

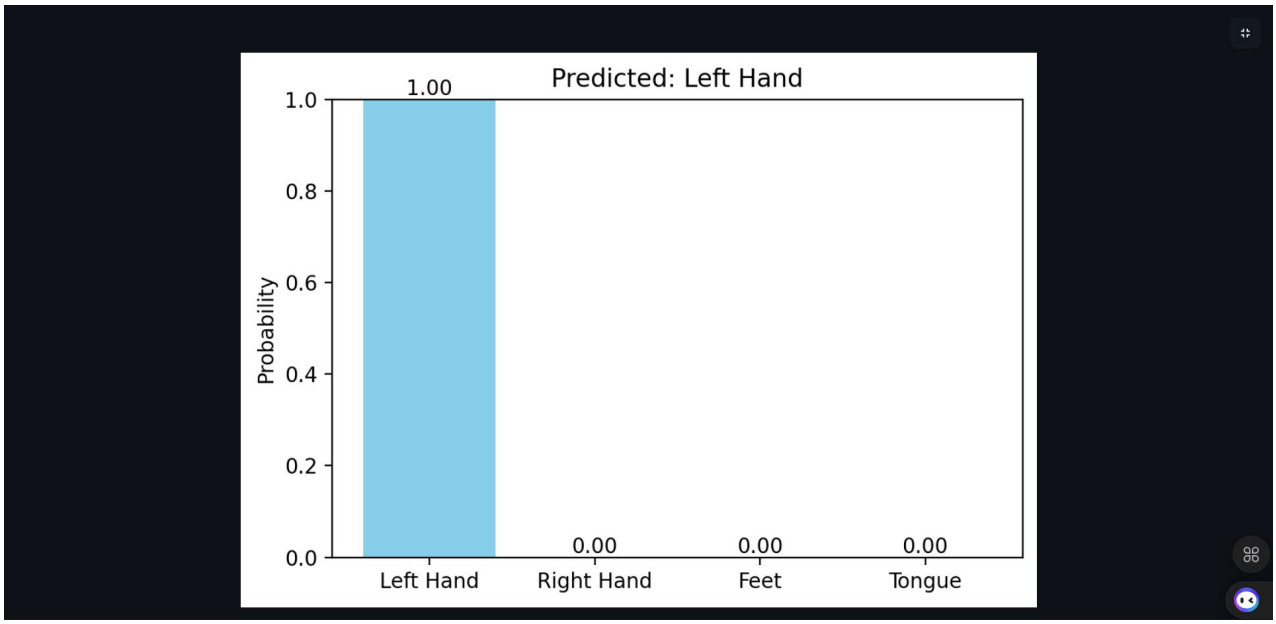

Fig 5.3 Predicted Output

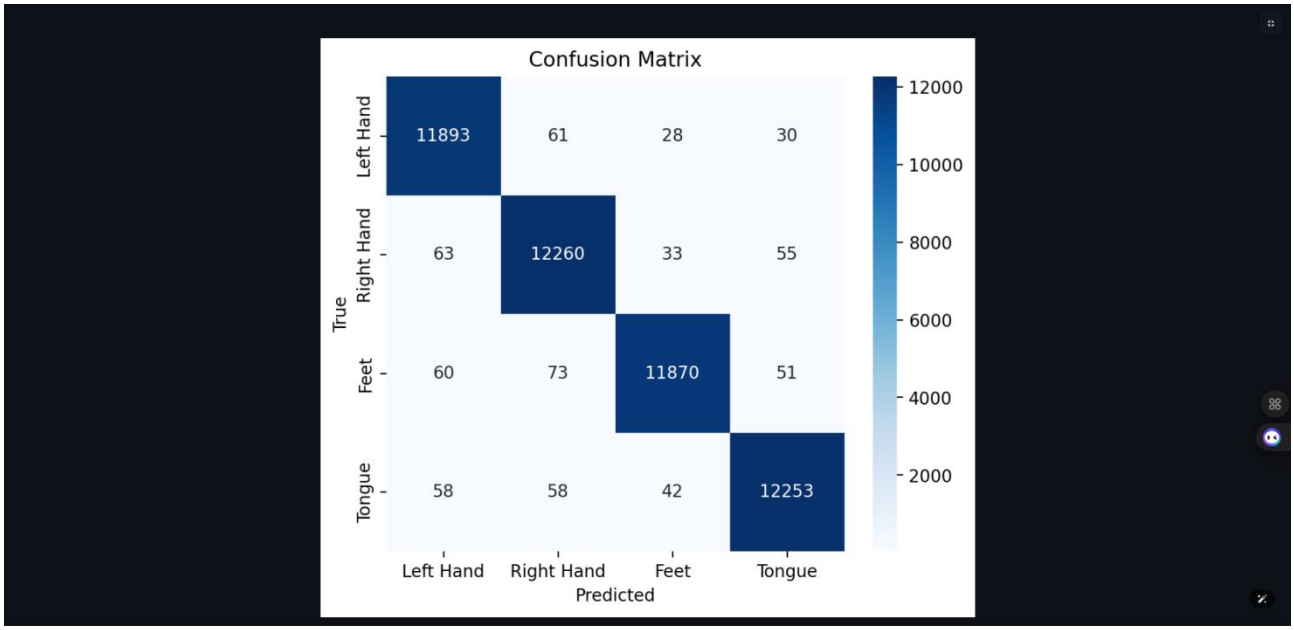

Fig 5.4 Confusion Matrix

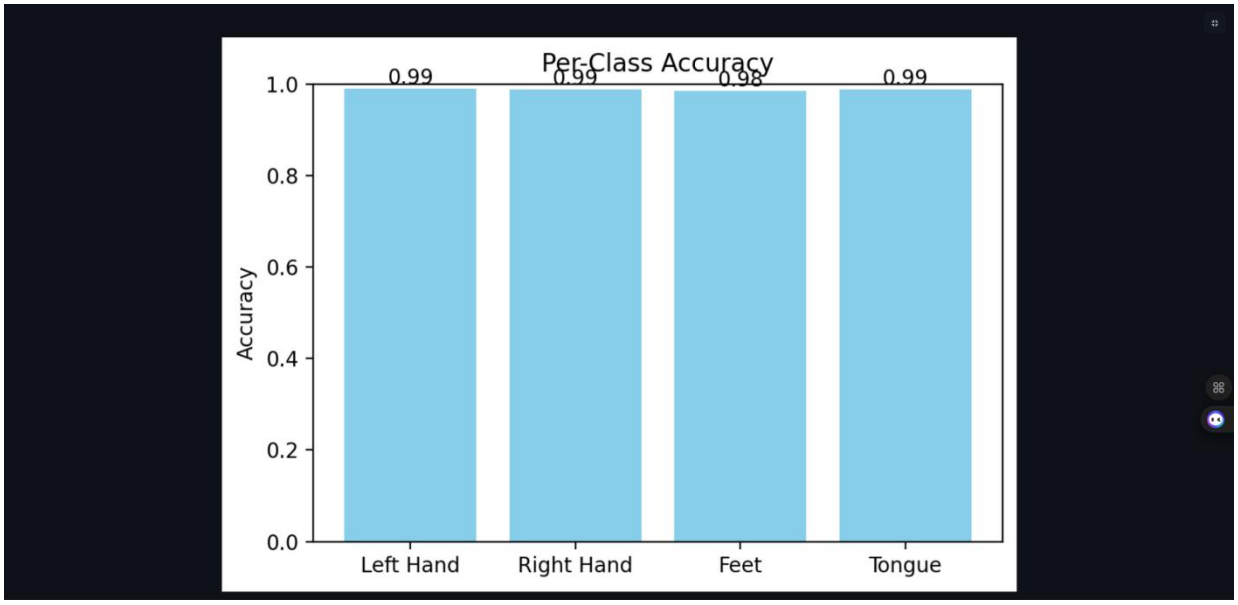

Fig 5.5 Per class Accuracy

CHAPTER 6

CONCLUSION

The development of the EEG Motor Imagery Predictor System marks a significant advancement in the field of Brain–Computer Interface (BCI) research, demonstrating how deep learning and interactive visualization can be combined to achieve real-time, interpretable EEG classification. The system's design integrates multiple stages — data preprocessing, spectral transformation, CNN-based classification, and visualization — to enable a seamless transition from raw EEG signals to meaningful motor intention predictions.

The implementation of the EnhancedEEGCNNV3 model represents an improvement over traditional architectures by employing deeper convolutional layers, optimized feature extraction, and regularization techniques such as batch normalization and dropout. This allows the model to learn intricate spatial-temporal dependencies within EEG spectrograms, which are often overlooked by conventional machine learning approaches. The incorporation of Short-Time Fourier Transform (STFT) ensures that the frequency and time variations in EEG signals are effectively captured, enabling the system to distinguish between subtle motor imagery patterns like left-hand versus right-hand movements.

