Classification of Motor Imagery Electroencephalogram

Electrical Engineering Thesis 2023/2024

Akram Chakrouni and Ilyaas Shousha

Delft University of Technology

Classification of Motor Imagery Electroencephalogram

Electrical Engineering Thesis 2023/2024

by

Akram Chakrouni and Ilyaas Shousha

Student Name	Student Number
Akram Chakrouni	5473829
Ilyaas Shousha	5593247

Supervisor:	dr. B. Hunyadi
Project Duration:	April, 2024 - June, 2024
Faculty:	Faculty Electrical Engineering, Mathematics and Computer Science, Delft

Preface

Akram Chakrouni and Ilyaas Shousha
Delft, June 2024

This paper is part of a Bachelor's graduation project to develop a streaming Brain-Computer Interface (BCI) to play a game using EEG signals. The project is divided into three groups: classification, measurements, and interface. Our group, the classification group, is specifically responsible for decoding EEG signals and providing outputs to the interface group based on the user's intentions. Various methods for pre-processing and classification are explored in this paper.

We would like to express our deepest gratitude to our supervisor, Prof. Dr. B. Hunyadi, for her guidance and support.

We hope this project will be continued and further developed after completing this thesis, as it is a fascinating subject with significant potential for future exploration and development.

Abstract

The thesis focuses on developing a brain-computer interface (BCI) aimed at differentiating between left-hand and right-hand motor imagery using EEG signals. Its primary objective was to create a scalable and user-specific model for accurately interpreting motor imagery tasks. The study involved the development of one SVM and two advanced deep learning models: the Advanced-EEG-MI-CNN and the Advanced-EEG-MI-CNN-LSTM-Transformer. These models were tested on the Physionet and BCIC IV 2a datasets, as well as on self-gathered data. Results indicated that the SVM model achieved an accuracy of 65% for all subjects and 78% for one subject, while the Advanced-EEG-MI-CNN model achieved an accuracy of 65% for all subjects and 78% for a single subject on average. The Advanced-EEG-MI-CNN-LSTM-Transformer demonstrated promising results in capturing temporal dependencies, with an accuracy of 80% for all subjects and 92% for a single subject on average. Future work includes enhancing user-specific model implementation, applying Independent Component Analysis (ICA) for cleaner data, and improving data pre-processing to minimize noise.

Contents

1 Introduction

1.1. Background

Brain-Computer Interfaces (BCIs) represent a groundbreaking technology that enables direct communication between the brain and external devices without the need for muscular activity. This technology has significant implications for individuals with severe motor disabilities, providing them with new means of interaction and control. BCIs typically rely on the interpretation of Electroencephalogram (EEG) signals, which are electrical signals produced by the brain. These signals are captured using electrodes placed on the scalp and are analyzed to infer the user's intentions.

BCIs have come a long way since the 1970s and 1980s when early research focused on basic signal acquisition and classification of mental states. Recent advances in machine learning and signal processing have led to the development of more accurate BCI systems. For example, research by Wolpaw et al. [29] demonstrated the potential of BCIs in enabling control over external devices through EEG signal classification. Furthermore, studies by Pfurtscheller and Neuper [17] explored the use of motor imagery for BCI control, laying the foundation for modern motor imagery-based BCI systems.

In addition to medical applications, BCIs are also being explored for their potential in the gaming industry. Gaming with BCIs allows players to control game elements using their brain signals alone, offering an immersive experience, especially for users who may be unable to use traditional input devices due to physical limitations. BCIs in gaming can also enhance the overall gaming experience by providing more intuitive and natural control mechanisms.

However, EEG signals are often noisy and vary considerably between individuals, posing challenges for the development of reliable BCI systems. To address these challenges, researchers have employed various machine learning techniques to improve the classification accuracy of EEG signals, with Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) being among the most commonly used models in this context.

1.2. Motivation

The motivation behind this research is to improve the accuracy and scalability of BCI systems, particularly for motor imagery tasks. Accurate classification of motor imagery is critical for developing effective BCI systems, which can be used in various applications such as medical rehabilitation, assistive technologies, and gaming. The integration of advanced machine learning techniques promises to enhance the interpretability of EEG signals, which are often complex and noisy. By achieving higher classification accuracy, BCIs can become more reliable and user-friendly, thereby expanding their practical applications and improving the quality of life for users with motor impairments.

1.3. Contributions

This thesis contributes to the field by:

- Developing one SVM and two advanced deep learning models for motor imagery classification.
- Implementing a scalable approach that improves with the addition of new user data.
- Providing a detailed comparative analysis of model performances on different datasets.

1.4. Constrains

Constraints include limited computational resources and variability in EEG signal quality across different users and sessions, which impact the generalizability of the models. Additionally, there is a time constraint of 10 weeks.

1.5. Thesis Structure

The thesis is organized as follows:

- **Chapter 2:** Literature review of BCI systems, motor imagery, and machine learning applications.
- **Chapter 3:** Theory behind SVM, CNN, and advanced models.
- **Chapter 4:** Methodology detailing data acquisition, preprocessing, and model development.
- **Chapter 5:** Presentation of results from various models.
- **Chapter 6:** Discussion of findings and their implications.
- **Chapter 7:** Conclusion summarizing the research and suggesting future work.

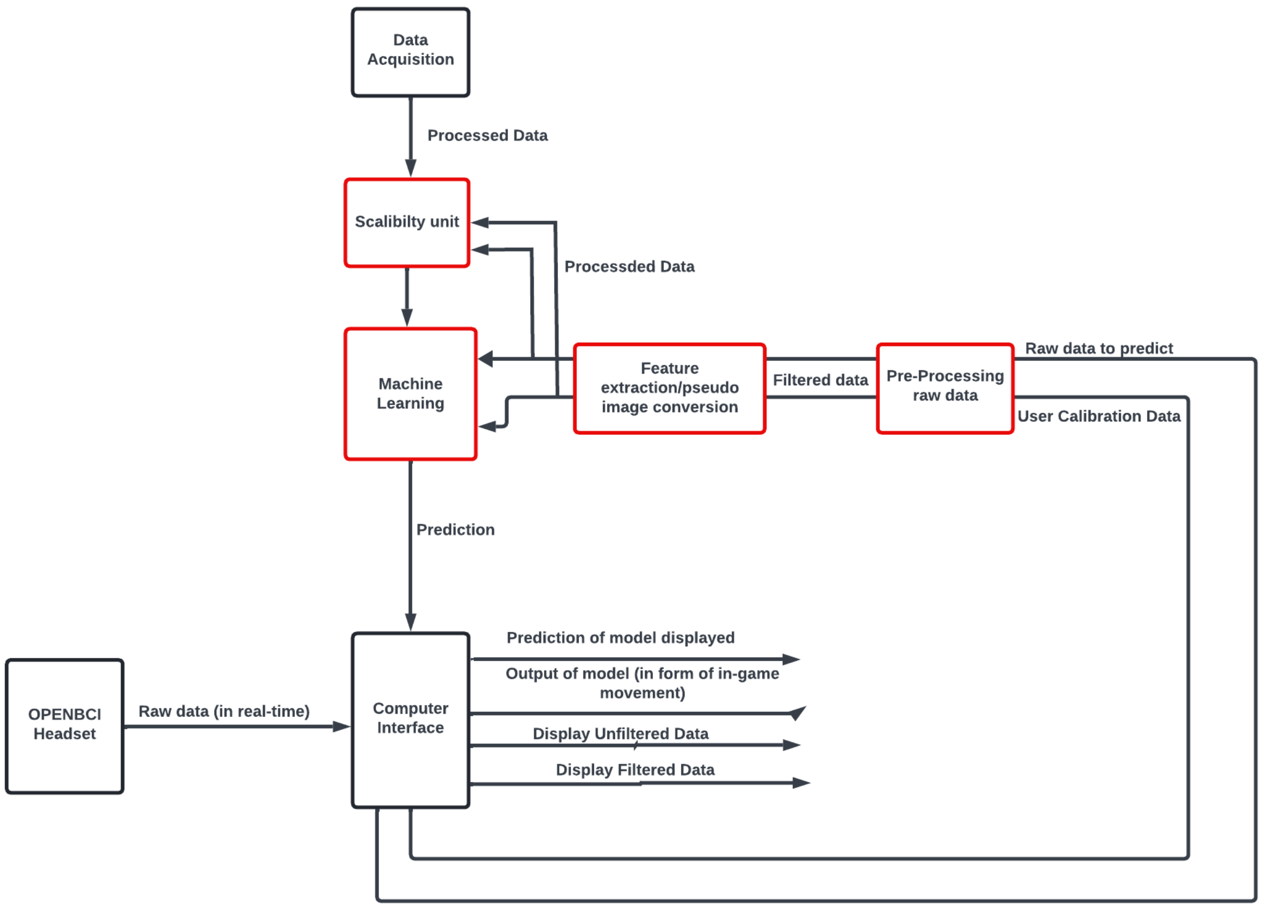

Figure 1.1: Overview of the implementation of the total BCI, with in red the aspects discussed in this report

2 Program of Requirements

This chapter will discuss the program of requirements that the design should contain.

General Specifications

Our general market focuses on Gaming, with a future focus on medical applications. This should be represented in the general specifications. Gamers generally would like reliability with less delay; errors in accuracy can be forgiven if priced adequately. The general specifications are listed below:

- The delay from headset to prediction should be less than 0.75 seconds.
- The classification accuracy must be at least 75%.
- The machine learning and streaming must be done in Python to ensure easy compatibility.

The Requirements for the machine learning group are as follows:

Must:

- Accuracy on online data should be at least 80 per cent.
- Accuracy on own data (OpenBCI) should be at least 75%.
- The time taken to process incoming data and classify it must be at most 500 ms. This ensures that users enjoy an immersive gaming experience without experiencing noticeable delays.
- The product should be ready to integrate with the other subgroups. Therefore, it should also work for a continuous input signal and provide a continuous output signal.

Should:

- The model can create user-specific models coupled to the user's account to improve the classification accuracy for each user independently.
- It should be scalable to accommodate potential data volume and user base increases over time.

3
Literature Review

3.1. Model

Today, many machine learning models are accessible for motor imagery tasks. Hosseini et al. [10] offers a comprehensive overview of machine learning applications utilised in EEG analysis. This includes well-established models such as Support Vector Machines (SVM), Linear Discriminant Analysis (LDA), and k-nearest Neighbors (kNN). The overview encompasses a detailed examination of each method, highlighting their strengths and the specific applications to which they best suit. This resource serves as a valuable guide for selecting the most appropriate machine learning approach tailored to EEG-based motor imagery analysis requirements.

In addition to traditional machine learning models, deep learning has emerged as a prominent force in Motor Imagery (MI) Brain-Computer Interface (BCI) research and applications. Deep learning techniques bring several advantages to MI-BCI tasks, such as Automatic Feature Learning, End-to-End Learning, and Adaptability. These capabilities enable deep learning models to extract relevant features from EEG signals automatically, learn complex patterns directly from raw data, and adapt to varying conditions without extensive manual intervention. As a result, deep learning holds significant promise for advancing the field of MI-BCI by facilitating more robust and efficient signal processing and classification techniques.

Unlike traditional machine learning models, deep learning models like Convolutional Neural Networks (CNNs) can interpret EEG signals as pseudo-images. For instance, in Schirrmeister et al. [21], the authors propose a deep learning approach using CNNs to decode EEG signals recorded during MI tasks. They represent EEG data as spectrogram-like images and feed them into a CNN architecture for classification. The study demonstrates the effectiveness of CNNs in automatically learning spatial and temporal features from EEG signals for MI classification tasks, achieving competitive performance compared to traditional machine learning methods.

3.2. Transformation

Transformation of EEG signals holds paramount importance in motor imagery classification. The specific transformation applied to EEG signals before feature extraction significantly impacts the accuracy of the model.

In past research, various transformation methods have been explored to enhance classification accuracy. One of the most common feature extraction methods is Common Spatial Patterns (CSP) [16]. CSP operates by maximising the variance of EEG signals for one class (e.g., imagining moving the left hand) while simultaneously minimising the variance for another class (e.g., imagining moving the right hand).

Additionally, researchers have investigated alternative transformation techniques. For instance, Samiee et al. [19] employed the short-time Fourier transform (STFT) to process EEG signals, known for its effectiveness in providing time-frequency analysis. Furthermore, Alyasseri et al. [3] proposed an EEG denoising method based on wavelet transform (WT). The WT method offers several advantages, including multiresolution analysis, time-frequency localisation, artifact removal, dimensionality reduction, adaptability, and robustness to noise. These methods contribute to the refinement of EEG signals for subsequent classification tasks in MI-BCI systems.

3.3. Feature Engineering

The influence of various features or input representations on the performance of machine learning models is widely acknowledged as a fundamental aspect of model optimisation. It is empirically observed that the selection and engineering of features significantly impact the model's ability to learn and generalise from the data accurately.

Feature selection methods are typically classified into three categories: wrapper, filter, and embedded. Wrapper methods evaluate feature subsets based on the accuracy of a preselected classifier. In contrast, filter methods assess subsets independently of a classifier, often using an upper bound or approximating the optimal Bayes error. Embedded feature selection mechanisms are integrated within specific classifiers. Notably, embedded feature selection tends to be faster than wrapper methods and yields higher classification accuracy than filter methods [11].

In recent years, research has focused on extracting novel features from existing datasets through algorithms for nonlinear dimensionality reduction. These algorithms facilitate the transition from high-dimensional spaces to two-dimensional representations. The resulting features are then integrated with the original dataset's features to enhance signal classification quality using trained classifiers or to construct new datasets for visualisation purposes [8].

3.4. Model Deployment and Scalability

Considerations for deploying machine learning models in real-world applications, including scalability, latency, and resource constraints, are vital for ensuring optimal performance and reliability. Exploring frameworks for deployment and implementing techniques for model monitoring and maintenance are essential steps to maintain the effectiveness and sustainability of deployed models over time.

4 Theory

4.1. Theory of SVM Model

4.1.1. Introduction to SVM

Support Vector Machines (SVM) are supervised learning models widely used for classification and regression tasks. Introduced by Vladimir Vapnik and his colleagues, SVMs are built on the fundamental principle of finding an optimal hyperplane that best separates the data points of different classes. This separation is achieved by maximising the margin, which is the distance between the hyperplane and the nearest data points from either class. The data points that lie closest to the hyperplane are known as support vectors, and they play a critical role in defining the position and orientation of the hyperplane.

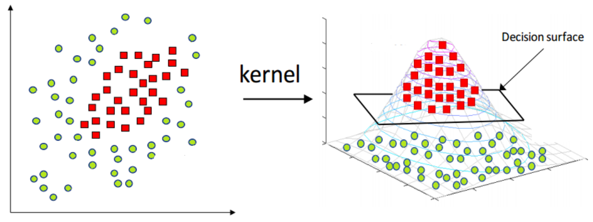

Figure 4.1: Classification non-linear with Kernel Source: [12].

The power of SVMs is significantly enhanced by using kernel functions. Kernels enable SVMs to handle non-linearly separable data by implicitly mapping the input features into a higher-dimensional space where a linear separation is possible, shown in figure 4.1. This mapping is done without explicitly computing the coordinates in the higher-dimensional space, making the computation efficient even for very high-dimensional spaces. Commonly used kernel functions include:

- **Linear kernel**: This is the simplest kernel function, defined as $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$. It is suitable for linearly separable data.
- **Polynomial kernel**: Defined as $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d$, where c is a constant and d is the degree of the polynomial. This kernel can model more complex relationships by increasing the degree d.
- **Radial Basis Function (RBF) kernel**: Also known as the Gaussian kernel, it is defined as $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma||\mathbf{x} - \mathbf{y}||^2)$, where γ is a parameter that determines the spread of the kernel. The RBF kernel is widely used because of its ability to handle non-linear relationships.
- **Sigmoid kernel**: Defined as $K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x} \cdot \mathbf{y} + c)$, where κ and c are kernel parameters. This kernel behaves similarly to neural networks with a sigmoid activation function.

The figure(4.2) below illustrates how each kernel classifies the data:

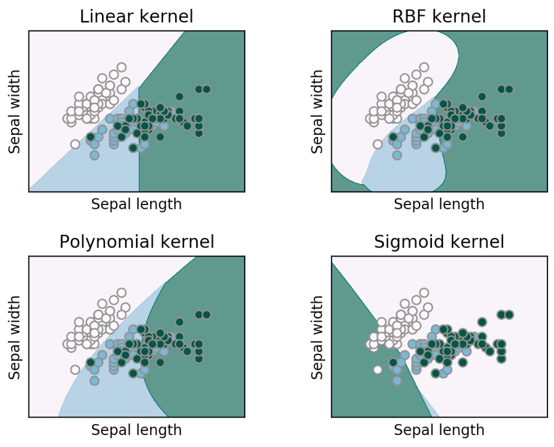

Figure 4.2: Different Kernels Source: [18].

4.1.2. Mathematical Formulation

The objective of an SVM is to find a hyperplane that maximises the margin between two classes. For a linearly separable dataset, this hyperplane can be mathematically defined by the equation:

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{4.1}$$

where $\mathbf{w}$ is the weight vector and b is the bias term.

The margin is the distance between the hyperplane and the closest data points from each class, known as support vectors. The primary goal of an SVM is to find a hyperplane that not only separates the classes but also maximises the distance to the nearest points (support vectors) from both classes. This maximisation ensures that the classifier is as robust as possible to any misclassifications on the training data.

For a linearly separable dataset, the distance d from a point $\mathbf{x}_i$ to the hyperplane is given by:

$$d = \frac{|\mathbf{w} \cdot \mathbf{x}_i - b|}{||\mathbf{w}||} \tag{4.2}$$

For the support vectors, this distance is exactly 1 unit away from the hyperplane, which gives:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) = 1 \tag{4.3}$$

where y_i is the class label ($+1$ or -1).

The margin is, therefore, 2 times the distance from the hyperplane to the nearest point on either side, which simplifies to:

$$\text{margin} = \frac{2}{||\mathbf{w}||} \tag{4.4}$$

To maximise the margin, we need to maximise $\frac{2}{||\mathbf{w}||}$. This is equivalent to minimising $\frac{1}{2}||\mathbf{w}||^2$ since minimising the square of the norm simplifies the calculations and is a standard form for optimisation problems.

This optimisation problem can be formalised as follows:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{4.5}$$

where $y_i \in \{-1, 1\}$ are the class labels of the data points.

To solve this constrained optimisation problem, we use the method of Lagrange multipliers. The Lagrangian for this problem is given by:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i - b) - 1] \tag{4.6}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers.

The dual form of the optimisation problem is derived by taking the partial derivatives of the Lagrangian with respect to $\mathbf{w}$ and b and setting them to zero:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{4.7}$$

Once the optimal values of α_i are found, the weight vector $\mathbf{w}$ and bias b can be calculated as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{4.8}$$

$$b = y_k - \mathbf{w} \cdot \mathbf{x}_k \quad \text{for any support vector} \quad \mathbf{x}_k \tag{4.9}$$

The decision function for classifying new data points $\mathbf{x}$ is then given by:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) - b \right) \tag{4.10}$$

4.1.3. Applications in BCI

Support Vector Machines (SVMs) are widely used for Brain-Computer Interface (BCI) classification due to several advantageous properties [23] :

- **Effective in High-Dimensional Spaces**: SVMs handle high-dimensional feature spaces well, which is common in BCI data.
- **Robust to Overfitting**: By maximising the margin between classes, SVMs reduce the risk of overfitting, which is crucial for small BCI datasets.
- **Kernel Trick Flexibility**: SVMs can use various kernel functions to handle non-linear relationships in BCI data effectively.
- **Good Generalization Performance**: The margin maximisation principle ensures good performance on unseen data, essential for real-time BCI applications.
- **Noise Handling**: SVMs, with appropriate regularisation, can handle noisy BCI data effectively.
- **Binary Classification Suitability**: Many BCI tasks are binary, well-suited for SVMs, and multi-class problems can be handled with techniques like one-vs-one or one-vs-all.

4.2. Theory of Neural Networks, CNNs, and LSTMs

4.2.1. Introduction to Neural Networks

Artificial Neural Networks (ANNs) are computational models inspired by the human brain's structure and function. They consist of interconnected layers of nodes (neurons) that process input data to learn patterns and make predictions. Each connection between neurons has an associated weight, which is adjusted during training to minimise the error in the network's output. This process is known as learning or training the network.

The basic structure of an ANN includes an input layer, one or more hidden layers, and an output layer. Each neuron in a layer applies an activation function to the weighted sum of its inputs to introduce non-linearity, allowing the network to model complex relationships. Common activation functions include:

- **Sigmoid**: Defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, it squashes the input to a range between 0 and 1.
- **ReLU (Rectified Linear Unit)**: Defined as $f(x) = \max(0, x)$, it introduces sparsity and mitigates the vanishing gradient problem.
- **Tanh**: Defined as $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, it maps inputs to a range between -1 and 1.

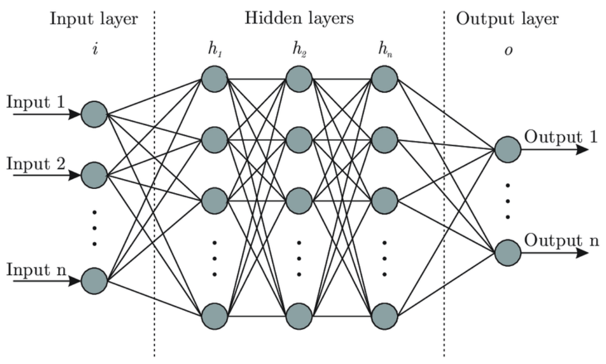

Figure 4.3: General visualization of a Neural Network, [25].

4.2.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are specialised neural networks that are well-suited for processing grid-like data, such as images. Introduced by Yann LeCun and colleagues [30], CNNs leverage the spatial structure of the data through convolutional layers, pooling layers, and fully connected layers.

Convolutional Layers

Convolutional layers apply a set of learnable filters (kernels) to the input data, producing feature maps that capture spatial hierarchies and local patterns. The convolution operation is defined as:

$$Y[i,j] = \sum_m \sum_n X[i+m, j+n] \cdot K[m,n] \tag{4.11}$$

X is the input, K is the kernel, and Y is the output feature map.

The filter is a small matrix that extracts features from an image through a convolution operation. This involves performing a dot product, where the filter matrix is multiplied element-wise with the corresponding input values and then summed to produce a single value. The filter is then moved across the input image, sliding over it until the entire image has been processed through convolution.

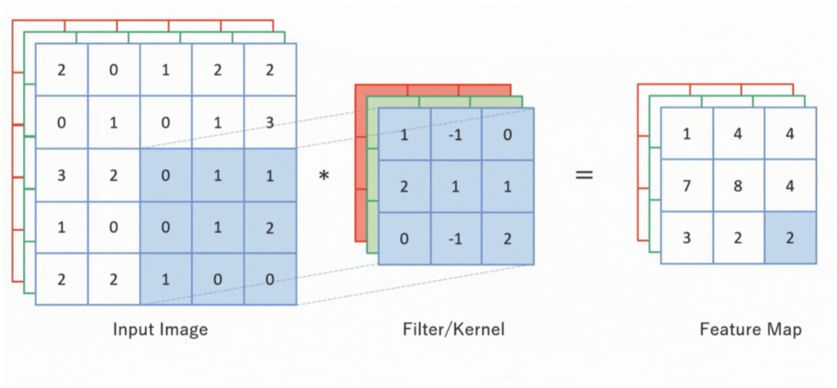

Figure 4.4: A convolutional layer filter sliding across the input image and creating a feature map through corresponding dot product operations, [5].

Pooling Layers

Pooling layers reduce the feature maps' spatial dimensions, which helps decrease the computational load and mitigate overfitting. Two common types of pooling are:

- **Max Pooling**: Takes the maximum value in each patch of the feature map.
- **Average Pooling**: Takes the average value in each patch of the feature map.

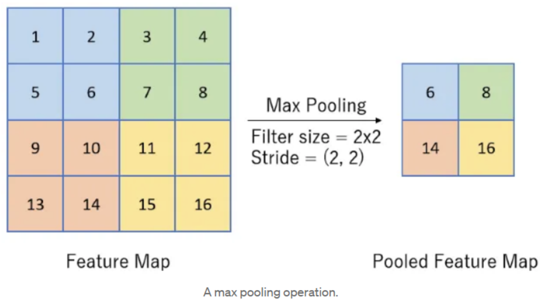

Figure 4.5: Max Pooling of the Feature Map, [5].

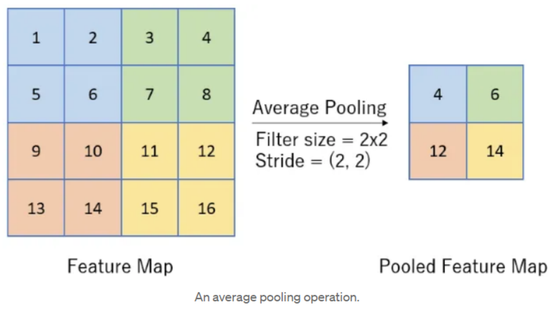

Figure 4.6: Average Pooling of the Feature Map, [5].

Fully Connected Layers

After several convolutional and pooling layers, the output is flattened and fed into fully connected layers, which perform the final classification or regression task. These layers are similar to traditional neural network layers and contribute to the model's decision-making process.

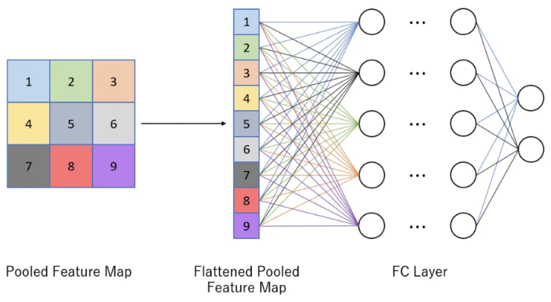

Figure 4.7: A pooled feature map being flattened and inputted into a fully connected (FC) network layer, [5].

4.2.3. Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) designed to model temporal sequences and long-range dependencies. Introduced by Hochreiter and Schmidhuber, [24], LSTMs address the vanishing gradient problem common in traditional RNNs through a sophisticated gating mechanism.

LSTM Architecture

An LSTM unit consists of three main gates:

- **Forget Gate**: Decides what information to discard from the cell state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

  where σ is the sigmoid function, h_{t-1} is the previous hidden state, x_t is the current input, W_f are the weights, and b_f is the bias.

- **Input Gate**: Determines which new information to store in the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  where i_t is the input gate activation, $\tilde{C}_t$ is the candidate cell state, W_i and W_C are the weights, and b_i and b_C are the biases.

- **Cell State Update**: Updates the cell state using the forget and input gates.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

  where C_{t-1} is the previous cell state and C_t is the updated cell state.

- **Output Gate**: Determines the output of the LSTM cell.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

  where o_t is the output gate activation, h_t is the current hidden state, W_o are the weights, and b_o is the bias.

These gates collectively enable the LSTM to maintain and update information over long sequences, making it well-suited for tasks such as time-series prediction, natural language processing, and EEG signal analysis in BCI applications.

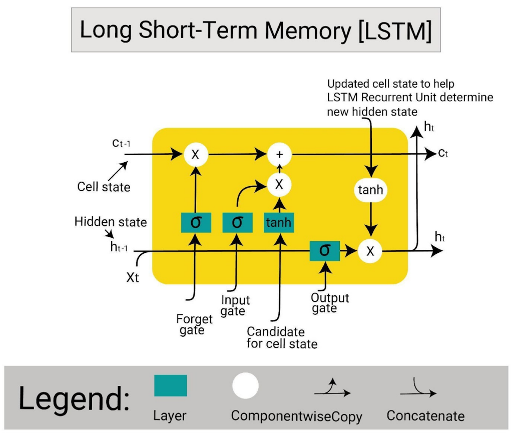

Figure 4.8: Structure of an LSTM cell, highlighting the flow of information and the roles of various gates, [26]

4.2.4. Combined CNN-LSTM and Transformer Architectures

Recent advancements in deep learning have seen the combination of CNNs and LSTMs to leverage both spatial and temporal data features. This hybrid approach benefits applications where data indicates spatial hierarchies and temporal dependencies, such as EEG signal processing.

CNN-LSTM Hybrid Models

In a CNN-LSTM hybrid model, the convolutional layers are responsible for extracting spatial features from the input data, while the LSTM layers capture temporal dependencies. This architecture is particularly effective for tasks where spatial features evolve over time.

Incorporating Transformers

Originally introduced for natural language processing tasks, transformers have been adapted to enhance sequence modelling in various domains. In EEG signal processing, transformers can be integrated into CNN-LSTM architectures to capture complex temporal dependencies and attention mechanisms further. This results in a robust model capable of learning complex patterns in the data.

4.2.5. Applications in BCI

Neural Networks, CNNs, and LSTMs are extensively utilised in Brain-Computer Interface (BCI) systems because they handle complex, high-dimensional data and capture intricate patterns. Their applications in BCI include:

- **Feature Extraction and Classification**: CNNs are highly effective in automatically extracting relevant features from raw EEG signals and classifying them into different mental states or commands.
- **Temporal Dynamics Modeling**: LSTMs excel in modelling the temporal dynamics of EEG signals, making them suitable for tasks requiring the analysis of temporal dependencies, such as predicting user intent over time.
- **Hybrid Architectures**: Combining CNNs with LSTMs allows for capturing both spatial and temporal features in EEG data, enhancing the accuracy and robustness of BCI systems.
- **Attention Mechanisms**: Integrating transformers into CNN-LSTM architectures enables attention mechanisms to focus on the most relevant parts of the EEG signal, improving classification performance.
- **Real-Time Applications**: These models can process EEG signals from streaming data, making them suitable for applications like neurofeedback, rehabilitation, and external control through mental commands.
- **Noise Robustness**: Advanced neural network architectures can be trained to be robust against the inherent noise in EEG signals, ensuring reliable performance in practical BCI scenarios.

4.3. Theory of Transformations for Feature Extraction

In Brain-Computer Interface (BCI) systems, effective feature extraction from EEG signals is crucial for accurate classification. Various transformations are used to extract relevant features from the raw EEG data. This section discusses three commonly used transformations: Short-Time Fourier Transform (STFT), Wavelet Transform, and Common Spatial Patterns (CSP).

4.3.1. Short-Time Fourier Transform (STFT)

Over time, the Short-Time Fourier Transform (STFT) analyses the frequency content of non-stationary signals, such as EEG signals. It divides the signal into short overlapping segments and applies the Fourier Transform to each segment.

Basics:

- The STFT represents a signal in both time and frequency domains.
- It involves windowing the signal into short segments and computing the Fourier Transform of each segment.

Mathematical Formulation:

- The STFT of a signal $x(t)$ is defined as:

$$X(t,f) = \int_{-\infty}^{\infty} x(\tau)w(\tau - t)e^{-j2\pi f\tau}d\tau \tag{4.12}$$

where $w(\tau - t)$ is a window function that slides over the signal $x(t)$. To minimise spectral leakage, the window function $w(\tau - t)$ is typically chosen as a Gaussian or Hamming window.

Application:

- STFT is used in EEG analysis to observe how the spectral content of EEG signals evolves over time, which is crucial for identifying time-varying patterns in brain activity.

Advantages in EEG Analysis:

- Provides a time-frequency representation, useful for identifying transient events and changes in brain activity.
- Allows for detecting frequency-specific patterns associated with different mental states or tasks.
- The resolution trade-off between time and frequency can be managed by selecting appropriate window lengths.

4.3.2. Wavelet Transform

The Wavelet Transform is a time-frequency analysis method that decomposes a signal into components at various scales and positions. It is well-suited for analysing non-stationary signals like EEG.

Overview:

- Unlike STFT, the Wavelet Transform provides multi-resolution analysis, offering good time resolution for high-frequency components and good frequency resolution for low-frequency components.

Mathematical Formulation:

- The Continuous Wavelet Transform (CWT) of a signal $x(t)$ is defined as:

$$W_x(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi^* \left(\frac{t-b}{a}\right) dt \tag{4.13}$$

where ψ is the mother wavelet, a is the scale parameter, and b is the translation parameter. The mother wavelet ψ can be chosen from various types, such as Morlet, Mexican Hat, or Daubechies wavelets, depending on the specific requirements of the analysis.

Application in Time-Frequency Analysis of EEG Signals:

- Used to extract features that capture both the temporal and spectral characteristics of EEG signals, which is essential for understanding the dynamics of brain activity.

Advantages:

- Effective in detecting transient phenomena and abrupt changes in EEG signals.
- Provides a more flexible and detailed time-frequency representation compared to STFT.
- The ability to zoom in on short-time high-frequency events and long-time low-frequency trends makes it particularly suitable for analysing EEG signals with varying frequency characteristics.

4.3.3. Common Spatial Patterns (CSP)

Common Spatial Patterns (CSP) is a method used to enhance the discriminative power of features extracted from multi-channel EEG data, especially in motor imagery classification tasks.

Basics:

- CSP finds spatial filters that maximise one class's variance while minimising the other class's variance.
- This technique transforms the EEG signals into a feature space where the differences between classes are more pronounced.

Mathematical Formulation:

- Given EEG data from two classes, CSP aims to find a projection matrix W that maximises the variance for one class while minimising it for the other. This is done by solving the generalised eigenvalue problem:

$$\Sigma_1 W = \lambda \Sigma_2 W \tag{4.14}$$

  Where Σ_1 and Σ_2 are the covariance matrices of the two classes, the solution involves finding the eigenvectors corresponding to the largest eigenvalues for one class and the smallest eigenvalues for the other class.

Usage in Feature Extraction for Motor Imagery Classification:

- CSP is handy for classifying motor imagery tasks, where the goal is to distinguish between different imagined movements by enhancing the spatial patterns of EEG signals.
- The spatial filters obtained from CSP transform the EEG data into a space where the differences between motor imagery classes are maximised.

Advantages:

- Enhances the separability of different mental states or tasks in the feature space, improving classification performance.
- Reduces dimensionality and computational complexity while maintaining or improving classification accuracy.
- Provides interpretable spatial patterns that can offer insights into the neural correlates of different mental states.

5 Methodology

5.1. Data Acquisition

The Data Acquisition group was tasked with optimizing the processing of EEG signals recorded during a motor imagery exposure setup, which are essential for developing models that meet specific requirements.

The EEG data recorded by the Data Acquisition group consists of 8 channels. This number is significantly fewer compared to the average number of channels typically found in online datasets (ranging from 25 to 64 channels). The limitation in the number of channels is primarily due to resource constraints; specifically, the group only had access to the OPENBCI headset, which supports a maximum of 8 channels, [27].

The channels selected by the Data Acquisition group are as follows:

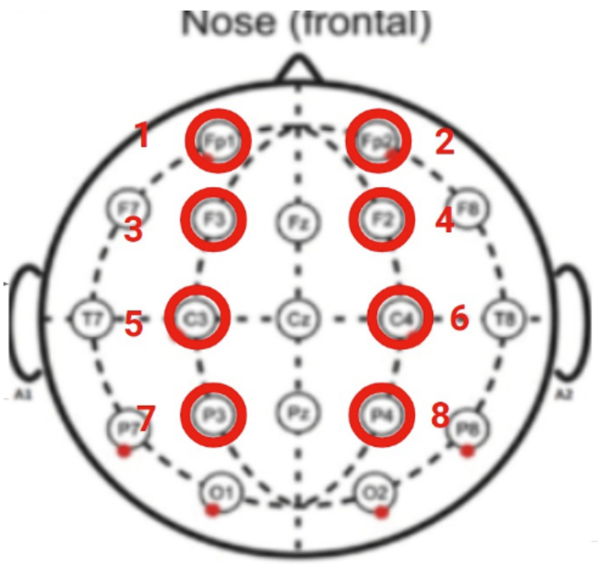

Figure 5.1: Electrode placement of Data Acquisition Group

5.2. Online Dataset

To ensure synchronous progress with the development of classification models, online datasets are utilized initially, while the Data Acquisition group gathers EEG data. The online datasets used are as follows:

- **BCI Competition IV 2a** [7]
- **PhysioNet** [20], [**goldberger2000physiobank**]

Below is a brief description of each dataset:

5.2.1. BCI Competition IV 2a

Setup

- **Subjects**: 9
- **Sessions**: 2 per subject, recorded on different days
- **Runs per Session**: 6 runs with short breaks in between
- **Trials per Run**: 48 trials (12 for each motor imagery class)
- **Total Trials per Session**: 288

Tasks

Motor Imagery Classes:

1. Left hand movement
2. Right hand movement
3. Both feet movement
4. Tongue movement

Data Recording

- **EEG Channels**: 22 (recorded using Ag/AgCl electrodes)
- **EOG Channels**: 3 (monopolar recording)
- **Sampling Rate**: 250 Hz
- **Bandpass Filtering**: 0.5 Hz to 100 Hz
- **Notch Filter**: 50 Hz to suppress line noise

The figure below shows the placement of the electrodes during the recording:

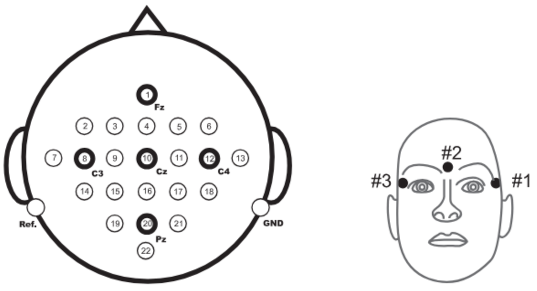

Figure 5.2: Electrode placement BCI Competition IV 2a, [7].

5.2.2. PhysioNet Dataset

This dataset consists of over 1500 one- and two-minute EEG recordings, obtained from 109 volunteers, as described below.

Experimental Protocol

Subjects performed different motor imagery tasks while 64-channel EEG recordings were taken using the BCI2000 system, [1]. Each subject performed 14 experimental runs: two one-minute baseline runs (one with eyes open, one with eyes closed), and three two-minute runs for each of the following four tasks:

1. A target appears on either the left or right side of the screen. The subject opens and closes the corresponding fist until the target disappears, then relaxes.
2. A target appears on either the left or right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears, then relaxes.
3. A target appears on either the top or bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears, then relaxes.
4. A target appears on either the top or bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears, then relaxes.

In summary, the experimental runs were:

1. Baseline, eyes open
2. Baseline, eyes closed
3. Task 1 (open and close left or right fist)
4. Task 2 (imagine opening and closing left or right fist)
5. Task 3 (open and close both fists or both feet)
6. Task 4 (imagine opening and closing both fists or both feet)
7. Repeat Tasks 1-4 in the same order

The data are provided in EDF+ format (containing 64 EEG signals, each sampled at 160 samples per second, and an annotation channel). The .event files and the annotation channels in the corresponding .edf files contain identical data.

Each annotation includes one of three codes:

- **T0**: Rest
- **T1**: Onset of motion (real or imagined) of the left fist (in runs 3, 4, 7, 8, 11, and 12) or both fists (in runs 5, 6, 9, 10, 13, and 14)
- **T2**: Onset of motion (real or imagined) of the right fist (in runs 3, 4, 7, 8, 11, and 12) or both feet (in runs 5, 6, 9, 10, 13, and 14)

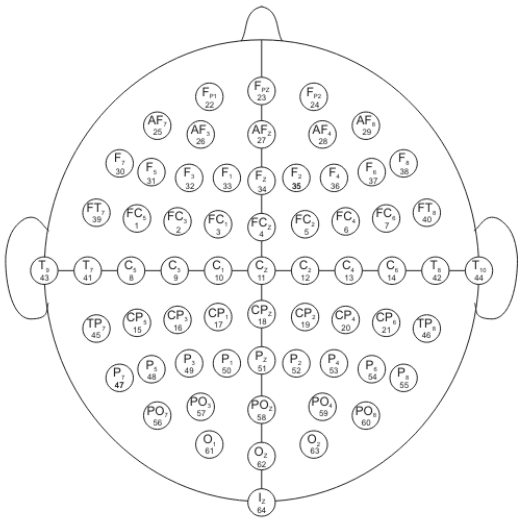

Figure 5.3: Electrode placement PhysioNet Dataset

Again, the dataset has been filtered to be similar to ours. Therefore, the right channels and tasks have been selected.

5.3. Preprocessing

To get the signals of the online datasets in the same format as the data of the Data Acquisition group, the following steps have been applied:

- **Channel Selection:** The online datasets consist of more channels than those used by the Data Acquisition group and have different channel names. To make the processing scripts universal for all datasets, the 8 channels mentioned in Section 5.1 are selected from the online datasets and renamed to match the channel names used by the Data Acquisition group.
- **Resampling:** The sampling rate of the online datasets needs to match the sampling rate of the Data Acquisition group. This ensures that the filtering process yields consistent results across all datasets.

The preprocessing applied to both online datasets follows the similar processing scheme used by the Data Acquisition group, for their recordings. This scheme consists of several steps:

- **Bandpass Filtering:** The EEG signals are filtered between 8 and 30 Hz using a 5th order Butterworth bandpass filter. This step focuses on extracting the frequency band relevant for motor imagery (MI) brain activity. By filtering out frequencies outside this range, subsequent processing steps can be standardized across different datasets.
- **Blink Removal:** Blink artifacts in the EEG signal create spikes that do not contain meaningful information for MI. These artifacts are removed to ensure that the model does not learn patterns from irrelevant spikes.
- **Normalization:** Normalization using z-score is applied to each channel. This normalization ensures that each channel has a mean of zero and a variance of 1 across all samples. Standardizing the data in this way helps in maintaining consistent scaling across different recordings.
- **Epoching:** The EEG signal is segmented into epochs using the Python library MNE [15]. Epochs are created precisely at the trigger execution of movements, aligning each epoch with a specific event related to motor imagery. This segmentation organizes each recording into a sample matrix of shape (number of samples, time points), which is suitable for subsequent machine learning models.

These preprocessing steps are crucial for preparing the EEG data for input into SVM and CNN models, ensuring that the data is standardized and relevant features are extracted for accurate classification of motor imagery tasks.

5.4. Model Development

5.4.1. SVM Model

Transformation

Before extracting the features, first should be discussed which transformation is going to be used to extract the features from. EEG patterns during motor imagery/execution is an oscillation, and is thus non-stationary. Transformations that are specifically designed to extract information from non-stationary signals, are short-time Fourier transform (STFT) [9] and wavelet transform (WT) [6].

Features

Various features from different transformations, such as STFT, WT, time domain, and CSP, were included in the analysis. These features were then visualized using a boxplot or subjected to a Wilcoxon Signed-Rank Test to assess the Null Hypothesis H_0: There is no difference between Left and Right.

Concerning the time domain, the following features were extracted:

- **Mean Voltage**: average value of the EEG signal over a given period.
- **Standard Deviation**: measures the amount of variation or dispersion of the EEG signal values.
- **Variance**: the square of the standard deviation, representing the spread of the signal values around the mean.
- **Skewness**: measures the asymmetry of the signal distribution around the mean.

- **Kurtosis**: measures the ”tailedness” or peakedness of the signal distribution.
- **Root Mean Square (RMS)**: the square root of the average of the squares of the signal values.
- **Zero-Crossing Rate**: the rate at which the signal crosses the zero voltage level.
- **Waveform Length**: the cumulative length of the waveform over a given period.
- **Max Amplitude**: the highest voltage value observed in the signal.
- **Peak-to-Peak Amplitude**: the difference between the maximum and minimum voltage values in the signal.
- **Mean Absolute Value (MaV)**: the average of the absolute values of the signal.
- **Area Under the Curve (AUC)**: the integral of the signal over a given period, representing the total signal accumulation.

Below it shows the boxplot(5.4) of the time-domain:

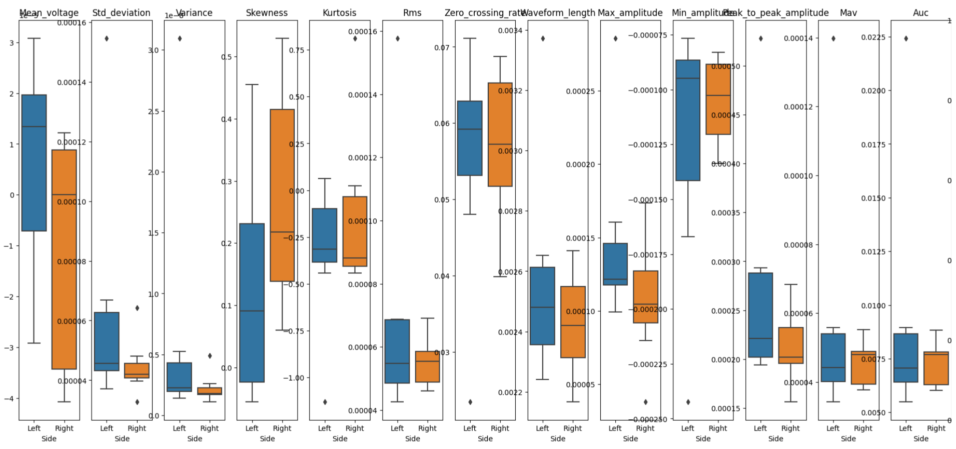

Figure 5.4: Boxplot Timedomain Features

As can be seen from the boxplot, some features show a big difference between left and right. From this, it can be concluded that some features are useful, for example, the feature Mean Voltage. The body of the right signal (orange) is larger than that of the left signal (blue), and they don't overlap. On the other hand, some features are not useful. For example, in MaV; the body of the right signal and the left signal are approximately the same. But the important thing is that the entire right signal body is inside the left signal body.

The Features for STFT and WT are the following:

- **Mu Band Power (STFT)**: the power of the EEG signal within the mu frequency band (8-13 Hz).
- **Beta Band Power(STFT)**: the power of the EEG signal within the beta frequency band (13-30 Hz).
- **Mean Power(STFT)**: average power of the EEG signal over all frequency bands.
- **Variance**: the spread of the power values around the mean power across the frequency bands.
- **Average Power(STFT)**: average power of the EEG signal, possibly over a specific period or condition.
- **SPD Average Power(STFT)**: average power of the signal power density (SPD), representing the average power per unit frequency.
- **Power(WT)**: overall power of the EEG signal as measured by wavelet transform.
- **Relative Power(WT)**: power of the EEG signal relative to a specific frequency band or total power.
- **Peak Frequency(WT)**: frequency at which the power spectrum density (PSD) reaches its maximum.
- **Bandwidth(WT)**: range of frequencies within which most of the signal's power is contained.

- **Spectral Entropy(WT)**: measure of the complexity or irregularity of the EEG signal's frequency distribution.

For the following features, Wilcoxon Signed-Rank Test was performed. The result was that the Null Hypothesis H_0 was rejected. Which means that there is a difference between left and right.

Additional features that are used are Phase Locking Value (PLV) and Common Spatial Pattern (CSP):

- **PLV**: measures the synchronization between two EEG signals by assessing how often their phases align over time. High PLV indicates strong synchronization, while low PLV indicates weak synchronization. During motor imagery tasks, like imagining left-hand movement, the right motor cortex becomes more active, leading to a larger phase difference with the left motor cortex, resulting in lower PLV. This change in synchronization patterns can be used to distinguish between different motor imagery tasks, enhancing the accuracy of brain-computer interfaces (BCIs).
- **CSP**: is a technique that extracts features from EEG data by finding spatial filters that maximize variance differences between two classes of brain signals, such as imagining left-hand versus right-hand movement. CSP highlights the differences in brain activity, making it easier to classify these signals accurately. By enhancing these distinctions, CSP improves the reliability of BCIs in interpreting motor imagery, leading to better performance in applications where users control devices with their thoughts.

Also here Wilcoxon Signed-Rank Test was performed. The result was that the Null Hypothesis H_0 was rejected. Which means that there is a difference between left and right.

The PLV and CSP were the most important features. For example, the accuracy didn't change significantly when only these features were used.

Hyperparameters tuning

The following parameters are tuned in the svm model:

- **Gamma**: Controls the influence range of training examples.
  - High gamma can lead to overfitting.
  - Low gamma may cause underfitting.
- **C**: Regularization parameter that balances training error and model complexity.
  - High C focuses on classifying all training examples correctly, possibly overfitting.
  - Low C results in a smoother decision surface.
- **Kernel**: Defines the transformation type applied to data to find an optimal boundary.
  - Common types include linear, polynomial, RBF, and sigmoid.
  - The choice of kernel significantly affects model performance.

To determine which values are the best for the parameters, Cross-Validation is performed. This is how **Cross-validation** works:

Cross-validation is a technique to estimate a model's performance and generalizability. It involves:

- Splitting the dataset into k folds.
- Training the model on $k-1$ folds.
- Validating on the remaining fold.
- Repeating the process k times, with each fold used as a validation set once.
- Averaging the results to ensure the model's robustness across different data subsets.

This method helps in tuning hyperparameters and ensures the model's robustness across different data subsets.

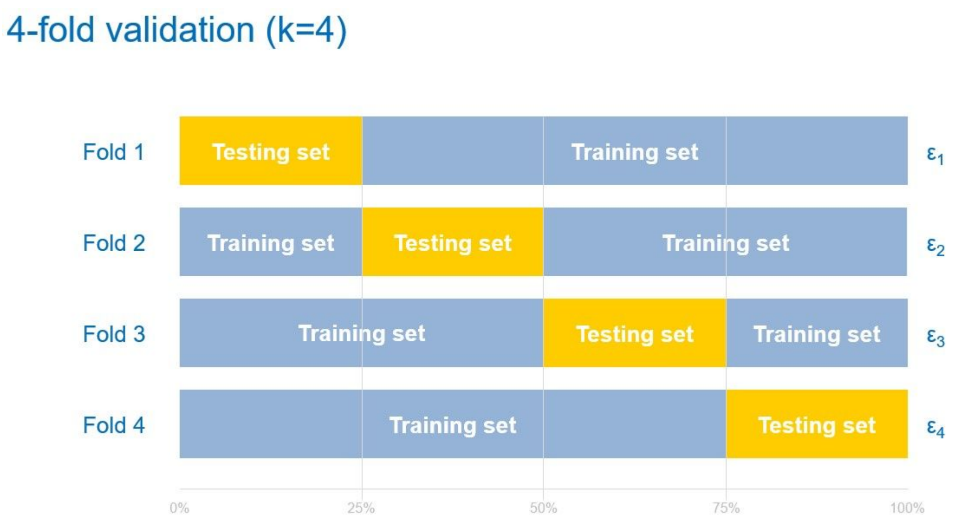

Figure 5.5: How Cross-Validation Works [14]

5.4.2. CNN Model

The neural network used for motor imagery classification is a Convolutional Neural Network (CNN). CNNs are chosen because they effectively capture patterns in EEG signals. Their layers learn and extract features from EEG data, which improves classification accuracy for different brain activities. CNNs also handle variations in EEG signals over time and space, making them ideal for interpreting complex brain patterns related to motor imagery [22]. According to research by Ali Al-Saegh et al., [2], CNNs are the most commonly used model for motor imagery classification, with a usage rate of 73.9%.

Input Preparation

Wavelet pseudo-images serve as inputs for the CNN model. Pseudo-images represent time-series data transformed into a format suitable for CNNs, resembling the structure of typical images used in CNNs. This representation includes:

- **Channels**: representing EEG channels.
- **Wavelet scales**: representing frequency ranges obtained from Continuous Wavelet Transform (CWT).
- **Time points**: representing the temporal dimension of EEG signals.

Wavelet transformation is applied independently to different frequency bands:

- **mu band** (8-13 Hz)
- **beta band** (14-30 Hz)

This results in 22 scales. Continuous Wavelet Transform is used approximately 45% of the time for creating these pseudo-images [2].

Neural Network Layers

CNN models incorporate combinations of ReLU layers, LSTM layers, Transformer layers, and Dropout layers.

ReLU (Rectified Linear Unit) layers are effective in CNNs because they introduce non-linearity, enabling complex pattern learning in EEG data. They also enhance computational efficiency and prevent vanishing gradients, thereby improving classification performance [4].

LSTM (Long Short-Term Memory) layers** are beneficial in CNNs for capturing temporal dependencies in EEG signals, enhancing overall classification accuracy [31].

Transformer layers excel in capturing long-range dependencies and handling temporal dynamics in EEG signals. They incorporate an attention mechanism that focuses on relevant parts of the input sequence, leading to improved classification performance.

Dropout layers are used to prevent overfitting in CNNs by randomly dropping units during training. This regularization technique enhances model generalization and performance on unseen data, [4].

Combining CNNs, ReLU, LSTM, Transformer, and Dropout layers enhances motor imagery classification accuracy. CNNs extract spatial features from EEG signals, ReLU layers introduce non-linearity, LSTM layers capture temporal dependencies, Transformers manage long-range dependencies with attention mechanisms, and Dropout layers prevent overfitting, collectively improving model robustness and accuracy [4].

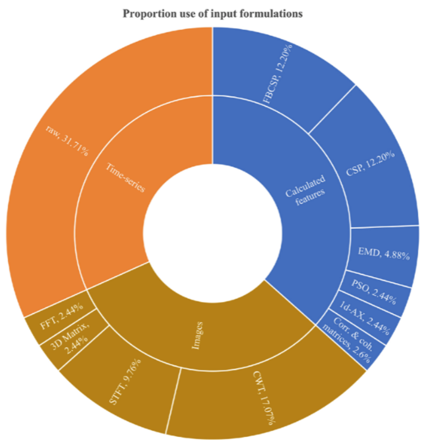

Figure 5.6: Proportion use of input formulation types across all the reviewed studies with the used techniques for achieving them., [2].

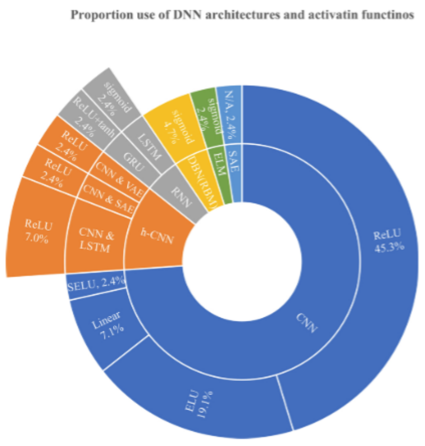

Figure 5.7: Proportion use of the DNN architectures across all the reviewed studies with the employed activation functions, [2].

Hyperparameter Tuning

Their is no scheme/recipe to get the best set hyperparameters for achieving the best preforming model, the selection of the paramaters is done by trail and error. The parameters that play a role in the performance of the CNN model are:

- **Depth of Layers**: The number of convolutional layers in the network. According to Schirrmeister et al., [22], deeper networks tend to yield better results in EEG classification by capturing complex patterns. Due to resource constraints (using personal laptops), the model was limited to 3 convolutional layers to balance performance and runtime.
- **Number of Filters**: The depth of the output volume from a convolutional layer. For this study, 32, 64, and 128 filters were chosen per layer, suitable for extracting meaningful features from EEG data.

- **Filter Size (Kernel Size)**: The dimensions of convolutional filters. A 3x3 kernel size was selected as it effectively captures spatial hierarchies in EEG signals, aligning with findings from Schirrmeister et al., [22].
- **Stride**: Is the number of pixels by which the filter moves across the input matrix. In the study of Schirrmeister et al., [22], the typically used stride size is 1. This allowed the model to capture fine-grained spatial information in EEG signals, which is crucial for MI classification.
- **Padding**: Extra layers of zeros around the input matrix to preserve spatial dimensions after convolution. The `'padding=1'`' has been applied to each convolutional layer, `'padding=1'`' means "same" padding, which ensures that the output feature map has the same spatial dimensions as the input.
- **Pooling**: Operations like max pooling or average pooling to downsample the input. Some studies in EEG classification have shown that both average and max pooling can be effective depending on the specific context and preprocessing steps applied to EEG data. During the developement of the model average pooling has been used, with the idea that it preserves more information about all parts of the input, compared to max pooling.
- **Regularization Method**: Regularization methods refer to techniques used to prevent overfitting and improve generalization capabilities. Dropout Rate has been chosen, which is the probability of neurons being randomly dropped out during training. In the review of Ali Al-Saegh et al., [2], the most used regularization method across the 40 papers was Dropout Rate, see figure A.3.
- **Learning Rate**: Determines how much to change the model in response to the estimated error. A starting point of 0.001 was chosen, which is a commonly used starting point and is often considered a reasonable choice for many deep learning tasks, including motor imagery tasks.
- **Batch Size**: Number of training examples used per iteration. A batch size of 256 was chosen for the Physionet dataset to balance runtime efficiency given its size (5000 samples). For the BCIC IV 2a dataset, a batch size of 64 was selected due to its smaller sample size. For the data of the Data Acquisition group a batch size of 4 has been used, due the scarcity of the data.
- **Optimizer**: Type of optimization algorithm used. The used optimizer it the Adam Optimizer, In the review of Ali Al-Saegh et al., [2], the Adam Optimizer was the most used, with an usage rate of 47%, see 5.9.

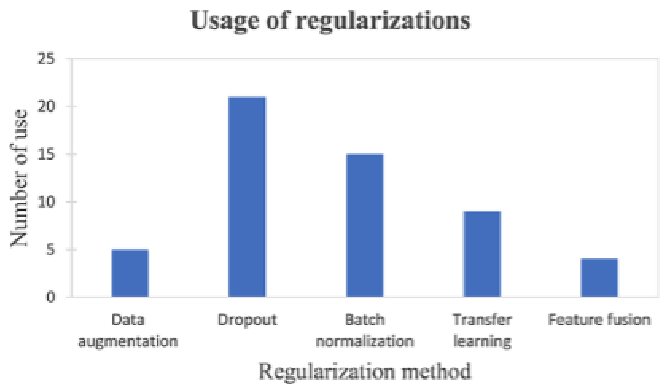

Figure 5.8: Amount of using the regularization methods across the reviewed articles, [2].

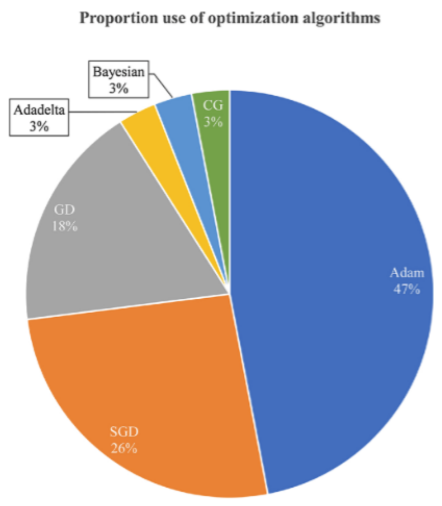

Figure 5.9: Proportion use of the different optimization algorithms across all the reviewed studies, [2].

5.5. Scalability and User-Specific Models

5.5.1. Scalability

Scalability is crucial for enhancing the performance of classification models in EEG motor imagery (MI) applications. As the amount of data increases, the model can learn from a more diverse set of examples, leading to better generalization and accuracy.

Our scripts are designed to automatically detect and integrate new data uploaded by the Data Acquisition group. This process uses online learning techniques to retrain the model incrementally. By continuously updating the model with new data, we ensure that it remains up-to-date and improves over time. This approach is supported by research of Steven C.H. Hoi et. [32], which provides empirical evidence showing that more data generally leads to improved classification accuracy.

When a new user is added, their calibration dataset also contributes to the scalability of our data. This means that each new user's data not only helps in personalizing their model but also enhances the overall dataset, benefiting the general performance of the system. Studies have shown that including diverse user data can improve the robustness and accuracy of EEG-based models, [22].

5.5.2. User-Specific Models

Motor imagery activity varies significantly from person to person. This variation necessitates the use of user-specific models to achieve optimal performance. Research has demonstrated that tailoring models to individual users can substantially improve classification accuracy, [13].

In our implementation, when a new user is added, we start with the latest updated pre-trained model and fine-tune it using the user's calibration dataset. This fine-tuning process involves giving more weight to the calibration data, emphasising the model's user-specific characteristics. This method has effectively adapted pre-trained models to individual users, enhancing the user-specific performance of EEG classification systems, [28].

Our system aims to provide high accuracy and personalised performance in EEG motor imagery applications by focusing on scalability and user-specific adaptation.

5.6. Evaluation Metrics

The following methods are used to evaluate the models:

Accuracy

Accuracy is the proportion of correctly predicted instances out of the total instances. It shows how often the classifier is right overall. The formula is as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{5.1}$$

Accuracy is a simple and widely used measure that gives a general idea of how well the model performs. It's especially helpful for datasets where the number of items in each group is about the same. For BCI motor imagery, where the detection of motor imagery signals is critical, accuracy helps in understanding the overall effectiveness of the model.

Precision

Precision measures the accuracy of the positive predictions by calculating the ratio of correct positive predictions to all predicted positive outcomes. Below is the formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \tag{5.2}$$

It's important to be precise when the cost of making a wrong guess is high. For BCI motor imagery, being precise means accurately detecting the signals related to imagining movement, which reduces the chances of the system giving out the wrong commands. This is important for users to trust the system and for it to work reliably.

Recall

Recall (or sensitivity) refers to the proportion of correctly identified positive outcomes compared to all positive outcomes. It indicates the model's ability to recognise all positive instances accurately.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \tag{5.3}$$

In BCI motor imagery, picking up on all the relevant signals is crucial. If we miss a valid signal, the user's intended action might not be recognised, which can be a big problem for assistive technologies and communication devices. High recall helps ensure the system is sensitive to all the right motor imagery signals.

F1 Score

The F1 Score is a method of combining precision and recall into a single measure. It provides a balanced view of both aspects. The formula can be seen below:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}} \tag{5.4}$$

The F1 Score is especially useful for imbalanced datasets where one class might be significantly underrepresented. In BCI motor imagery, where certain motor imagery signals might be rare compared to others, the F1 Score provides a balanced evaluation, ensuring that both precision and recall are considered. This helps in developing a more reliable BCI system that accurately detects and responds to motor imagery signals.

6 Results

6.1. SVM

The SVM model is performed on different datasets; the results of each dataset are shown below.

6.1.1. Physionet

Table 6.1: Performance Metrics for SVM on Physionet Dataset

Metric	All Subjects	One Subject (average)
Accuracy	59%	66%
Precision	56%	63%
Recall	55%	64%
F1 Score	56%	65%

6.1.2. BCIC IV 2a

Table 6.2: Performance Metrics for SVM on BCIC IV 2a Dataset

Metric	All Subjects	One Subject (average)
Accuracy	65%	78%
Precision	64%	74%
Recall	62%	74%
F1 Score	64%	75%

6.1.3. Data Acquisition group

Table 6.3: Performance Metrics for SVM on Data Acquisition Dataset

Metric	One Subject (average)
Accuracy	55%
Precision	54%
Recall	55%
F1 Score	54%

6.1.4. Comparative Analysis

As can be seen from the results above, the BCIC IV 2a dataset performs the highest for this model, with an accuracy of 65% for all subjects. After that comes Physionet with 59% and, at last, the Data Acquisition group with 55%.

6.2. CNN

6.2.1. Models

During the experiments, two advanced deep learning models were employed: the Advanced-EEG-MI-CNN and Advanced-EEG-MI-CNN-LSTM-Transformer mo. These models were evaluated to compare their performance on the same EEG dataset. Below is a detailed description of the experimental setup.

- **Advanced-EEG-MI-CNN:**
  - Convolutional Layers: The model consists of three convolutional layers with batch normalisation and average pooling. Each convolutional layer uses a 3x3 kernel with a padding of 1 to preserve the spatial dimensions of the input.
  - Fully Connected Layers: Following the convolutional layers, the output is flattened and passed through three fully connected layers with ReLU activations and dropout for regularisation. The final output layer has two neurons corresponding to the two classes in the MI task: left-hand movement and right-hand movement.
- **Advanced-EEG-MI-CNN-LSTM-Transformer:**
  - Convolutional Layers: Similar to the Advanced-EEG-MI-CNN model, this architecture starts with three convolutional layers with batch normalisation and average pooling.
  - LSTM Layer: The output from the convolutional layers is reshaped and fed into an LSTM layer to capture temporal dependencies in the data.
  - Transformer Encoder: Following the LSTM, a Transformer encoder layer is applied to further model sequential dependencies. The output from the Transformer is used for the final classification.
  - Fully Connected Layers: The final part of the network consists of three fully connected layers with ReLU activations and dropout, leading to the output layer with two neurons.

6.2.2. Results

Model: Advanced-EEG-MI-CNN

Physionet Dataset

Table 6.4: Performance Metrics for Advanced-EEG-MI-CNN on Physionet Dataset

Metric	All Subjects	One Subject (average)
Accuracy	65%	78%
Precision	62%	74%
Recall	58%	73%
F1 Score	60%	73%

BCIC IV 2a Dataset

Table 6.5: Performance Metrics for Advanced-EEG-MI-CNN on BCIC IV 2a Dataset

Metric	All Subjects	One Subject (average)
Accuracy	80%	92%
Precision	76%	87%
Recall	74%	83%
F1 Score	75%	85%

Data Acquisition Dataset

Table 6.6: Performance Metrics for Advanced-EEG-MI-CNN on Data Acquisition Dataset

Metric	One Subject
Accuracy	54%
Precision	51%
Recall	48%
F1 Score	49%

Model: Advanced-EEG-MI-CNN-LSTM-Transformer

Physionet Dataset

Table 6.7: Performance Metrics for Advanced-EEG-MI-CNN-LSTM-Transformer on Physionet Dataset

Metric	All Subjects	One Subject (average)
Accuracy	63%	72%
Precision	59%	68%
Recall	56%	66%
F1 Score	58%	66%

BCIC IV 2a Dataset

Table 6.8: Performance Metrics for Advanced-EEG-MI-CNN-LSTM-Transformer on BCIC IV 2a Dataset

Metric	All Subjects	One Subject (average)
Accuracy	78%	92%
Precision	74%	82%
Recall	70%	87%
F1 Score	72%	85%

Data Acquisition Dataset

Table 6.9: Performance Metrics for Advanced-EEG-MI-CNN-LSTM-Transformer on Data Acquisition Dataset

Metric	One Subjects (average)
Accuracy	53%
Precision	47%
Recall	50%
F1 Score	49%

6.2.3. Comparative Analysis

The BCIC IV 2a dataset consistently showed higher accuracy than the Physionet dataset across both Advanced-EEG-MI-CNN and Advanced-EEG-MI-CNN-LSTM-Transformer models. For all subjects, the accuracy for the BCIC IV 2a dataset was notably higher, with Advanced-EEG-MI-CNN achieving 80% and Advanced-EEG-MI-CNN-LSTM-Transformer achieving 78%. In contrast, the Physionet dataset yielded lower accuracies of 65% for Advanced-EEG-MI-CNN and 63% for Advanced-EEG-MI-CNN-LSTM-Transformer. This trend was also evident when evaluating a single subject, where the BCIC IV 2a dataset again outperformed the Physionet dataset. The own dataset showed significantly lower accuracy, with both models achieving only around 50%, suggesting that it may not contain enough data to effectively differentiate between the two classes.

The analysis highlights the differences in performance between the Physionet, BCIC, and Data Acquisition datasets using Advanced-EEG-MI-CNN and Advanced-EEG-MI-CNN-LSTM-Transformer models. The BCIC dataset demonstrated higher accuracy across both models, suggesting it is a more reliable dataset for these types of analyses. The Acquisition dataset has a low accuracy indicates that the data may be insufficient or not well-suited for class differentiation. Future research should focus on improving the quality of the Physionet and Data Acquisition datasets or exploring additional datasets to ensure robust model performance. Further investigation into precision, recall, and F1 score metrics will provide a more comprehensive understanding of model efficacy.

6.3. User Specific Models

Due to time constraints, we were unable to experiment with varying the weights on the user dataset to create user specific models. This prevented us from fully implementing a comprehensive evaluation to individual users.

7 Discussion

This research aimed to develop a BCI system that can effectively distinguish between left-hand and right-hand motor imagery using EEG signals. This section discusses the findings, compares the performance of different models, and outlines potential future work.

Performance of SVM Model

The SVM model was tested on the BCIC IV 2a dataset, achieving an average accuracy of 65% for all subjects and 78% for a single subject. The precision, recall, and F1 score metrics also indicated moderate performance, with the highest values being 74% for precision and 73% for the F1 score in the single-subject scenario. This suggests that while the model shows potential for direct implementation in gaming experiences, it is not yet developed enough for practical use. Its performance is not groundbreaking, and significant progress is needed for medical applications to achieve the high accuracy required in this field.

Performance of CNN Models

Two CNN models were developed: the Advanced-EEG-MI-CNN and the Advanced-EEG-MI-CNN-LSTM-Transformer. The Advanced-EEG-MI-CNN model performed with an accuracy of 80% for all subjects and 90% for a single subject. Adding LSTM and Transformer layers in the Advanced-EEG-MI-CNN-LSTM-Transformer model aimed to capture temporal dependencies in the data. However, this model did not significantly outperform the simpler CNN model, suggesting that the temporal dependencies were not as critical for this specific task or that the implementation requires further refinement. Once again, the model shows potential but requires significant progress to be effectively utilised in the gaming and medical fields.

Comparative Analysis

A comparative analysis revealed that while the Advanced-EEG-MI-CNN and Advanced-EEG-MI-CNN-LSTM-Transformer models showed similar accuracy levels, the simpler Advanced-EEG-MI-CNN was more computationally efficient. The SVM model achieved an accuracy of 65% across all subjects and 78% for a single subject; as for the Advanced-EEG-MI-CNN, it was 80% across all subjects and 92% for a single subject. Adding LSTM and Transformer layers in the Advanced-EEG-MI-CNN-LSTM-Transformer model did not provide a significant performance improvement over the simpler CNN model. In practical applications with limited computational resources, the Advanced-EEG-MI-CNN may be preferable due to its efficiency. Further refinement and optimisation are necessary for all models to achieve higher accuracy and practical usability in gaming and medical applications.

The model performance on the self-gathered data is significantly lower than the objective performance, primarily due to the scarcity of the data. With more data, the accuracy of the Convolutional Neural Network (CNN) models would likely improve, as they would have more examples to learn from. Furthermore, the blink noise was not correctly removed, which could be a contributing factor to the low validation accuracy observed during the training of the CNN models.

Future Work

Due to the time constraint of 10 weeks, not every objective could be completed or optimised. The remaining work necessary to achieve more respectable results includes:

- **Enhancing User-Specific Model Implementation:** Future research should focus on refining user-specific models to better adapt to individual differences in EEG signals.

- **Applying ICA for Data Cleaning:** Implementing ICA can help isolate and remove noise from EEG data, leading to cleaner signals and potentially higher classification accuracy.

- **Improving Data Preprocessing:** Further efforts are needed to enhance the preprocessing steps to minimise noise and artifacts in the EEG data, thereby improving the overall performance of the models.

8 Conclusion

This research successfully developed and evaluated an SVM model and two advanced deep-learning models for BCI motor imagery classification. The models demonstrated moderate accuracy levels, with the Advanced-EEG-MI-CNN model achieving up to 80% accuracy for All subjects. The SVM model, tested on the BCIC IV 2a dataset, also achieved an accuracy of 65% across all subjects. The precision, recall, and F1 score metrics indicated moderate performance, with the highest values from the Advanced-EEG-MI-CNN model being 76% for precision, 74% for recall and 75% for the F1 score in the single-subject scenario.

While adding LSTM and Transformer layers in the Advanced-EEG-MI-CNN-LSTM-Transformer model did not significantly enhance performance, it provided valuable insights into the potential and limitations of these architectures. This underscores the importance of user-specific models and highlights areas for future improvement, including data-cleaning techniques and preprocessing methods.

These findings lay the groundwork for more robust and scalable BCI systems, with the potential for significant impact in assistive technologies and beyond. Further refinement and optimisation are necessary for all models to achieve higher accuracy and practical usability in gaming and medical applications.

References

[1] In: (). URL: http://www.bci2000.org.

[2] Jassim M. Abdul-Jabbar Ali Al-Saegh Shefa A. Dawwd. “Deep learning for motor imagery EEG-based classification: A review”. In: *ScienceDirect* (January 2021). URL: https://www.sciencedirect.com/science/article/pii/S1746809420303116.

[3] Zaid Abdi Alkareem Alyasseri et al. “EEG Signals Denoising Using Optimal Wavelet Transform Hybridized With Efficient Metaheuristic Methods”. In: *IEEE* (2019). DOI: 10.1109/ACCESS.2019.2962658. URL: https://ieeexplore.ieee.org/document/8944069.

[4] Syed Umar Amin et al. “Deep Learning for EEG motor imagery classification based on multi-layer CNNs feature fusion”. In: *ScienceDirect* (Dec. 2019). URL: https://www.sciencedirect.com/science/article/pii/S0167739X19306077.

[5] Alejandro Ito Aramendia. “Convolutional Neural Networks (CNNs) : A Complete Guide”. In: *Medium* (Jan 1, 2024). URL: https://medium.com/@alejandro.itoaramendia/convolutional-neural-networks-cnns-a-complete-guide-a803534a1930.

[6] Mohammed Azmi Al-Betar et al. “EEG Signals Denoising Using Optimal Wavelet Transform Hybridized With Efficient Metaheuristic Methods”. In: *IEEE* (Dec. 2019). DOI: 10.1109/ACCESS.2019.2962658. URL: https://ieeexplore.ieee.org/document/8944069.

[7] C. Brunner et al. “BCI Competition 2008 – Graz data set A”. In: *Institute for Knowledge Discovery, Graz University of Technology, Austria; Institute for Human-Computer Interfaces, Graz University of Technology, Austria* (2008). URL: https://www.bbci.de/competition/iv/.

[8] Liliya A. Demidova, Rostislav A. Isaev, and Ruslan R. Salyamov. “Kernel-Based Embedded Feature Selection for Motor Imagery Based BCI”. In: *IEEE Transactions* (2023). DOI: 10.1109/SUMMA60232.2023.103 URL: https://ieeexplore.ieee.org/document/10349633.

[9] Moncef Gabbouj, Kaveh Samiee, and Péter Kovács. “Epileptic Seizure Classification of EEG TimeSeries Using Rational Discrete Short-Time Fourier Transform”. In: *IEEE* (Sept. 2014). DOI: 10.1109/TBME.2014.2360101. URL: https://ieeexplore.ieee.org/document/6909003.

[10] Mohammad-Parsa Hosseini, Amin Hosseini, and Kiarash Ahi. “A Review on Machine Learning for EEG Signal Processing in Bioengineering”. In: *IEEE* (2020). DOI: 10.1109/RBME.2020.2969915. URL: https://ieeexplore.ieee.org/document/8972542.

[11] Mehdi Kamandar. “Kernel-Based Embedded Feature Selection for Motor Imagery Based BCI”. In: *IEEE Transactions on Affective Computing* (2023). DOI: 10.1109/ICEE59167.2023.10334784. URL: https://ieeexplore.ieee.org/document/10334784.

[12] Deepak Kumar. *The Role of SVM Model in Current Data Science*. Mar. 2021. URL: https://www.linkedin.com/pulse/role-svm-model-current-data-science-deepak-kumar/.

[13] F. Lotte et al. “A review of classification algorithms for EEG-based brain-computer interfaces: a 10 year update”. In: *National Library of Medicine* (Feb. 2018). URL: https://pubmed.ncbi.nlm.nih.gov/29488902/.

[14] MathWorks. *Cross-Validation*. 2023. URL: https://ch.mathworks.com/discovery/cross-validation.html.

[15] “MNE Preprocessing”. In: *LiberianGeek* (). URL: https://mne.tools/dev/auto_tutorials/preprocessing/index.html.

[16] J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. “Designing optimal spatial filters for single-trial EEG classification in a movement task”. In: *ScienceDirect* (1999). DOI: 0.1016/S1388-2457(98)00038-8. URL: https://www.sciencedirect.com/science/article/abs/pii/S1388245798000388?via%3Dihub.

[17] Gert Pfurtscheller hrista Neuper Michael Wörtz. “ERD/ERS patterns reflecting sensorimotor activation and deactivation”. In: *Pubmed* (2006). DOI: 10.1016/S0079-6123(06)59014-4. URL: https://pubmed.ncbi.nlm.nih.gov/17071233/.

[18] Premanand S. *Support Vector Machine: A Better Understanding*. Nov. 2023. URL: https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/.

[19] Kaveh Samiee, Péter Kovács, and Moncef Gabbouj. “Epileptic Seizure Classification of EEG Time-Series Using Rational Discrete Short-Time Fourier Transform”. In: *IEEE* (2014). DOI: 10.1109/TBME.2014.23 URL: https://ieeexplore.ieee.org/document/6909003.

[20] G. Schalk et al. “BCI2000: A General-Purpose Brain-Computer Interface (BCI) System”. In: *IEEE Transactions on Biomedical Engineering* 51.6 (2004), pp. 1034–1043.

[21] Robin Tibor Schirrmeister et al. “Deep learning with convolutional neural networks for EEG decoding and visualization”. In: *Human Brain Mapping* (2017). DOI: 10.1002/hbm.23730. URL: https://onlinelibrary.wiley.com/doi/10.1002/hbm.23730.

[22] Robin Tibor Schirrmeister et al. “Deep learning with convolutional neural networks for EEG decoding and visualization”. In: *National Library of Medicine* (). URL: https://pubmed.ncbi.nlm.nih.gov/28782865/.

[23] A. Schlögl et al. “Characterization of four-class motor imagery eeg data for the bci-competition 2005”. In: *Journal of Neural Engineering* 2.4 (2005), p. L14.

[24] Jurgen Schmidhuber Sepp Hochreiter. “LONG SHORT-TERM MEMORY”. In: (1997). URL: https://www.bioinf.jku.at/publications/older/2604.pdf.

[25] Lavanya Shukla. *Designing Your Neural Networks*. URL: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed (visited on 09/19/2019).

[26] Michael Stolzer. “How to Design LSTM Network Using PyTorch”. In: *LiberianGeek* (Jan 30, 2024). URL: https://www.liberiangeek.net/2024/01/design-lstm-network-using-pytorch/.

[27] “ULTRACORTEX ”MARK IV” EEG HEADSET”. In: (). URL: https://shop.openbci.com/products/ultracortex-mark-iv.

[28] Alexandre Barachant Vinay Jayaram. “MOABB: trustworthy algorithm benchmarking for BCIs”. In: *IOPscience* (25 September 2018). URL: https://iopscience.iop.org/article/10.1088/1741-2552/aadea0.

[29] Jonathan R Wolpaw et al. “Brain-computer interfaces for communication and control”. In: *Pubmed* (2002). DOI: 10.1016/s1388-2457(02)00057-3. URL: https://pubmed.ncbi.nlm.nih.gov/12048038/.

[30] Yann Lecun Y. Bengio. “Convolutional Networks for Images, Speech, and Time-Series”. In: *ResearchGate* (November 1997). URL: https://www.researchgate.net/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series.

[31] Hussain Montazery Kordy Zahra Khademi Farideh Ebrahimi. “A transfer learning-based CNN and LSTM hybrid deep learning model to classify motor imagery EEG signals.” In: *Read.qxmd* (2022 Februrary 11). URL: https://read.qxmd.com/read/35168083/a-transfer-learning-based-cnn-and-lstm-hybrid-deep-learning-model-to-classify-motor-imagery-eeg-signals.

[32] Chiyuan Zhang et al. “Understanding deep learning requires rethinking generalization”. In: *Cornell University* (Feb. 26, 2017). URL: https://arxiv.org/abs/1611.03530.

A
Source Code

Here are the links to the GitHub repositories:

- CNN
- SVM

A.1. CNNs models

A.1.1. Advanced-EEG-MI-CNN

```python
"""
The Advanced_EEG_MI_CNN implementation.
"""
class Advanced_EEG_MI_CNN(nn.Module):
    def __init__(self):
        super(Advanced_EEG_MI_CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=8, out_channels=32, kernel_size=(3, 3), padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.pool1 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(3, 3), padding
            =1)
        self.bn2 = nn.BatchNorm2d(64)
        self.pool2 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(3, 3), padding
            =1)
        self.bn3 = nn.BatchNorm2d(128)
        self.pool3 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))
        self.dropout = nn.Dropout(p=0.5)

        # Compute the size of the input to the fully connected layer
        dummy_input = torch.zeros(1, 8, 43, 1251)
        dummy_output = self._forward_conv_layers(dummy_input)
        fc_input_size = dummy_output.numel()

        self.fc1 = nn.Linear(fc_input_size, 256)
        self.fc2 = nn.Linear(256, 64)
        self.fc3 = nn.Linear(64, 2)

    def _forward_conv_layers(self, x):
        x = self.pool1(F.relu(self.bn1(self.conv1(x))))
        x = self.pool2(F.relu(self.bn2(self.conv2(x))))
        x = self.pool3(F.relu(self.bn3(self.conv3(x))))
        return x

    def forward(self, x):
        x = self._forward_conv_layers(x)
        x = x.view(x.size(0), -1)
        x = self.dropout(F.relu(self.fc1(x)))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

A.1.2. Advanced-EEG-MI-CNN-LSTM-Transformer

```
"""
The Advanced_EEG_MI_CNN_LSTM_Transformer implementation.
"""
class Advanced_EEG_MI_CNN_LSTM_Transformer(nn.Module):
    def __init__(self):
        super(Advanced_EEG_MI_CNN_LSTM_Transformer, self).__init__()

        # Convolutional layers
        self.conv1 = nn.Conv2d(in_channels=8, out_channels=32, kernel_size=(3, 3), padding=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.pool1 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))

        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(3, 3), padding
            =1)
        self.bn2 = nn.BatchNorm2d(64)
        self.pool2 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))

        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(3, 3), padding
            =1)
        self.bn3 = nn.BatchNorm2d(128)
        self.pool3 = nn.AvgPool2d(kernel_size=(2, 2), stride=(2, 2))

        # LSTM layer
        self.lstm = nn.LSTM(input_size=128*6*157, hidden_size=128, num_layers=1, batch_first=
            True)

        # Transformer layer
        self.transformer_encoder_layer = nn.TransformerEncoderLayer(d_model=128, nhead=8)
        self.transformer_encoder = nn.TransformerEncoder(self.transformer_encoder_layer,
            num_layers=1)

        # Fully connected layers
        self.fc1 = nn.Linear(128, 256)
        self.fc2 = nn.Linear(256, 64)
        self.fc3 = nn.Linear(64, 2)

        # Dropout layer
        self.dropout = nn.Dropout(p=0.5)

    def _forward_conv_layers(self, x):
        x = self.pool1(F.relu(self.bn1(self.conv1(x))))
        x = self.pool2(F.relu(self.bn2(self.conv2(x))))
        x = self.pool3(F.relu(self.bn3(self.conv3(x))))
        return x

    def forward(self, x):
        batch_size = x.size(0)
        # Forward pass through convolutional layers
        x = self._forward_conv_layers(x)

        # Flatten for LSTM
        x = x.view(batch_size, -1, 128*6*157)

        # Forward pass through LSTM
        x, (hn, cn) = self.lstm(x)

        # Transformer encoding
        x = self.transformer_encoder(x)

        # Take the output from the last sequence step for classification
        x = x[:, -1, :]

        # Forward pass through fully connected layers
        x = self.dropout(F.relu(self.fc1(x)))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

A.2. Results

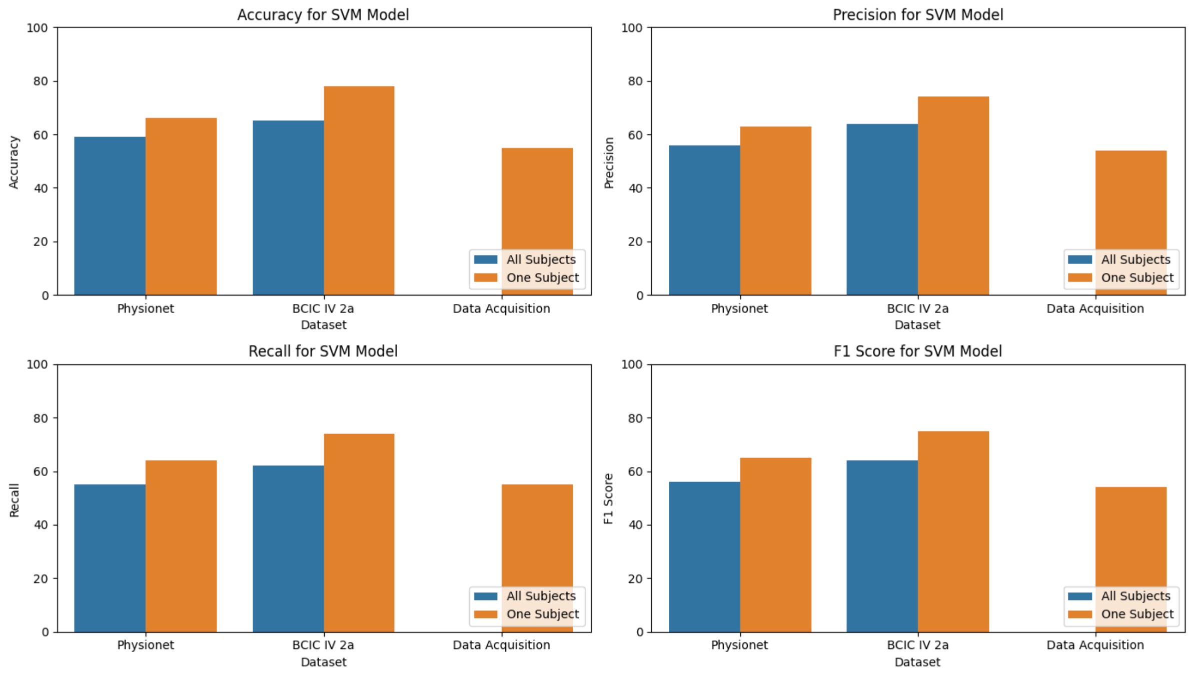

Figure A.1: Overview of performance of SVM model across the datasets.

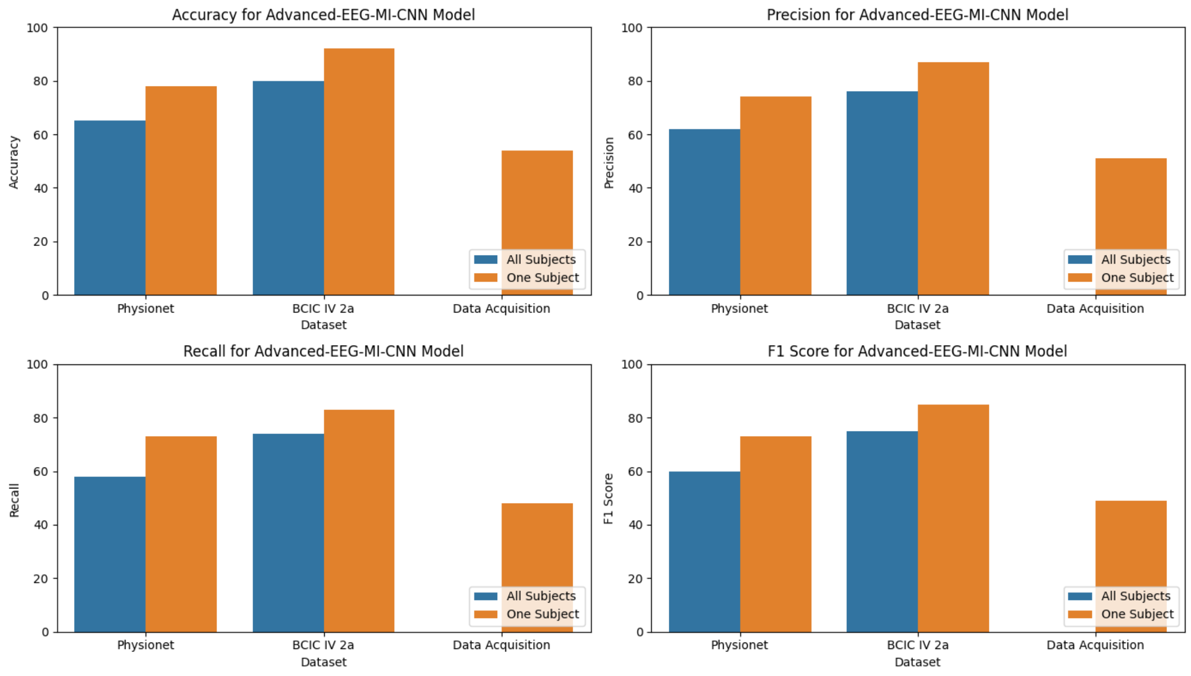

Figure A.2: Overview of performance of Advanced-EEG-MI-CNN model across the datasets.

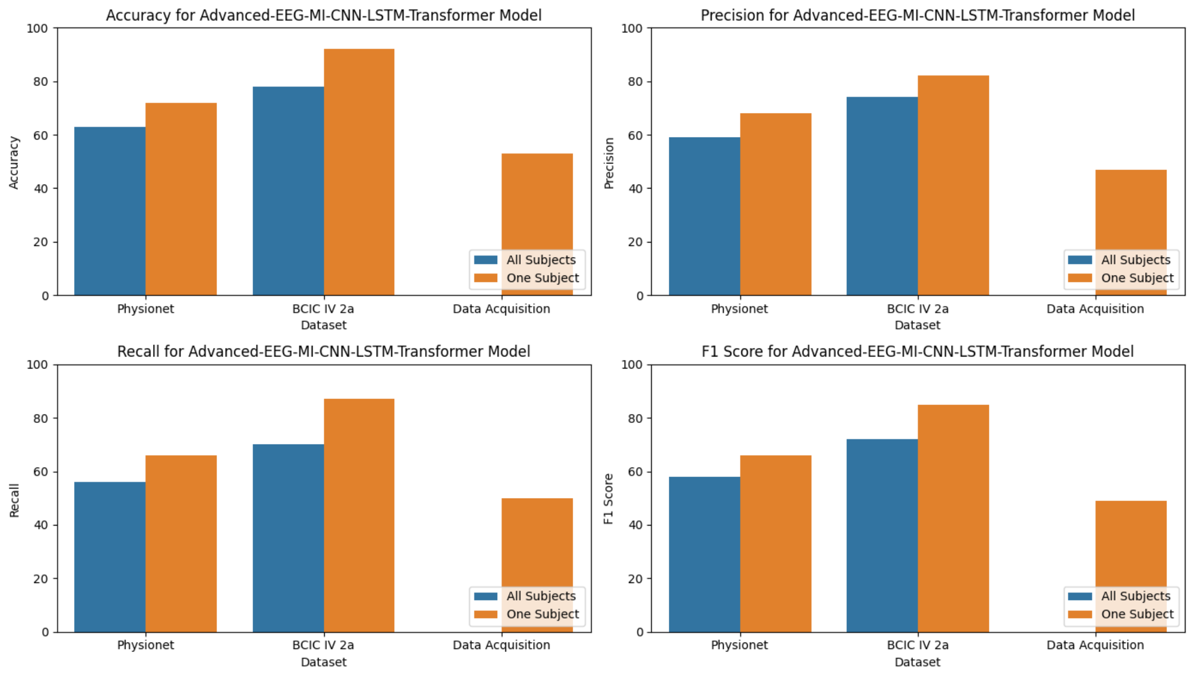

Figure A.3: Overview of performance of Advanced-EEG-MI-CNN-LSTM-Transformer model across the datasets.

B Statement use of AI

AI tools were used while writing this report. To summarize curtain papers and explain methods. And improve the vocabulary and make self-written alinea's more concise. Grammarly is also used to correct our grammar throughout the paper.