Another critical contribution of this project lies in its Streamlit-based real-time interface, which bridges the gap between deep learning models and end users. The interface allows users to select trial indices, visualize EEG spectrograms, inspect class-wise prediction probabilities, and analyze overall performance through confusion matrices and per-class accuracy plots. Such transparency not only improves user trust but also enhances the interpretability of neural network decisions — a crucial factor in medical and neuro-rehabilitation applications.

The evaluation results demonstrate that the model achieves high classification accuracy across the four target motor imagery classes (left hand, right hand, feet, and tongue), validating the robustness of the proposed methodology. Furthermore, the confusion matrix analysis confirms that the CNN model successfully generalizes across different trial instances while maintaining consistent prediction quality. This performance makes the proposed system highly suitable for real-world BCI applications, such as robotic limb control, motor restoration in paralyzed patients, and cognitive training systems.

From a system design perspective, the modular architecture allows for easy scalability and customization. Each stage — from signal acquisition to model deployment — is independent yet interoperable, which enables future extensions without altering the overall pipeline. The use of PyTorch for model training and Streamlit for deployment ensures that the framework remains lightweight, accessible, and deployable on both local and cloud environments. This flexibility makes it a practical tool not just for researchers but also for educational and clinical practitioners who require an intuitive EEG analysis platform.

Moreover, the system promotes transparency and reproducibility, as all major components — preprocessing, training, and visualization — can be replicated or improved upon. Researchers can modify hyperparameters, integrate new datasets, or test additional deep learning architectures with minimal changes. The inclusion of statistical evaluation tools such as accuracy scores and confusion matrices further reinforces the scientific rigor of the system.

In conclusion, this project provides a comprehensive solution for EEG-based motor imagery prediction, combining deep learning accuracy with real-time interpretability. The results affirm that CNN-based architectures, when coupled with effective preprocessing and intuitive visualization, can revolutionize the field of EEG signal classification. The proposed model not only enhances classification performance but also lays the groundwork for future intelligent BCI systems capable of assisting humans through direct neural interaction.

CHAPTER 7

REFERENCE

1. EEG Motor Imagery Dataset – BCI Competition IV Dataset 2a, Graz University of Technology, Austria. [Online] Available: https://www.bbci.de/competition/iv/
2. Pfurtscheller, G., & Neuper, C. (2001). Motor Imagery and Direct Brain-Computer Communication. Proceedings of the IEEE, 89(7), 1123–1134.
3. Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., et al. (2017). Deep Learning with Convolutional Neural Networks for EEG Decoding and Visualization. Human Brain Mapping, 38(11), 5391–5420.
4. Lawhern, V. J., Solon, A. J., Waytowich, N. R., et al. (2018). EEGNet: A Compact Convolutional Neural Network for EEG-Based Brain–Computer Interfaces. Journal of Neural Engineering, 15(5), 056013.
5. Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., & Arnaldi, B. (2007). A Review of Classification Algorithms for EEG-Based Brain–Computer Interfaces. Journal of Neural Engineering, 4(2), R1–R13.
6. Roy, Y., Banville, H., Gramfort, A., Falk, T. H., & Faubert, J. (2019). Deep Learning-Based Electroencephalography Analysis: A Systematic Review. Journal of Neural Engineering, 16(5), 051001.
7. Python Software Foundation. Python 3.11 Documentation. [Online] Available: https://docs.python.org/3/
8. PyTorch Documentation – Deep Learning Framework. [Online] Available: https://pytorch.org/
9. Streamlit Documentation – Building Interactive ML Web Apps. [Online] Available: https://docs.streamlit.io/
10. Scikit-learn Documentation – Machine Learning in Python. [Online] Available: https://scikit-learn.org/
11. NumPy Documentation – Fundamental Package for Scientific Computing. [Online] Available: https://numpy.org/
12. Matplotlib Documentation – Python Plotting Library. [Online] Available: https://matplotlib.org/
13. Seaborn Documentation – Statistical Data Visualization. [Online] Available: https://seaborn.pydata.org/
14. Brunner, C., Leeb, R., Müller-Putz, G., Schlögl, A., & Pfurtscheller, G. (2008). BCI Competition 2008 – Graz Data Set A. Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology.