



**Advancements in EEG Signal Classification: Deep
Learning Approaches and Hardware Implementation on
FPGA**

Thesis for the Degree of Doctor of Philosophy (PhD)

Yu Xie

Supervisor: Dr. Istvan Oniga

UNIVERSITY OF DEBRECEN

Doctoral Council for Natural Sciences and Engineering

Doctoral School of Informatics

Debrecen, 2025

Hereby I declare that I prepared this thesis within the Doctoral Council for Natural Sciences and Engineering, Doctoral School of Informatics, University of Debrecen in order to obtain a PhD Degree in Engineering at Debrecen University.

The results published in the thesis are not reported in any other PhD theses.

Debrecen, 2025.

.....
signature of the candidate

Hereby I confirm that Yu Xie candidate conducted his/her studies with my supervision within the Information Technology Systems and Networks with Industrial applications Doctoral Program of the Doctoral School of Informatics between 2019 and 2025 The independent studies and research work of the candidate significantly contributed to the results published in the thesis.

I also declare that the results published in the thesis are not reported in any other theses.

I support the acceptance of the thesis.

Debrecen, 2025.

.....
signature of the supervisor

Advancements in EEG Signal Classification: Deep Learning Approaches and Hardware Implementation on FPGA

Dissertation submitted in partial fulfilment of the requirements for the doctoral (PhD)
degree
in Informatics

Written by Yu Xie certified Computer Science Engineer

Prepared in the framework of the Doctoral School of Informatics doctoral school of
the University of Debrecen
(Information Technology Systems and Networks with Industrial applications
programme)

Dissertation supervisor: Dr. Istvan Oniga

The official opponents of the dissertation:

Dr.
Dr.

The evaluation board:

chairperson: Dr.
members: Dr.
Dr.
Dr.
Dr.

The date and venue of the dissertation defence: 2025...

Contents

1.	Introduction	1
1.1.	Background and Motivation.....	1
1.2.	Challenges in the research.....	3
1.3	Structure of the Thesis	4
2.	Review of EEG signal processing methods and classification algorithm....	7
2.1.	Pre-processing of EEG signal	8
2.1.1	Principal component analysis (PCA)	8
2.1.2	Independent component analysis (ICA).....	9
2.1.3	Common average reference (CAR)	10
2.1.4	Common Spatial patterns (CSP)	10
2.1.5	Adaptive filter.....	11
2.1.6	Digital filters	12
2.2.	Feature Extraction	13
2.2.1	Power Spectrum Density (PSD)	13
2.2.2	Principal Component Analysis	14
2.2.3	Independent Component Analysis	15
2.2.4	Auto-Regressive (AR)	15
2.2.5	Wavelet transform (WT).....	16
2.3.	Classification.....	17
2.3.1	Artificial Neural Network (ANN).....	17
2.3.2	Deep Learning	18
2.3.3	K-Nearest Neighbors (KNN)	19
2.3.4	Linear Discriminant Analysis (LDA)	20
2.3.5	Support Vector Machine (SVM).....	20
2.3.6	Naive Bayes (NB).....	22

2.3.7	Decision Tree (DT).....	22
3.	Review of Hardware Acceleration Techniques for EEG Signals	25
3.1	EEG-BASED HARDWARE SOLUTIONS	25
3.1.1	Pre-processing.....	26
3.1.2	Feature Extraction.....	28
3.2	Efficient Convolutional Neural Networks.....	35
3.2.1	Pruning.....	36
3.2.2	Quantization.....	38
3.2.3	Tensor Decomposition.....	40
3.2.4	Knowledge Distillation	42
3.2.5	Neural Architecture Search (NAS)	44
4.	Research on EEG Signal Algorithm of Eye State	51
4.1	Eye State Research on PhysioNet data.....	51
4.1.1	PhysioNet EEG Dataset	51
4.1.2	Data analysis and processing	52
4.1.3	Construction of a multilayer perceptron (MLP) network	55
4.1.4	Construction of Simplified Convolutional Neural Network (SCNN)	57
4.1.5	Comparison results of MLP and SCNN.....	61
4.1.6	Conclusion	63
4.2	CNN efficiency test on own data	63
4.2.1	Data collection and preprocessing	63
4.2.2	The proposed network structure	68
4.2.3	Comparison results of CWT-CNN, STFT-CNN, RAW-CNN..	70
4.2.4	Conclusion.....	71
5.	Research on EEG Signal Algorithm of Motor Imagery	73
5.1	EEG signal recognition based on transfer learning	73

5.1.1	Fine-tuning transfer learning model.....	74
5.1.2	Performance of transfer learning model.....	75
5.2	Multi-branch CNN architecture based on different frequency bands	77
5.2.1	BCI Competition IV Dataset 2b.....	77
5.2.2	Proposed Multi-branch CNN Structure.....	78
5.2.3	Performance of the Proposed Model.....	80
5.3	Multi-branch CNN architecture based on frequency band and time domain	87
5.3.1	BCI Competition IV Data Set 2a	87
5.3.2	Data Augmentation (DA).....	88
5.3.3	Multibranch Convolutional Networks	90
5.3.4	Performance of DA and Multi-branch CNN	92
5.4	Conclusion	98
6.	Hardware implementation	101
6.1	Development Platform	101
6.2	Hardware circuit module design	102
6.3	Experimental evaluation.....	108
6.4	Conclusion	111
7.	Summary and future directions.....	113
	Acknowledgment	117
	Bibliography	119
	Appendix: Publication list.....	133

1. Introduction

1.1. Background and Motivation

The human brain is a highly complex system, and understanding its functions is crucial for addressing neurological disorders and enhancing human-machine interaction. As research into the brain progresses, we continually uncover new theoretical insights. This theoretical research not only aids in understanding how the brain works but also provides a new perspective on human cognition. Firstly, it can help diagnose brain diseases earlier and more accurately, generating detailed diagnostic reports to support the development of targeted treatment plans. Secondly, it fosters the integration of neuroscience with related fields such as computer science, information and communication engineering, neuroengineering, psychology, and clinical medicine. This interdisciplinary approach promotes mutual advancement and paves the way for rapid development and theoretical breakthroughs across these disciplines.

In 2013, the United States launched the BRAIN Initiative, aiming to advance research in neuroscience and related brain diseases through innovative neurotechnologies. By 2020, the initiative's annual research funding reached \$500 million. Similarly, in 2013, the European Union initiated the Human Brain Project, investing €1 billion to integrate computer technology and neuroscience for in-depth brain research. In 2014, Japan announced the Brain/MINDS project, dedicating ¥3 billion to studying brain functions and related diseases. Major breakthroughs in brain science and brain-like artificial intelligence could revolutionize human-machine interaction and significantly aid individuals with movement disorders.

A Brain-Computer Interface (BCI) is a computer-based system designed to capture brain activity signals [1]. By processing these signals through operations such as signal processing, feature extraction, and classification, BCIs convert the decoded information into commands for external devices, enabling human-computer interaction. BCI systems typically utilize methods like electroencephalography (EEG), electrocorticography (ECoG), and near-infrared spectroscopy (NIRS) to record brain signals. Among these methods, EEG stands out for its non-invasiveness, affordability, and high temporal resolution, making it widely used in BCI research.

Currently, research on BCI systems is primarily focused on theoretical and experimental aspects, though some advanced technologies are ready for clinical trials. As an emerging field, BCI systems have vast application prospects in several areas:

- **Neuroscience Research [1]:** BCI systems can capture electrical signals generated by brain activity and decode them using machine learning algorithms to analyze brain signals. As an important interdisciplinary research direction, BCI technology can integrate and improve existing methods, further promoting the development of neuroscience and related disciplines.
- **Clinical Medical Rehabilitation [2]:** BCI technology can assist individuals with motor disabilities in rehabilitation training, helping them regain control over their limbs. By processing and converting brain signals, users' movements or intentions can be translated into commands to control assistive devices like wheelchairs and robotic arms, facilitating interaction between users and the external world and addressing some of their challenges.
- **Entertainment [3]:** BCI technology can be combined with virtual reality to create unprecedented entertainment experiences. It provides an immersive virtual reality environment, offering users a more engaging human-computer interaction.
- **Smart Home [4]:** With the rapid development of communication and IoT technologies, BCI can be widely applied in daily life. BCI-based smart homes can synchronize with human emotions, enhancing quality of life.
- **Transportation and Military [5]:** BCI technology, when integrated with smart driving technologies, enables remote control of vehicles and aircraft. Combined with intelligent robotics, BCI can allow machines to perform hazardous tasks. In the military, BCI can use brainwave information for encryption and more efficient coordinated operations.

In summary, BCI technology is at the forefront of brain science and brain information processing research, offering high scientific value and extensive application prospects. As a multidisciplinary frontier technology, BCI integrates advanced engineering techniques and methods, significantly promoting the development of neuroscience and related fields, and plays a crucial role in understanding brain cognitive mechanisms, diagnosing, and treating brain diseases.

1.2.Challenges in the research

Despite significant performance improvements in brain-computer interface (BCI) systems in recent years, researchers still face numerous challenges in promoting BCI technology. One major issue is that the current performance of BCI systems remains relatively low compared to traditional input methods, making them insufficient for supporting fast and complex communication and control. This performance bottleneck is mainly due to an incomplete understanding of the neural mechanisms underlying information encoding in brain signals. Furthermore, decoding EEG signals requires more efficient computational modeling methods, including advanced signal processing and machine learning techniques. Given these various open challenges, real-world BCI systems are still in their infancy. Designing a practical and effective BCI system primarily involves addressing the following challenges:

1. EEG signals offer a direct window into brain activity, but decoding motor intentions from these signals is complex due to their inherent high dimensionality, variability, and noise.
2. The large amount of EEG data generated during experiments requires efficient processing and classification techniques. This process often relies on manual operations, and data preprocessing (such as filtering and denoising), parameter selection (such as frequency bands and time windows), and feature engineering (including feature extraction and selection) are complex and time-consuming, heavily dependent on expertise in the field.
3. The advent of deep learning techniques, especially convolutional neural networks (CNNs), has led to significant advancements in fields like computer vision and natural language processing. However, CNNs require large sample datasets to support their learning efficacy, and EEG signals are typically considered a small dataset. This often leads to overfitting when deep learning is applied to EEG signal processing.
4. While CNNs have achieved promising results in motor imagery recognition, deploying these models in real-time applications presents computational challenges. Real-time processing is crucial for BCI systems to provide timely feedback and enable seamless interaction. However, deep learning models, particularly CNNs, are computationally intensive and require

powerful processing capabilities, which can be a bottleneck in real-time applications.

This thesis aims to address these challenges by exploring new methods for EEG signal processing, feature extraction, classification, and hardware acceleration.

1.3 Structure of the Thesis

The subsequent chapters of this thesis are organized as follows:

Chapter 2: This chapter provides a comprehensive overview of EEG signal processing and classification, covering pre-processing, feature extraction, and classification techniques.

Chapter 3: Expanding on the previous chapter, this section focuses on hardware acceleration techniques for EEG signal processing. It discusses EEG-based hardware for pre-processing and feature extraction, and CNN optimization techniques for BCI.

Chapter 4: This chapter implements and evaluates EEG signal recognition algorithms for eye state detection, using efficient techniques discussed in previous chapters. It covers data analysis, constructs and assesses MLP and SCNN models and CWT-CNN model, compares their performance, and conducts tests on different datasets.

Chapter 5: This chapter extends eye state detection research to motor imagery analysis using a multi-branch CNN architecture. It evaluates the model's performance on BCI Competition IV datasets (2a and 2b), demonstrating its effectiveness in motor imagery recognition. This sets the groundwork for hardware implementation discussed in the next chapter. In this chapter, a data enhancement algorithm is presented for motor imagery EEG signals, which boosted the classification performance of convolutional neural networks without altering the original data traits.

Chapter 6: This chapter transitions from software-based algorithm development to practical hardware implementation. It detailed the hardware implementation of CNN and validated the hardware acceleration circuit on the PYNQ-Z2 platform.

Chapter 7: The concluding chapter offers a thorough overview of the dissertation, emphasizing its key contributions and future directions.

This dissertation presents a total of eight theses, categorized into three groups:

Thesis Group 1: Review of EEG-based Software and Hardware Solutions.

- **Thesis 1:** A Comparative Study of Traditional and New Methods for EEG.
- **Thesis 2:** Integrated Hardware-Accelerated Techniques for Enhanced EEG Signal Processing.

These will be discussed in Chapter 2 and 3.

Thesis Group 2: Development and Optimization of the EEG Signal Recognition Model.

- **Thesis 3:** Determination of optimal frequency feature indicator.
- **Thesis 4:** Exploration of combining traditional EEG processing solutions with CNN.
- **Thesis 5:** Data enhancement algorithm based on EEG signal.
- **Thesis 6:** Exploration of two branch CNN.

These will be covered in Chapters 4 and 5.

Thesis Group 3: Hardware Acceleration and Optimization.

- **Thesis 7:** Optimization plan based on convolutional hardware implementation.
- **Thesis 8:** Software-configurable lightweight CNN inference accelerator.

These will be presented in Chapter 6.

2. Review of EEG signal processing methods and classification algorithm

As an essential part in BCI, EEG processing can be divided into five key stages according to its application method.

- **Acquisition of EEG signals:** This is the initial phase where involves capturing the brain's electrical activity through various methods and devices for any subsequent analysis and processing.
- **Pre-processing of EEG signals:** In this stage, noise interference is removed from the raw EEG signals. This interference can include artifacts from eye blinks, heartbeats, and muscle movements. Removing these noise signals simplifies the subsequent analysis and processing of the EEG data.
- **Feature extraction:** Features are extracted from the pre-processed EEG signals to distinguish between different EEG patterns. This step also reduces the vector dimension to mitigate computational complexity.
- **Classification:** The extracted features are classified using appropriate classification methods. Selecting a suitable classifier is crucial for achieving accurate classification results.
- **Application of classification results:** The final stage involves converting the classification results to control external devices or to provide judgments.

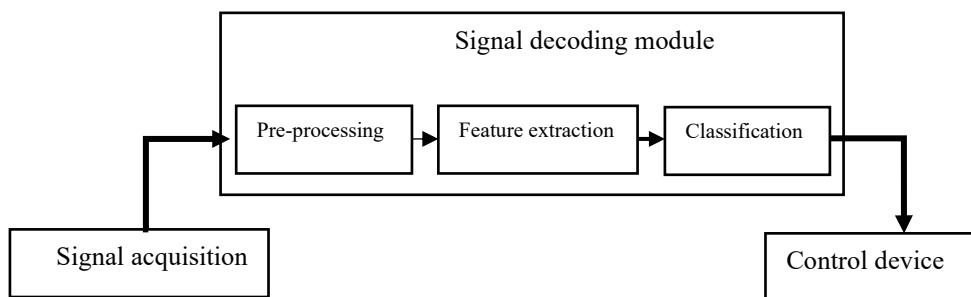


Figure 2.1 The system composition of BCI

This chapter addresses various algorithms used in signal decoding modules and discuss their strengths and weaknesses.

2.1. Pre-processing of EEG signal

Since EEG signals are very weak physiological signals, typically in the microvolt range, they are highly hurt by artifacts such as eye blinks, muscular activity, and electrical grounding disturbances. These interference signals significantly complicate EEG signal processing and elevate the computational demands for further analysis. Several pre-processing techniques are introduced in this section. Table 2.1 offers a comparison of these methodologies.

2.1.1 Principal component analysis (PCA)

PCA is a statistical technique extensively applied for reducing dimensionality. It transforms a set of possibly correlated variables into a smaller number of uncorrelated variables called principal components [6]. These principal components are ordered such that the first few retain most of the variation present in the original variables, thus enabling a more efficient analysis and visualization of high-dimensional data. PCA works by identifying the directions (principal components) along which the variance of the data is maximized. This is achieved through an orthogonal transformation, which ensures that the principal components are orthogonal (perpendicular) to each other. Each principal component is a linear combination of the original variables, with the coefficients of this combination being the eigenvectors of the data's covariance matrix. The corresponding eigenvalues indicate the amount of variance explained by each principal component.

The following are the key steps and formulas involved in PCA:

- **Standardization (if needed):** Standardize the data if the variables have different units or scales:

$$X_{standardized} = \frac{X - \mu}{\sigma} \quad (2.1)$$

where μ is the mean and σ is the standard deviation of the data.

- **Compute the Covariance Matrix:** For a dataset X with n observations and p variables:

$$M = \frac{1}{n-1} X^T X \quad (2.2)$$

where M is the $p \times p$ covariance matrix.

- **Eigen Decomposition:** Compute the eigenvalues and eigenvectors of the covariance matrix M :

$$MV_i = \gamma_i V_i \quad (2.3)$$

where γ_i are the eigenvalues and V_i are the eigenvectors.

- **The Principal Components:** The principal components are the projections of the data onto the eigenvectors:

$$Z = XV \quad (2.4)$$

where Z is the transformed dataset (principal components), and V is the matrix of eigenvectors.

The benefit of PCA is its ability to lower the dimensionality of factor vectors, thereby simplifying signal feature extraction and classification. However, it is important to handle PCA with care, as non-principal components, although they have lower variance, might still contain critical information relevant to the analysis[7]. Thus, a balance must be achieved between dimensionality reduction and the retention of important data features to avoid loss of valuable information.

2.1.2 Independent component analysis (ICA)

ICA is a well-regarded method that applies blind source separation to break down complex datasets into statistically independent components [8]. This technique leverages statistical methods to isolate distinct sub-parts within the data. The theoretical foundation of the ICA algorithm assume that various interference sources, such as eye movement artifacts, ECG, EMG, and electrical grounding noise, originate from statistically independent sources. By segregating these sources using ICA, critical information relevant to EEG signals can be extracted[6].

The following are the key steps and formulas involved in ICA:

Data were recorded for n channels are shown as $x(t) = [x_1(t), \dots, x_n(t)]^T$. Assume it is a merger of multiple independent components $s(t) = [s_1(t), \dots, s_n(t)]^T$, that is,

$$x(t) = As(t) \quad (2.5)$$

where $x(t)$ is the signal collection, A is the inverse matrix of the ICA weights matrix, and $s(t)$ is the estimated source data.

Because the components are independent of each other, the output is approximately equal to $s(t)$ by a matrix W , that is,

$$y(t) = Wx(t) = WAs(t) \quad (2.6)$$

where W indicates the inverse matrix of A , $y(t) = [y_1(t), \dots, y_n(t)]^T$ is the approximate estimate of $s(t)$. Finally, the $s(t)$ in the irrelevant components can be separated.

By rejecting artifacts, ICA helps separated corrupted data segments, thus preserving the integrity of the original dataset, and enhancing the signal-to-noise ratio for subsequent analysis. When dealing with large datasets, ICA generally outperforms PCA in addressing complex signal interferences [9].

2.1.3 Common average reference (CAR)

CAR technique is a spatial filtering method frequently employed in EEG signal processing to enhance the signal-to-noise ratio by eliminating the common components of the EEG signal while retaining the unique signals from specific electrodes. This technique involves computing the average signal from all electrodes and subtracting this average from the signal at each electrode, thereby removing common noise sources like electrical grounding noise. The CAR method is highly effective in reducing noise in EEG signals and is extensively used in various EEG applications [10]. The algorithm for CAR can be represented as follows [11]:

$$V_i^{CAR} = V_i^{ER} - \frac{1}{n} \sum_j^n V_j^{ER} \quad (2.7)$$

Where V_i^{CAR} is the filtered voltage for a particular electrode, V_i^{ER} indicates the difference in voltage between the channel electrode and the reference electrode voltage of a particular electrode. $\frac{1}{n} \sum_j^n V_j^{ER}$ is the average voltage of all electrode.

2.1.4 Common Spatial patterns (CSP)

The CSP initiates from the Common Spatial Subspace Decomposition (CSSD) algorithm, a method of spatial filtering utilized for multichannel EEG data analysis. The main objective of CSP is to find spatial filters that maximize the variance of EEG signals for one class while minimizing it for another class. This is achieved by computing the covariance matrices of EEG signals for different classes and then applying eigenvalue decomposition to find the spatial filters. The resulting filters,

known as common spatial patterns, are used to transform the original EEG signals into a new feature space where the classes are more separable[12].

CSP can be expressed as:

$$W = \frac{\omega^T X_1^T X_1 \omega}{\omega^T X_2^T X_2 \omega} = \frac{\omega^T M_1 \omega}{\omega^T M_2 \omega} \quad (2.8)$$

where W represents the spatial filter matrix, T represents the transpose of the matrix, X is the data matrix of class (rows are training samples, columns are the number of channels), M_1 and M_2 are the covariance matrices of the two classes of EEG signals, respectively.

CSP does not require prior choice of particular frequency bands, making it applicable across a wide range of EEG signal types. It also provides a feature vector that captures task-related signal components while reducing noise and irrelevant components. CSP is sensitive to noise, but EEG signals generally have a low signal-to-noise ratio. CSP usually needs EEG recordings from multiple electrodes to obtain effective spatial filters, limiting its applicability in dataset where only a few electrodes are available. Calculating CSP involves eigenvalue decomposition of covariance matrices, which can be computationally intensive, particularly for high-dimensional EEG data.

2.1.5 Adaptive filter

An adaptive filter is a method capable of autonomously tuning its parameters without prior knowledge of the statistical properties of the input signal and noise. It gradually estimates the necessary statistical characteristics during operation to optimize its parameters for effective filtering. A full adaptive filter comprises four main elements: the signals under consideration, the architecture defining the input or output connections, the parameters, which can be iteratively adjusted to modify the filter's input-output relationship.

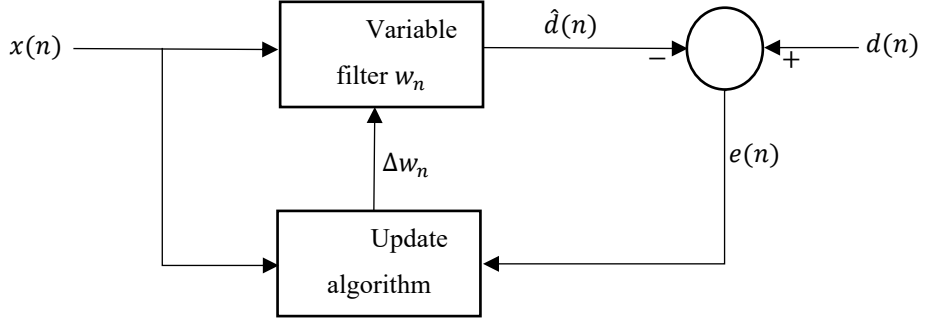


Figure 2.2 Example of adaptive filters

The input is the sum of the desired signal $d(n)$ and the interference noise $v(n)$, that is,

$$x(n) = d(n) + v(n) \quad (2.9)$$

The filter exhibits a finite impulse response architecture, wherein the impulse response aligns with the filter coefficients. These coefficients, denoted as

$$w_n = [w_n(0), w_n(1), \dots, w_n(p)]^T \quad (2.10)$$

The error $e(n)$, referred to as the cost function, represents the variance between the desired signal $d(n)$ and the estimated signal $\hat{d}(n)$, that is,

$$e(n) = d(n) - \hat{d}(n) \quad (2.11)$$

2.1.6 Digital filters

Digital filters usually have four categories, low-pass, high-pass, band-pass, and band-stop filters. While these filters are effective in many cases, they can struggle when EEG and artifact signal frequencies overlap. For instance, notch filters are a go-to for eliminating 50Hz electrical interference. However, they might introduce phase alterations, impacting the EEG signal's phase relationships critical for BCI tasks like P300 and SSVEP. EEGLAB employ zero-phase filtering techniques, such as the 'filtfilt' function in MATLAB, ensuring minimal phase distortion during signal processing.

Table 2.1 Comparison of the pre-processing methods

Approach	Advantages	Disadvantage
ICA	● High calculation efficiency ● Works well for large data sized	● Require more computational for decomposition
PCA	● Reduction of feature dimension	● May lost critical information on non-principal in components
CAR	● The best among all the reference methods	● Needs sufficient head coverage
CSP	● Good effect on EEG data processing based on motor imagination ● does not need to select a specific frequency band in advance.	● sensitive to noise ● needs to be used with multiple electrodes
Adaptive Filtering	● Good for the signals with overlapping spectra	● Needs two signals (including reference signal)
Digital Filter	● Easily remove electrical grounding noise	● Requires EEG signal and artifact occupy distinct frequency bands

2.2.Feature Extraction

Feature extraction plays a pivotal role in the EEG signal processing process by transforming EEG data into a set of relevant features that can be more easily and effectively analyzed[13]. Its purpose is usually noise reduction, dimensionality reduction, highlighting relevant patterns, etc. The extracted features serve as the input for machine learning algorithms and classifiers. Carefully selected and validated features ensure that the model performs well across different task and conditions. Table 2.2 at the end of this subsection provides a comparison of various extraction methods.

2.2.1 Power Spectrum Density (PSD)

PSD is a fundamental technique in the analysis of EEG signals. it is used to quantify the power of various frequency bands within the brain's electrical activity. PSD analysis involves decomposing a signal into its constituent frequencies using techniques like the Fourier Transform. The resulting PSD plot shows the power (usually in decibels) against the frequency (usually in Hertz).

The PSD can be estimated using the Discrete Fourier Transform (DFT). Given a discrete-time signal $x[n]$, the DFT $X[k]$ is calculated as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{N}} \quad (2.12)$$

The PSD $P_X(f)$ is then given by:

$$P_X(f) = \frac{1}{N} |X[k]|^2 \quad (2.13)$$

where N is the number of samples, and $|X[k]|$ is the magnitude of the DFT at frequency bin k .

There are various methods to calculate PSD, with the most common being:

- **Fourier Transform (FT):** The signal is transformed from the time domain to the frequency domain, and the square of the magnitude of the Fourier coefficients gives the power.
- **Welch's Method:** An improvement over the basic FT, Welch's method involves dividing the signal into overlapping segments, applying a windowing function, computing the periodogram of each segment, and then averaging these periodograms.
- **Multitaper Method:** Uses multiple orthogonal window functions to provide a more accurate and stable estimate of the PSD.

PSD is primarily used to identify frequency bands and detect abnormalities. The advantages of PSD is straightforward to compute and interpret. assumes that the signal's statistical properties do not change over time, which may not always be true for EEG signals.

2.2.2 Principal Component Analysis

PCA can be used both in data preprocessing and feature extraction from EEG datasets. The main objective of PCA is to reduce the number of dimensions in a dataset while preserving most of the data's variability. When applied to EEG data, PCA helps isolate valuable signals from time-series data while eliminating artifacts such as noise and irrelevant information. One significant application of PCA in EEG analysis is within BCI P300 speller systems [14]. These systems assist severely disabled individuals by enabling communication through a P300 speller. PCA was used to weaken unnecessary features and improve classification accuracy [14]. However, it is important to note that PCA is not suitable for processing linear and continuous data or complicated sets of features[9].

2.2.3 Independent Component Analysis

ICA, similar to Principal PCA, serves both as a preprocessing and feature extraction method. The primary goal of ICA in preprocessing is to remove artifacts and noise from EEG signals. In the context of feature extraction, the goal of ICA is to identify and isolate the underlying sources or features within the EEG signals that are relevant to the specific analysis or classification task. Paper [15] assessed and enhanced ICA-based correction for tasks involving visual search in images and sentence learning with free eye movements. The improved ICA methodology successfully removed nearly all artifacts, including spike potentials and related spectral broadband noise from both visual tasks, while causing minimal distortion to neural signals.

2.2.4 Auto-Regressive (AR)

AR analysis is a prevalent method in EEG signal analysis that required representing the signal as a linear combination of its previous values. The AR model frames time domain data as a linear prediction problem, where the current point is estimated based on a linear association with prior data points [16].

The AR model of order p is defined by the following formula:

$$X_t = c + \sum_{i=1}^p \phi_i X_{t-i} + \epsilon_t \quad (2.14)$$

Where: X_t is the current value of the time-series, c is a constant term, ϕ_i are the parameters of the model (coefficients), X_{t-i} are the past values of the time-series, ϵ_t is white noise error term.

AR models can be classified into two types: adaptive and non-adaptive models [9]. Adaptive models update their parameters continuously as new data is acquired, while non-adaptive models use fixed parameters throughout the analysis. Constructing an AR model can be computationally demanding and requires meticulous parameter tuning. Despite these challenges, the AR method remains a valuable and powerful tool in EEG signal analysis due to its effectiveness in capturing the dynamics of non-stationary signals.

2.2.5 Wavelet transform (WT)

WT is a method used for time-frequency analysis, expanding upon the Short-Time Fourier Transform's concept of signal localization. It is adept at extracting valuable information from diverse continuous signals like speech and image data. WT dynamically adjusts to the signal's time-frequency analysis needs, different with fixed resolution Fast Fourier Transform (FFT). The signal undergoes decomposition into different scales and is refined through telescopic translation, enhancing time resolution at higher frequencies and frequency resolution at lower frequencies. During wavelet transform decomposition, only the low-frequency segment of the signal is further decomposed, while the high-frequency segment is no longer decomposed. Wavelet Packet Transform (WPT) offers enhanced frequency resolution in the high-frequency segment compared to standard wavelet decomposition (WD) [17]. However, selecting the optimal wavelet base function presents a challenge due to the absence of a universally applicable function. Nevertheless, WT has proven effective in various EEG studies, including cognitive task analysis, BCI development, and driver fatigue detection, for extracting features from EEG signals.

Table 2.2 Comparison of extraction approaches

Approach	Advantages	Disadvantages
PSD	● Feature stability ● Applicable for stationary signals	● Not suitable for unstable signals ● Unable to analyze time-domain signals
PCA	● Lossless dimension reduction	● Unable to process complicated set of data
ICA	● fast calculation speed ● high efficiency	● Require more computational for decomposition
AR	● Require short duration of data ● Reduce spectra loss problems and give better frequency resolution	● Not available for stationary signals
WT	● Works well for nonstationary signals ● Can analyze signal in time and frequency domain	● Lacking of specific methodology to apply to pervasive noise
WPT	● Works well to analyze the nonstationary signals ● provide precise frequency resolution	● Increase computational time

2.3. Classification

Classification is a fundamental task in data processing and machine learning. It trains network to assign data points to predefined categories or classes. Another purpose is prediction. It applies the training model to new data and classifies it as one of the predefined categories. Some common Classification Algorithms were introduced in this section, such as K-Nearest Neighbor (k-NN), linear discriminant analysis (LDA), support vector machine (SVM), and naive Bayes (NB), Artificial Neural Network (ANN)[18] and Deep Learning (DL)[19]. A comparative analysis of these classification methods is presented in Table 2.3 at the end of this subsection.

2.3.1 Artificial Neural Network (ANN)

From the inception in the 1980s, ANNs have significantly transformed the artificial intelligence domain. Modeled after human neural networks, ANNs can be trained to address various classification and regression issues by forming multiple connections between neurons. These networks are non-linear classifiers composed of numerous neurons, making them ideal for processing vast amounts of data simultaneously. One of the most commonly used ANN architectures is the Multi-Layer Perceptron Neural Network (MLPNN), which includes input layers, hidden layers, and output layers. The MLPNN framework offers several benefits, such as fast processing, straightforward implementation, and minimal dataset requirements. However, setting up an MLPNN can be challenging because its performance heavily relies on the number of neurons in the hidden layer [9].

ANN provide solutions for complex problems that traditional methods in biomedical signal processing cannot easily address. By training on historical data, ANNs can study to identify method in EEG without needing to explicitly account for the full feature of diseases[20]. For instance, paper [21] utilized a trained ANN for tuberculosis diagnosis. It reaches sensitivity and specificity rates of 100% and 72%, respectively. Paper [22] employed an ANN-based MS-ROM/I-FAST system for the differential diagnosis of children with autism from their EEG signal. They reach a positive result by a few minutes raw EEG signal. Hence, ANNs are valuable tools for classifying and recognizing EEG signals and hold significant potential for diagnosing various diseases.

2.3.2 Deep Learning

DL is a subdomain of machine learning and an advanced form of ANNs. It is characterized by numerous layers and parameters, different with conventional limited hidden layers ANNs. The main objective of DL is to study, and abstract complex patterns and connection from big data. EEG classification tasks utilizing DL span a range of applications, including emotion recognition, motor imagery recognition, psychological load detection, epilepsy detection, event-related potential detection, and sleep scoring. The research of EEG faces several challenges, particularly in the realm of foundational theories. Although there is a substantial volume of research, groundbreaking algorithms have yet to emerge. Instead, much of the recent work has centered on optimizing and making incremental improvements to existing methods. Neural networks, despite their promise in processing and recognizing EEG signals, still operate largely as “black boxes,” making their internal workings difficult to fully understand.

Popular DL models used in EEG signal processing include CNNs, recurrent neural networks (RNN), deep belief networks, stacked autoencoders, and MLPs[23]. A CNN comprises multiple layers, including the input, convolution, pooling, fully connected, and output layers. The convolutional layer is tasked with feature extraction, identifying relevant patterns in the data. The pooling layer then reduces the number of training parameters and helps mitigate model overfitting by down sampling the feature maps. In the fully connected layer, all neurons are linked to the last pooling layer, integrating the extracted features into a cohesive analysis. Finally, the output layer contains many neurons to deliver the classification result.

The RNN is designed specifically for sequential data. In RNN, the neurons are affected not only by the current input but also by the output from the previous time step, allowing the network to maintain a form of memory. This capability makes RNNs particularly well-suited for tasks involving time-series data, such as EEG signal classification. A prominent architecture within the RNN family is the Long Short-Term Memory (LSTM) network. LSTMs are designed to address the limitations of traditional RNNs, particularly their inability to learn long-term dependencies due to issues like vanishing gradients. LSTMs feature a unique memory module with three gates: the input gate, output gate, and forget gate. These gates regulate the flow of information into, out of, and withdrawal from the cell, respectively. LSTM retains important information over extended time periods and discard irrelevant data. The

capability of LSTM networks can be fine-tuned by modifying the number of hidden layers and neurons within those layers. This customization allows for optimization based on the specific characteristics of the data and the task.

In recent research, the performance of various deep learning-based EEG signal classification models was evaluated. And some of them reveal that CNNs and RNNs outperformed other methods in terms of classification accuracy[23].

2.3.3 K-Nearest Neighbors (KNN)

KNN is introduced by Cover and Hart in 1968 and recognized as one of the simplest and most straightforward machine learning algorithms. Over time, KNN has found applications in various fields, including face recognition, text recognition, and medical image processing[24, 25]. The fundamental principle of KNN is to classify a given data point based on the predominant class among its k-nearest neighbors within a feature space. The algorithm begins with establishing a training sample set where all sample categories are pre-identified. To classify a new data point, KNN computes the similarity between the new sample and each training sample, selects the k samples with the highest similarity, and assigns the new sample to the majority class of these k-nearest neighbors.

KNN is an algorithm that determines categories based on “distance”. The distance calculation formula usually uses Euclidean distance and Manhattan distance. The calculation formula for Euclidean distance is:

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.15)$$

The calculation formula of Manhattan distance is:

$$d(x, y) = |x_2 - x_1| + |y_2 - y_1| \quad (2.16)$$

KNN falls under the category of instance-based learning, or lazy learning. In this approach, the algorithm does not explicitly learn a model; instead, it retains the training dataset with its classification and feature values and processes new samples when they are received.

One notable drawback of the KNN algorithm is its high computational complexity, which scales with the size of the training dataset. This characteristic makes KNN more suitable for smaller datasets, as the computational demands can become prohibitive for large-scale data.

2.3.4 Linear Discriminant Analysis (LDA)

LDA is introduced by Fisher in 1936. As a linear technique, the main goal of LDA is to find a projection direction that maximizes the separation between different classes while minimizing the spread within each class. This is achieved by identifying a linear transformation that maximizes the ratio of the between-class scatter to the within-class scatter. The same projection method is then used to classify new samples, determining their class type based on their position along the straight line.

This technique is a classical approach to dimensionality reduction, aiming to project the original x -dimensional predictor variables onto a single one-dimensional line. The goal is for the discriminant scores of each class to follow a normal distribution, while maximizing the separation between class means. In practice, the degree of overlap between the score distributions serves as a key measure of the method's effectiveness. These discriminant scores are computed using a discriminant function, which takes the following mathematical form:

$$D = w_1Z_1 + w_2Z_2 + w_3Z_3 + \dots + w_xZ_x \quad (2.17)$$

LDA is valued for its low computational complexity and simplicity. However, it has limitations when dealing with data that do not follow Gaussian distributions or have complex structures not captured by mean values alone[26]. While traditionally used for binary classification, LDA can be adapted for multi-class problems through specific strategies [1] [27]. In EEG analysis, LDA is a useful tool for distinguishing between different brain states. Researchers must carefully consider the nature of EEG data to apply LDA effectively and obtain meaningful insights.

2.3.5 Support Vector Machine (SVM)

SVM are popular linear classifiers used for binary classification tasks through supervised learning. The core idea of SVM is to find an optimal decision boundary, or hyperplane, that effectively separates different classes of data in the feature space.

SVM uses kernel function to realize the mapping from low-dimensional space to high-dimensional space. Commonly used kernel functions include RBF kernel, linear kernel, Laplace kernel.

SVMs have many significant advantages as a classification method. First, SVMs guarantee the uniqueness of the global optimal solution and can be efficiently obtained. Second, SVMs can handle nonlinear classification boundaries by using kernel

functions without excessive additional computational overhead. In addition, SVMs have good generalization and robustness, and often outperform other mainstream classification algorithms in many tasks, especially in small-scale datasets.

However, an important disadvantage of SVMs is that the computational complexity of their training process is mainly affected by the number of training samples rather than the feature dimension. When the sample size is large, SVM are often unable to cope with it and may fail due to memory or computational time limitations. In this case, optimization algorithms specially designed and accelerated for large-scale SVM training problems must be used.

SVMs are categorized into three primary types based on their construction models[28]:

- **Linear Separable SVM:** is used when the data can be perfectly separated by a straight line (in two dimensions) or a hyperplane (in higher dimensions). The goal is to find the hyperplane that maximizes the margin between the two classes.
- **Linear SVM:** applicable when the data is not perfectly separable, but the classes can still be distinguished by a linear boundary. In such cases, SVM introduces slack variables to allow some misclassifications, aiming to maximize the margin while minimizing classification errors.
- **Nonlinear SVM:** is used when the data cannot be separated by a linear boundary. Nonlinear SVM employs kernel functions to transform the data into a higher-dimensional space where a linear boundary can be applied. Common kernel functions include polynomial, radial basis function (RBF), and sigmoid kernels.

By utilizing these different types of SVM models, it is possible to handle a wide range of classification problems with varying data structures and complexities. Paper [29] utilized SVM classification to evaluate driving fatigue levels. The study identified a critical threshold that differentiates between mild and severe fatigue. It shows that driving for at least two continuous hours significantly affects fatigue severity. The SVM algorithm showed promising accuracy, particularly in severe fatigue cases. Paper [30] improved method for analyzing epilepsy EEG data, using PCA for feature extraction. The extracted features were then classified with several algorithms, including Naive Bayes (NB), KNN, LDA, and least squares SVM (LS-SVM). The

findings demonstrated that LS-SVM achieved an impressive classification accuracy of up to 100%, outperforming existing epilepsy EEG classification methods by 7.10%. It shows that SVM proves to be a powerful tool for effective classification and detection.

2.3.6 Naive Bayes (NB)

The NB classifier, rooted in Bayes' theorem, is a functional classification algorithm widely used across diverse fields. It shows comparable performance to other established classifiers [31-33]. Its core concept involves computing the probability of each category given a specific item and assigning the item to the category with the highest probability [34]. NB assumes independence and lack of correlation among features in the samples, boasting remarkable speed, efficiency, and simplicity in algorithmic structure, particularly suitable for high-dimensional data handling [35]. Enhanced versions like tree-enhanced and network-enhanced NB have been proposed to boost performance and classification accuracy. Researchers can explore its potential further by integrating novel enhancements and adaptations, thus augmenting its performance and broadening its applications in EEG research.

2.3.7 Decision Tree (DT)

DT are a type of supervised learning algorithm used for classification and regression tasks. They operate by recursively dividing the input space into smaller regions, based on the features of the data, to create a tree-like structure. Each internal node in the tree represents a decision based on a feature, and each leaf node represents the outcome or prediction. [36]. In the late 1970s and early 1980s, J. Ross Quinlan introduced the ID3 (Iterative Dichotomiser 3) algorithm for generating decision trees. Later, he proposed the C4.5 algorithm as an enhanced version of ID3, optimizing its performance. C4.5 has since become a foundational method for developing new supervised learning techniques. The J48 algorithm is a derivative of C4.5, created to translate the algorithm into Java from its original C implementation. This algorithm prioritizes the best locally evaluated step without considering if it will lead to the overall best solution, breaking down problems into subproblems through the creation of subtrees from the root to the leaves [37, 38]. DT algorithm prioritizes the best locally evaluated step without considering if it will lead to the overall best solution, breaking down problems into subproblems through the creation of subtrees from the root to the leaves[39]. In essence, classification algorithms seek to find models that achieve the highest accuracy or lowest error rate when applied to a test dataset. Decision trees are popular due to their simplicity, interpretability, and ability to handle both numerical

and categorical data. They are useful for understanding the decision-making process and identifying important features in a dataset.

The selection of an appropriate classifier algorithm is crucial in achieving the best classification performance, as no single algorithm possesses absolute advantages across all scenarios. In the context of EEG analysis, the choice of classifier plays a pivotal role in accurately characterizing and classifying EEG signals. Researchers should carefully assess the specific characteristics of their EEG dataset, such as the dimensionality of the extracted features and the trade-off between bias and variance. As the dimensionality of the feature space increases, the amount of training data required grows exponentially. Consequently, the classifier tends to exhibit high bias and low variance. To address this challenge, it is essential to comprehensively explore and leverage various classifier algorithms, as this approach represents the evolving trend in classifier development. The comparison of the classification methods is shown in Table 2.3.

Table 2.3 Comparison of the classification methods

Approach	Advantages	Disadvantages
ANN and DL	● High accuracy ● flexible structure	● Performance depends on the number of neurons in the hidden layer
K-NN	● Easy to understand, ● simple to implement	● Sensitive to irrelevant and redundant features ● Requires large storage space
LDA	● Easy to use ● Low computational complexity	● Requires a linear model
SVM	● Performance is better compared to other linear classifiers	● Low computational complexity
NB	● Easy to understand	● Independent variables
DT	● High interpretability ● Ease of use	● Limited expressiveness ● Easy overfitting

The results presented here provide the basis for the first thesis point, which is belong to the thesis group 1 and it has been published to Carpathian Journal of Electronic & Computer Engineering with title “A Review of Processing Methods and Classification Algorithm for EEG Signal[J2].”

Thesis 1: I bring together the latest advancements in EEG signal processing and classification, offering a comprehensive and insightful review of both traditional and

state-of-the-art techniques. Additionally, my work highlights the growing significance of deep learning in the field of EEG analysis and emphasizes the need for further exploration and development of deep learning algorithms.

3. Review of Hardware Acceleration Techniques for EEG Signals

As the importance of EEG spreads over many fields, the demand for fast data processing solutions is growing. Researchers aim to revolutionize this field by substantially enhancing computational efficiency and classification precision. While Chapter 2 covered numerous EEG-based data processing technologies, this chapter focuses on the hardware solutions for implementing these technologies. Most software-based methods have high computational demands, but specific application scenarios with constraints on size, security, energy consumption, and precision require customized hardware implementations. There are several commonly used hardware solutions: FPGA-Based, GPU-Based, ASIC-Based, Hybrid Approaches. Table 3.1 provides a detailed comparison of these hardware solutions, highlighting their strengths, weaknesses, and application contexts.

Table 3.1 Comparison of hardware solutions

Solution	Advantages	Disadvantages	Best Used For
FPGA	● Parallel processing ● Customizability, low latency	● Complex programming ● High initial cost	● Real-time processing ● Low-latency tasks
GPU	● High throughput ● Optimized libraries ● scalability	● High power consumption ● Requires cooling	● High-dimensional data processing ● Deep learning
ASIC	● High efficiency ● Low power consumption ● Compact size	● High development cost ● Lack of flexibility	● Portable/wearable devices ● Low-power applications
Hybrid	● Combines strengths of multiple solutions	● Increased complexity in design and integration	● Comprehensive systems requiring both speed and flexibility

3.1 EEG-BASED HARDWARE SOLUTIONS

The conversion of raw EEG signals into meaningful information is a complex and multi-step process, requiring advanced techniques. We will explore hardware

solutions designed of pre-processing, feature extraction, and the different types of features in section 2.1.

3.1.1 Pre-processing

Before studying the valuable information contained in EEG signals, it is essential to pre-process the data to reduce the significant noise and artifacts often present. Artifacts in EEG data usually are both physiological and non-physiological.

Although some datasets come with pre-cleaned EEG data, implementing pre-processing steps in hardware allows for a more thorough and practical assessment of the resources needed for EEG-based solutions. Physiological artifacts are linked to activities like head movements (EMG), blinking (EOG), muscle contractions, heart activity (ECG), and respiration (RS) [40, 41]. Non-physiological artifacts stem from external factors such as electrode malfunctions (Ep), power sources (AC), ventilation, magnetic fields, and cable movements (Cable) [42, 43]. Table 3.1 illustrates that the EEG signal spectrum and various artifacts ranges from 0 to 100Hz. The definition of EEG frequency bands can vary slightly across different literature. The frequency bands of Mu and Alpha waves overlap, though they originate from different brain regions. Alpha waves typically appear at the back of the head when the brain is relaxed or the eyes are closed, while Mu waves are observed in the motor cortex when the person is at rest. Mu waves decrease or disappear during actual or imagined movement. Paper [44, 45] introduces the method of using ICA to distinguish Alpha waves and Mu waves.

Table 3.1 EEG frequency bands and their artifacts

EEG signal band	Frequency (Hz)
Alpha	8 - 13
Beta	12 - 30
Delta	0.5 - 3.5
Theta	4 - 7
Mu	8 - 12
Gamma	26 - 100
Artifacts	Frequency (Hz)
ECG	0.5 - 40
EOG	- 17
EMG	0.1 - 2.5 / 20 - 100
RS	0.1 - 2.5
Ep	0.1 - 35
Cable	2 - 15
AC	50

Using Digital filter is a relatively easy to achieve noise reduction method. Paper [46] employed a bandpass filter covering 8 to 45 Hz to remove low-frequency artifacts, though this inadvertently diminished the Theta and Delta brainwaves. Paper [47] and [48] utilized a lower cutoff frequency of 2 Hz, which allowed slow eye movement artifacts to persist alongside the EEG recordings. [49] involved an 81-order digital filter, which provided improved graphical analysis for Beta and Mu brainwaves and was successfully implemented on the DE0-Nano-Soc development board. These methods primarily used direct bandpass filtering without addressing the complex interference scenarios caused by the various artifacts mentioned earlier. Paper [50] provided three crucial steps for digital filter: down sampling, bandpass filtering, and incorporating a Mean Average Reference (MAR) filter. Down sampling at twice the rate, without altering the frequency content, reduces the power spectral amplitude by half, thereby enhancing the clarity of the signal. The bandpass filter operates within the 2 Hz to 60 Hz range, ensuring the EEG signals stay within the desired spectrum. The MAR filter works by re-referencing electrode signals relative to their collective average, effectively eliminating artifacts associated with the physical reference electrode [51].

In contrast, paper [52] took a more complex approach, initiating an adaptation strategy utilizing ICA based on kurtosis. This method aims to extract artifacts that occur simultaneously with EEG signals within the same frequency band. However, the application of this technique is currently limited to offline settings, and its feasibility for hardware implementation remains uncertain. In [53], fast ICA was employed for detecting epileptic seizures from multi-channel signals. It aids in artifact removal and reduces energy dissipation. Eigenvalue decomposition (EVD) in conjunction with the Jacobwe algorithm resulted in a 77.2% reduction in area. The practical performance of this architecture was realized on an FPGA and validated using appropriate databases. These blind source separation methods represent a common approach for solving the challenges of EEG-based data in hardware-driven procedures [54, 55]. However, the effectiveness of these techniques is directly associated with the number of independent components that can be collected, which in turn depends on the quantity of data sources or electrodes. As a result, these methods may encounter limitations in scenarios where there are only a few channels. This understanding has led to the pursuit of more cost-effective alternatives integrating the ICA engine [56, 57]. For example, matched filters

can be combined with threshold cross-checks to assess performance based on the number of electrodes.

3.1.2 Feature Extraction

EEG data contains a wealth of information. To analysis this information effectively, it must be converted into meaningful features. This section explores the techniques of feature extraction that transform EEG signals into quantifiable characteristics. We explore five categories of features in this domain: frequency-based, time-based, time-frequency, spatial, and raw signals. A summary of the reviewed articles is presented in Table 2 at the end of this section.

3.1.2.1 Frequency-based Features

Frequency-based features analyze the rhythmic patterns of brain activity, offering valuable insights into brain behavior. It has demonstrated the correlation between spectral components and brainwave rhythms[58, 59]. Frequency-based features are crucial for both software and hardware platforms.

In the realm of software platforms, the implementation of these methods varies significantly and typically operates under the supposition of unlimited computational resources. Only a few methods with low computational complexity can be directly implemented using hardware, including PSD, Fourier Transform, Discrete Cosine Transform (DCT), Singular Spectrum Analysis (SSA), and combinations of filter banks. For example, the hardware implementation of FFT does not consume much resources and can speed up the calculation of PSD. On the other hand, the implementation of hardware solution, higher-order spectral (HOS) analysis, Linear Frequency Cepstral Coefficients (LFCC), Mel Frequency Cepstral Coefficients (MFCC), the Burg method, and Auto-Regressive Reflection Coefficients (ARRC), requires more hardware resources and research and design.

Cross-Frequency Coupling (CFC) functions exploit synchronized variations across different frequency bands [60, 61]. It typically appears as connectivity graphs or indices. Connectivity graphs may significantly reduce EEG data rates and memory requirements. Despite this, CFC algorithm itself requires a lot of computing resources. Such as the requirement for Coordinate Rotation Digital Computer (CORDIC) processors to compute cross-frequency Hilbert transforms [62].

PSD has emerged as one of the most widely utilized methods in the realm of EEG processing. In paper [50] and [47] employed an approximations PSD method. They used only one power value per channel calculated based on the average activity within the beta band (12 to 30 Hz). This approach bypasses the need for FFT accelerators. Which makes it more compatible with hardware implementations. However, this method results in the loss of detailed information within the frequency bands. Conversely, [52] utilizes an FFT with 128 frequency bins, which provides sufficient resolution to capture fluctuations both among the various EEG wavebands.

3.1.2.2 Time-based Features

Although the research methods based on time characteristics do not dominate EEG hardware solutions, EEG signals also contain time domain characteristics, such as signal amplitude and dynamic changes over time. Therefore, to comprehensively analyze EEG signals, the hardware implementation of time characteristics is essential.

Higher Order Crossings (HOC) is a widely adopted approach. It involves monitoring zero crossings and their successive differences for analyzing temporal dynamics of EEG signals. Higuchi's Fractal Dimension (HFD) is particularly well-regarded for its capability to capture the non-linear characteristics of data. It is highly suitable for multi-events. In addition, there are several other time-based feature extraction methods that have proven effective. Such as Hjorth parameters, Grassberger-Procaccia (GP), Hybrid Adaptive Filtering (HAF), Hurst exponent, Local Binary Patterns (LBP), and Event-Related Late Positive Potentials (LPP). Furthermore, advanced predictive techniques such as Neural Time Series Prediction Processing (NTSPP) and the use of Wasserstein distance metrics to compare probability distributions are also employed, enhancing the depth of EEG signal analysis and understanding.

Different with frequency-based solutions, time domain approaches offer the advantage of relatively lower hardware resource demands. Nonetheless, it is important to recognize that techniques involving non-linear transformations, such as HFD or Hjorth parameters, tend to increase the resource consumption of hardware design.

Paper [48] introduces the HOC and skewness (SK) information to form the feature vector. For the approximate SK indicator (ASKI), it significantly reduces the gate count by 86 times. For HOC, they adopt a simple method using a comparator and a 3-bit SR flip-flop to monitor the previous sample value. Detecting mark amplitude

between consecutive samples triggers a simple counter to tally zero crossings. These metrics are then standardized and as input for neural network. The “activity” parameter of Hjorth parameters requires computing variance. The “mobility” parameter involves taking the square root of the ratio of successive variances. The “complexity” parameter requires division operations using first-order derivatives [63]. Despite the additional computational complexity these parameters introduce, they provide valuable insights into the EEG signals, enhancing the overall analysis and interpretation of the data.

Sample Entropy (SE) stands out as another time-based method [64] that proves to be relatively hardware-friendly as its simplicity [64]. SE serves as a measure of time series complexity, defined as the logarithm of a ratio. In [65] an SE architecture was proposed that runs ten times faster on FPGAs compared to CPUs. Additionally, methods with minimal hardware resource utilization and power consumption, such as statistical mean and other time-based techniques like Root Mean Square (RMS), Variance (Var), and Standard Deviation (SD), have been observed in the articles surveyed. These methods, both on software and hardware platforms, exhibit an average accuracy of no more than 60%. Consequently, they are not good choices for EEG processing.

3.1.2.3 Time-Frequency Features

Time-frequency features are a class of signal processing techniques that combine information from both the time and frequency domains to analyze non-stationary signals. These features aim to capture how the frequency content of a signal changes over time, providing insights into dynamic processes occurring in the signal. WT and its various forms is a widely adopted approach in Time-frequency domain. It includes Discrete Wavelet Transform (DWT), Continuous Wavelet Transform (CWT), and Adaptive Tunable Q Wavelet Transform, and so on. Other algorithms have also attracted attention, such as Short-Time Fourier Transform (STFT) and Hilbert-Huang Transform (HHT). HHT decomposes a signal into Intrinsic Mode Functions (IMFs) using Empirical Mode Decomposition (EMD). It then analyzes the instantaneous frequency of each IMF to capture time-varying frequency components. It illustrates robustness in investigating EEG signals characteristics such as non-stationary and non-linear time series [66]. But the process of extracting IMFs in HHT includes iterative computations, which are challenging to parallelize, thus presenting a difficulty for achieving real-time hardware implementations.

WT and STFT are both viable options for hardware implementation. [67] introduced an STFT design implemented on the Xilinx Zedboard. 20 frequency feature components are extracted and categorized into five different EEG wave frequency bands for classification. [68] implemented both a deep learning model and an STFT module on FPGA to enhance seizure detection. [69] presented a generalized spectrogram technique where STFT was employed on signals recorded from six electrodes. These signals were then combined using asymmetric indices. [70] integrated DWT with LDA for two-class classification [71] introduced a implementation on the Basys 3 Artix-7 FPGA Board. for Morlet CWT for EEG signals.

STFT divides the signal into smaller windows for FFT processing individually, whereas WT operates by applying filter banks across the whole signal. While both techniques offer comparable outcomes, Wavelet Transform excels in handling non-periodic and rapidly changing features. Conversely, STFT is preferable for real-time processing due to its utilization of shorter time windows.

3.1.2.4 *Spatial Features*

In contrast to other based features, spatial features rely on the presence of time-based, frequency-based, or time-frequency EEG data to compute. One of the technologies called CSP is widely used.

In a binary classification task, the objective of CSP is to identify spatial filters that maximize the variance of EEG signals for one class while minimizing it for the other class. This process generates spatial filters that accentuate discriminative features specific to each class, thereby enhancing classification accuracy:

The application of the CSP formula involves computing the spatial filter w through the simultaneous diagonalization of the sample covariance matrices derived from both classes.

$$J(w) = \frac{w^T \overline{C}_1 w}{w^T \overline{C}_2 w} \quad (3.1)$$

where T represents transpose. $\overline{C}_1$ and $\overline{C}_2$ denotes the average covariance matrix of two different conditions or classes, respectively, which are defined as follows:

$$\overline{C}_k = \frac{1}{N} \sum_{n=1}^{N_k} \frac{D_{(k,n)} D_{(k,n)}^T}{\text{Trace}(D_{(k,n)} D_{(k,n)}^T)}, k = 1, 2 \quad (3.2)$$

where $\text{Trace}(D_{(k,n)}D_{(k,n)}^T)$ represent the solution of the matrix trace. N_k is the number of samples for k th class, that is, the number of single trial data. $D_{(k,n)}$ is the n th trial data of the k th class. The CSP algorithm frames this as a generalized eigenvalue problem:

$$\overline{C}_2^{-1}\overline{C}_1w = \lambda w \quad (3.3)$$

The conventional CSP feature extraction method derives features by computing the logarithm of variances from spatially filtered signals, details as follows:

$$f_p = \log \left\{ \frac{\text{var}(Z_p)}{\sum_{i=1}^{2m} \text{var}(Z_i)} \right\}, p = 1, 2, \dots, 2m \quad (3.4)$$

where the first and last m rows of the EEG signals are usually taken as the final spatial filter. Here Z is given by $Z = w^T D$.

In paper [72], CSP was utilized as a filter on an Altera Stratix-IV FPGA platform. They validated using BCI competition dataset for motor imagery tasks. In [73], CSP and Mahalanobis Distance (MD) was employed. The entirety of the signal processing task took place on software embedded within a Nios II processor on a Stratix IV FPGA. Although this method effectively reduced FPGA resource consumption, it needs more 0.5 seconds time delay. [73] is an improvement on [72]. The authors opted to accelerate some critical tasks in hardware. Although most of the processing remained in software, it successfully reduced the delay to 0.399 seconds. In [74], the authors employed Scalp-Coupled Common Spatial Spectral Patterns (SCSSP) as the feature extraction method and Support Vector Machine (SVM) as the classifier on FPGA.

In the preceding section, the utilization of asymmetric indices as part of spatial features was introduced. These coefficients are designed to unveil asymmetrical and coordinated neural activity by examining the interplay between the brain's two hemispheres [75]. This approach relies on computing asymmetric indices, typically derived from electrode pairs. It can be seamlessly integrated with frequency, time, or time-frequency features. In [47] and [50], they used asymmetric indices alongside channel-based PSD averages. These implementations, utilizing eight channels, compute asymmetric indices based on the PSD averages of electrode pairs.

Another space-based method is Directed Transfer Function (DTF). DTF features act as causal metrics utilized for discerning brain connectivity patterns.[76]. DTF

quantifies the causal impact of one EEG channel on another at particular frequencies. Implementing DTF in hardware is theoretically feasible, but it presents more challenges compared to asymmetric indices. Because of the requirement to compute transfer matrices of multivariate autoregressive models, execute square root operations, and perform division operations.

3.1.2.5 Raw Signals

The significance of retaining raw data and its potential for yielding distinctive capacity is underscored in this section. Due to the emergence and development of deep neural networks (DNN), the original signal can be directly applied to the classifier. It shows good robustness and accuracy. Presently, no hardware-based implementations have emerged following this paradigm. Because the challenge of addressing the complexity of DNNs in the absence of specific features[77, 78]. Real-time hardware solutions for EEG analysis usually favor shallow neural networks for their compact and space-efficient designs. These networks cannot directly process raw EEG data. However, future innovations could introduce new paradigms. For instance, near/super-threshold techniques in Fully Depleted Silicon on Insulator (FDSOI) technology [79] or adaptive body biasing in advanced semiconductor technologies may enable more advanced processing capabilities [80]. These methods have the potential to enable hardware implementations of DNNs that can directly process raw EEG signals in wearable classifiers.

Another method is to use CNN as feature extraction. In [81], overlapping filters and Differential Entropy (DE) and PSD analysis are employed. These methods help derive suitable kernels that guide the CNN in automatically extracting features from the raw EEG signals. The main advantage of this approach is its ability to implicitly merge spectral and entropy information. This integration is particularly beneficial for learning and classification tasks involving raw EEG data.

Table 3.2 Feature Extraction Methods

Feature domain	Authors	Algorithm	Work Description
Frequency-based	[50]	PSD	A power spectrum approximation per channel
	[47]		Hardware friendly, no FFT accelerator required
	[52]	PSD	Easily lose band information Dataset: Own FEXD

		Hardware: Digilent Atlys with Spartan-6 A FFT Accelerator with 128 frequency bins Suitable for capturing fluctuations within and between brainwave frequency bands
	[62] CFC	Use connection diagrams to represent EEG Reduces the need to store and compute EEG CORDIC is required to calculate the Hilbert transform of each frequency band
Time-based	[48] HOC, SK	Hardware: TSMC 0.18um 1P6M CMOS chip Databases: DEAP and SEED Use ASKI to reduce gate counts, 3-bit SR triggers
	[63] Hjorth parameters	Hardware: Zynq-7000 FPGA Dataset: Bonn University EEG database Classifier: KNN
	[64] SE	For statistics, complexity and irregularity system
	[65]	More suitable for short and noisy datasets Relatively hardware-friendly
Time-Frequency	[67] STFT	Hardware: Xilinx Zedboard Extracted 20 features from 128 channels
	[68] STFT	Used for epilepsy detection Only simulation, not applied to real FPGA chips
	[69] STFT	Hardware: ADVANTEST V93000 PS1600 chip Dataset: DEAP STFT features mixed with asymmetric indices
	[70] DWT	Hardware: Basys 3 Artix-7 FPGA Board Classifier: LDA Suitable for developing low-cost, marketable products
	[71] CWT	Hardware: Spartan 3AN FPGA Efficient Morlet wavelet function architecture trade-off between speed and silicon area
Spatial	[72] CSP	Hardware: Altera FPGA platform (Stratix-IV)

		Dataset: motor imagery in BCI competition Classifier: MD Time delay of 0.5 seconds
	[73] CSP	Improve of [72], reducing the delay to 0.399s
	[74] SCSSP	Hardware: Virtex-6 FPGA Dataset: BCI Competition IV-dataset 2a Classifier: SVM
	[47] Asymmetric indices + PSD	Based on a scaled version of IHPR with a customized LUT
	[50] Asymmetric indices + PSD	Improve of [47], LIHPR reduces the gate counts and power consumption then IHPR
	[76] DTF	Used to describe the incidental effect of one EEG channel on another channel at a specific frequency Difficult to implement in hardware as it requires calculation of the transfer matrix, square roots and division of the multivariate autoregressive model
Raw	[81] PSD CNN	Dataset: Own For emotion recognition and fatigue-driving

3.2 Efficient Convolutional Neural Networks

Traditional feature extraction and classification methods like LDA, ICA, PCA, SVM, and KNN are easier to implement on hardware, but often fail to deliver satisfactory performance. Additionally, these models have weak generalization capabilities, making them less effective for handling complex EEG tasks. CNNs have demonstrated remarkable capabilities across various domains, ranging from image recognition to natural language understanding. It has become a dominant force in the software-driven deep learning field. Despite their excellence in feature learning and classification, CNNs often require substantial computational resources, especially for real-time applications like BCI. Therefore, researchers have developed a large number of hardware acceleration methods for CNN. This section introduces five highly efficient CNN hardware techniques specifically applied to EEG signal processing: pruning, quantization, tensor decomposition, knowledge distillation, and neural architecture search. The applications of these techniques in EEG signal processing are summarized in Table 3.3 at the end of this section.

3.2.1 Pruning

Dating back to the early 1990s, pruning emerges as a pivotal technique for mitigating the computational complexity of CNNs [82]. It revolves around identifying and eliminating redundant or less informative neural connections within a network. The core concept entails removing unimportant weights, filters, channels, or even entire layers from the original DNN. This aims to reduce both the number of memory access and computational operations required. The pruning process typically categorizes into two main strategies: weight pruning and structural pruning.

3.2.1.1 Weight Pruning

At the heart of efficient neural network design lies weight pruning. Its foundation rests on the understanding that not all neural connections carry equal importance. Through iterative identification and pruning of these less crucial connections, the network achieves sparsity without notable performance degradation. Various strategies have emerged to execute weight pruning effectively.

For example, the concept of leveraging second-order derivatives, specifically the Hessian matrix of the loss function, Optimal Brain Damage (OBD)[82] and Optimal Brain Surgeon (OBS)[83], was introduced to identify redundant weights. Weight pruning can generally be divided into three steps [84, 85]. Initially, the network undergoes training to discern crucial connections. Subsequently, less significant connections are pruned. Finally, the network is subjected to retraining to refine the remaining connections. This iterative process gradually results in a more condensed network structure. Moreover, neurons that become inactive due to all their connected weights being pruned can be abandoned.

Under the same memory constraints, certain studies indicate that pruned larger models can surpass smaller dense models in terms of accuracy [86, 87]. Conventional weight pruning techniques typically function in the spatial domain, however, [88] introduced a frequency-domain approach wherein spatial weights are transformed into frequency weight coefficients. This method dynamically prunes across different frequency bands during each iteration, resulting in higher compression ratios.

[84, 85, 88] apply a global magnitude threshold to the entire DNN, which can impact network performance because using the same compression rate for each layer is not always optimal. To handle this demand, variable compression rates have been introduced based on the importance of each layer within the network. Combining

selective learning with the identification of weight importance concerning the loss, and discarding other weights by cutting off the gradient flow is proposed in [89]. [90] utilizes a Gaussian Mixture Model (GMM) for each layer's weight distribution to establish layer-wise compression rates. The selection of pruning layers depends on the number of weights with slight oscillation estimated applying the GMM.

Moreover, amplitude-aware pruning methods such as [91] incorporate considerations for energy reduction alongside compression ratios and accuracy loss when devising pruning strategies.

Weight pruning method entails the removal of less significant weights or neurons, irrespective of their network position, leading to “unstructured” networks. The abundance of sparse matrix operations arising from weight pruning may necessitate specialized software or hardware setups, potentially restricting the usability of pruned networks on general-purpose hardware.

3.2.1.2 *Structural Pruning*

Conversely, structural pruning concentrates on eliminating complete filters, channels, or even entire layers. This method not only diminishes computational demands but also yields more concise network architectures, particularly beneficial in resource-limited settings. In contrast to unstructured pruning, structural pruning alters the input and output shapes of layers and weight matrices, yet it keeps compatible with dense matrix operations, thus necessitating no customized hardware support. The idea of structural pruning essentially focuses on identifying and eliminating unimportant filters. They are commonly classified into three main types:

- **Criterion-Based Pruning:** It encompasses methods where filters are prioritized according to particular criteria for pruning. Techniques such as L1-norm-based pruning [92] and Soft Filter Pruning (SFP) rank filters [93] based on both L1 and L2 norm criteria. Moreover, criteria involving activation neurons, such as Average Percentage of Zeros (APoZ) [94], aid in detecting and eliminating filters with low APoZ values.
- **Ranking-Based Pruning:** HRank [95] utilize feature map ranking as the metric for filter pruning. Feature maps with lower ranks are deemed to contain less essential information, thus becoming primary candidates for pruning.
- **Optimization-Based Pruning:** ThiNet [96] prunes filters in the present layer while aiming to minimize the reconstruction error in the subsequent layer. Conversely, NISP [97] aim to limit the reconstruction error specifically in the “Final Response Layer” (FRL).

Whole-layer pruning methods, extending to filter and channel pruning, represent an extension of structural pruning [98-100]. Layer pruning methods optional eliminate

layers from the network based on various criteria, leading to a dense pattern. Patterns acquired through layer pruning decrease inference time and runtime memory usage while preserving accuracy levels similar to models acquired through filter pruning.

An alternative perspective has emerged, diverging from the typical three-step pruning process of training large models, pruning, and fine-tuning. In structural pruning, re-randomly initializing and training the model has better performance than fine-tuning [101]. During the training process, innovative techniques such as network slimming[102] and group LASSO loss[103] automatically detect and prune irrelevant channels, helpfully compressing the pattern without requiring post-pruning fine-tuning.

3.2.2 Quantization

Quantization is another technique designed to enhance the efficiency of CNNs. By reducing the precision of the network's weights and activations, quantization significantly decreases the model's memory demand and computational requirements. The most common way is to convert a high-precision format (such as FP32) to a low-bit fixed-point format. Quantization can be combined with weight pruning to further enhance DNN's efficiency and significantly reduce energy consumption. It makes quantization become a vital strategy for efficient hardware EEG data processing.

During quantization, neural network data is restricted to a set of discrete levels. These levels can follow different distribution patterns, such as uniform or non-uniform distributions. Uniform distribution[104] (characterized by even step sizes) and non-uniform distribution[105] (typically represented by logarithmic distributions) are general quantization pattern. Quantization can be realized using deterministic methods or stochastic methods. Deterministic methods, such as projecting data onto the nearest discrete levels, ensure precision and consistency in the quantization process[104]. In contrast, stochastic methods require probabilistically identifying which of the two neighboring discrete levels the data will be projected onto. This introduces an element of randomness, which can potentially capture the distribution of the original data.

The source of quantizing neural network data can be discover until 1990s [106]. Quantization provides two main methods:

Post-training quantization [107, 108]: After the full training of network, FP32 weights and activations are converted to lower-precision version.

Quantization-aware training[109-111]: It includes quantization error into the training loss, which usually increases network accuracy.

Quantization has made significant strides. FP32 precision parameters successfully are decrease to INT8 while maintaining high accuracy [107]. [108] have advanced to 4-bit quantization, effectively removing the necessity for post-quantization fine-tuning. [109] demonstrated a 1.5% accuracy loss in the case of the ResNet-50 model using INT8 format for both training and inference. Moreover, the generalized bit precision[110] offer greater enhanced. This method allows for different bit depths to be used for storing weights and activation, offering more versatility compared to traditional INT8 quantization. For example, in [110], AlexNet was quantized using 1-bit weights and 2-bit activations, resulting in a top-1 accuracy of 51%.

INT8 training methods that are aware of quantization, as described in[111], improve the practicality and performance of quantization in deep neural networks. This is achieved by optimizing calculations in both forward and backward passes, incorporating loss-aware compensation and parameterized range clipping.

Within the realm of quantization methods, binarization is the lowest precision. The data is presented only two possible values: (0, 1) or (-1, 1). Resource-efficient XNOR and bit-count operations replace resource-intensive matrix multiplications due to this extreme decrease of data bit. Given its potential to drastically cut storage and computational demands, binarization has attracted considerable interest, especially for implementing deep neural networks on devices with limited resources. Nonetheless, binarization inevitably leads to considerable loss of information. Moreover, the binary nature of values poses optimization hurdles for binary neural networks.

Several approaches have emerged to mitigate the information loss and optimization difficulties, such as BinaryConnect[112], Binary Neural Networks (BNN) [113] and XNOR-Net[114]. BinaryConnect employs random methods for binarization process during both forward and backward propagation. BNN improves BinaryConnect by binarizing activations, representing a notable milestone in the research of binarization quantization. BinaryConnect and BNN use fusion of deterministic and stochastic binarization operation, simplifying the hardware implementations. Conversely, XNOR-Net utilizes an innovative method by incorporating scaling factors for binary parameters to mimic floating-point parameters.

Through this method, the weight quantization in XNOR-Net to be shown as $w \approx \alpha * bw$, where α shows the floating-point scaling factor of the binary weight bw .

Binarization has been continuously optimized in recent years [104, 114-117]. DoReFa-Net[104] focuses on reducing quantization errors when training. It quantizes gradients to speed up training. We Beyond focusing on individual layers, alternative binarization methods such as loss-aware binarization[115] and incremental network quantization[116] try to reduce the total binary weight loss throughout the entire network. IR-Net[117] introduces an innovative solution by employing an adaptive error decay estimator to maintain information flow during both forward and backward propagation.

Various quantization methods offer a toolkit for tailoring neural networks to meet particular needs in EEG signal processing, achieving a balance between computational efficiency, accuracy, and memory.

3.2.3 Tensor Decomposition

Tensor decomposition is a method designed to decompose weight tensors into tiny tensors or matrices, effectively reducing the parameter count in CNNs while maintaining accuracy. It's noteworthy that parameters in DNNs, found within convolutional layer weight tensors and fully connected layer weight matrices, often demonstrate low-rank characteristics [118]. This observation leads researchers to categorize algorithm design directions into two main classes: low-rank matrix decomposition and tensor decomposition.

3.2.3.1 Low-Rank Matrix Decomposition

At the heart of low-rank matrix decomposition techniques is the approximation of DNN layer weight matrices through the multiplication of several low-rank matrices. Singular Value Decomposition (SVD) [119] is a widely used low-rank approximation method, focusing on representing a matrix based on its singular values.

SVD resolves matrix $A \in R$ to three main parts:

$$A = UDV \quad (3.1)$$

where $U \in R$ and $V \in R$ are orthogonal matrices, and $D \in R$ is a diagonal matrix containing singular values. Achieving a compact and efficient network model involves retaining only the most crucial components within the decomposition matrices.

SVD-based techniques have been pivotal in numerous applications. For example, in [120], SVD is employed to decompose the product of weight matrices and input data, thereby enhancing model efficiency. In [121], researchers initiate sparsity into low-rank factorized matrices by assigning a lower rank to not important neurons, thereby enhancing compression rates. Furthermore, in [122], channel-wise SVD decomposition of convolutional layers is utilized to improve efficiency by partitioning kernels into two consecutive layers of different sizes.

However, the computational overhead of employing SVD at each training step can be significant and assessing the importance of DNN layers during training poses challenges. Some idea methods have been presented to solve this problem. In [123], a training method employing SVD is proposed to explicitly attain low-rank approximation without necessitating SVD at each training step. Moreover, [124] presents a combined matrix factorization form that concurrently decomposes layers with shared close architecture and improves them based on SVD. Together, these improvements highlight the key part of low-rank matrix decomposition in enhancing the capability of EEG data analyzing network and other DNN applications.

3.2.3.2 Tensorized Decomposition

Tensors, multi-dimensional data arrays, offer as a general framework for expressing complicated data. Within the realm of deep learning, tensors include a range of orders, with two-dimensional matrices often representing second-order tensors, frequently utilized in fully connected layers. At the same time, weight tensors of convolutional layers are depicted as 4th-order tensors. With the increasing of tensor orders, the quest for higher compression ratios has spurred the investigation of more complicated methods. Traditional low-rank matrix approximations face challenges in fully utilizing the complex structure inherent in tensor-based networks. Tensor decomposition broadens the scope of low-rank matrix decomposition to encompass higher-order tensors, fragmenting weight tensors into smaller components and consequently diminishing the network's complexity.

Noteworthy technique in tensor decomposition encompass Tucker decomposition[125], CANDECOMP/PARAFAC (CP) decomposition [126, 127], Tensor Train (TT) decomposition[128-130], and Tensor Ring (TR) decomposition[131]. In [125], Tucker decomposition is employed to reduce convolutional weight kernels with pre-established ranks. Concurrently, the CP

decomposition disassembles convolutional kernels into a set of 1st-order tensors, thereby decreasing parameters and training time. This approach necessitates an iterative procedure that ingeniously integrates decomposition with fine-tuning for every convolutional layer. [126, 127] diverge in their approach to decomposing convolutional layers. The former selectively decomposes only a subset of convolutional layers, whereas the latter decomposes all convolutional layers across the network. [132] presents a robust low-rank decomposition method that combines Tucker decomposition with CP decomposition. In this method, the core tensors obtained through Tucker decomposition are subsequently decomposed with CP decomposition.

Alternative tensor decomposition methods, like TT decomposition, seem to be more advantageous for RNNs [128]. Its implementation notably decreases RNN network parameters by up to 1000 times. But it encounters precision-balance challenges in CNN compression. Paper [129] suggests a TT format tailored for CNN models to address this problem. Furthermore, TT decomposition proves to be functional for compressing three-dimensional CNNs (3DCNNs) in [130], presenting a novel method for choosing TT ranks to optimize compression benefits. Another new strategy requires leveraging TR decomposition, as demonstrated in [131], which utilizes a progressive genetic algorithm to optimize rank selection.

To summarize, CP decomposition indicates tensors as the summation of 1st-order tensors, providing a distinct approach to compression. Conversely, Tucker decomposition dissects tensors into sets of matrices and a condensed kernel tensor. TT decomposition establishes complex three-dimensional tensor structures, especially proficient in handling high-order tensors. TR decomposition extends from TT, appearing as a linear combination of TT decompositions. These tensor decomposition methods persist in advancing the efficiency of EEG signal processing.

3.2.4 Knowledge Distillation

Knowledge distillation is a method aimed at inheriting knowledge from big, complex CNNs (referred to as teachers) to tiny, more simplified CNNs (referred to as students). The student network undergoes training to emulate the behavior of the teacher, yielding a concise model with performance comparable to that of the teacher. This concept originates from previous work [133] and has been further developed in the case of deep learning [134]. The main challenge of knowledge distillation lies in

the methods for transferring knowledge from the teacher to the student model. It includes three fundamental components: the knowledge to be transferred, the distillation algorithms used, and the architecture that defines the relationship between the teacher and student models. Therefore, knowledge appears in multiple forms, such as logits, activations, or features derived from intermediate layers of the teacher model. Distillation algorithms can be grouped into three categories: offline, online, and self-distillation.

Offline distillation [135-138] entails extracting knowledge from a well-trained teacher network. It utilizes the soft-label outputs of the teacher network and relies on a set of teacher model to train the student model. [135] employs data augmentation to leverage the output distributions of multiple teacher models. Particularly, [136] presents a specialized distillation approach for quantized network, showing that quantized student model can nearly match the accuracy of all precision teacher model while reaching high compression rates and faster inference. Comparing with [137], they introduces a data-free method, where the student network is trained using synthetic data answer generated by the teacher network. Online distillation [139-141] takes place concurrently during the training of both the teacher and student network. It utilizes online knowledge distillation by incorporating the soft-label outputs of the teacher model. [139] proposes training the student network at various checkpoints of the teacher model until convergence is reached. At the same time, Collaborative Learning Knowledge Distillation (KDCL) [140] dynamically produces high-quality soft targets via an ensemble method for one-stage online training.

Moreover, as noted in [138, 141], knowledge distillation spreads its impact to the intermediate layers of the teacher model. Layer Selective Learning (LSL) [138] presents a two-layer selection model, comprising the inter-layer Gram matrix and layered inter-class Gram matrix, which facilitates the extraction of intermediate layers in both the student and teacher model for knowledge distillation. In [141], an enhanced group knowledge distillation network harnesses features from intermediate layers, enabling the concurrent training of the produced student network and the ensemble of teachers without relying on well-trained teacher networks.

Conversely, knowledge distillation have made the development of fast neural network structure research for self-knowledge distillation [142, 143]. It eliminates the necessity of training large teacher networks. [142] presents an additional self-learning

method that transfers selected knowledge to the classifier network utilizing soft labels and intermediate features. In [143], a data augmentation method is utilized to collect knowledge using the best accuracy student network from past epochs while concurrently training the present student model, thus improving model robustness.

By incorporating hardware resource constraints during the training of both the teacher and student networks, specific hardware performance targets can be achieved. Paper [144] introduces NEMOKD, a knowledge distillation method that balances training volume, hardware latency, cost, and accuracy. This approach is compared on the Movidius Myriad X VPU and Xilinx Z7020 FPGA platforms. The hardware-aware distillation method significantly reduces computational demands and memory load, which is a new direction for wearable EEG devices.

3.2.5 Neural Architecture Search (NAS)

NAS is an automated method for uncovering the best neural network structure. NAS utilizes diverse search technology and capability analysis methods to traverse a pre-established search space, aiming to discover efficient CNN structure tailored for certain tasks with minimal or no manual selection. Different with above mentioned technology such as pruning, quantization, or tensor decomposition, which improve model structures, NAS does not depend on a certain model. Early NAS methods [145] originally generated numerous candidate network structure based on pre-established search space standard. Each structure goes thorough training until convergence and is subsequently order based on its accuracy on a validation dataset. These rankings offer feedback for refining the search scheme and generating novel neural structure. The NAS iteration process will not stop until the requirement is met. Finally, the best structure is carefully selected and evaluated with test data. Unfortunately, due to the need to train and evaluate many neural networks within a wide search space, it entails considerable computational and time costs.

NAS have some of direction of development to alleviate computational demands, such as the search space, search algorithms, and network evaluation standard. Recent NAS usually progress via three main phases: super network training, sampled networks evaluating, and discovered networks refining. It reduces the pathway to recognizing fast network structure while mitigating the computational costs linked with detailed training and evaluation.

3.2.5.1 Search Space

The complexity increases rapidly when try to explore network neurons and their mutual connection to explain a comprehensive network structure [145]. A successful method to limit the search space is to examine a small set of compact structure part, synthesizing the final network structure through iteratively repeated part. Some search space-based technology have been proposed such as NASNet [146], ENAS [147] and AutoDispNet [148]. For example, AutoDispNet concentrates on identifying two distinct cell types, referred to as normal cells and reduction cells, within a pre-established overarching network structure. Conversely, ENAS launched an approach in which all sub-network collectively share weights, leading to a substantial increase in GPU processing time but implement over a thousand times fast compared to previous NAS[145].

Efficient networks implemented in the NAS relies on achieving a balance between inference time (latency) and memory consumption, all while keeping system accuracy at a satisfactory level. MnasNet [149] is a platform-aware NAS designed specifically for mobile devices. Its goal is to navigate the complicated structure of network to uncover the best trade-offs between latency and accuracy.

3.2.5.2 Search Algorithm

In the complicated realm of NAS, search method determines the study of a designated search space. Generally, NAS entails a generator that produces sample structure and an evaluator responsible for evaluating their post-training performance. Due to the resource-limitedness of the training process, the selection of the search method profoundly impacts the overall promotion of NAS. Many search algorithms, including reinforcement learning (RL) and neural evolution algorithms, have appeared to solve this issue. RL has achieved notable improvement in NAS-RL [145] and MetaQnn [150], showcasing high accuracy in image identification tasks. Neuroevolution algorithms initially addressed the optimization of neural structures and weights [151]. However, the emphasis has shifted primarily towards optimizing neural structures lately [152]. [152] has demonstrated that both reinforcement learning and neural evolution invariably outperform random search (RS) regarding test accuracy. In particular, Neuroevolution invariably achieves a trade-off between high performance and compactness.

3.2.5.3 Performance Evaluation Strategy

The method for evaluating neural structure performance is a basic feature of the NAS field. At the beginning of NAS, the method involved evaluating the condition of each sampled network through well training from scratch [145]. But it cost highly expensive regarding computational resources and time. Progressive NAS [153] and ReNAS [154] have optimized this problem by devising accuracy predictors. These predictors are well trained by data from sampled networks in the search space. Well trained predictors direct the search process automatically. The cost also can be notably decrease through One-Shot NAS [155]. Every sampled network received weights from a well-trained super-network. Thus, it does not need additional training. This weight-sharing method is central to One-Shot NAS, significantly decreasing the computational load related to NAS and thereby enhancing overall efficiency.

Although automatically designed neural networks possess powerful capabilities, the complex construction may not always be satisfactory for deployment on wearable EEG devices due to hardware limitations. The hardware-aware NAS methods [156, 157] have lately appeared. [158] proposed a two-stage algorithm to define the search space. Under the same latency constraints, their method was compared across CPU, DPU, and VPU platforms, achieving higher accuracy than manually designed search spaces. [159] propose a memory-constrained search method that incrementally scales and prunes the network, ensuring efficiency across different devices while adhering to resource and memory limitations. These techniques combine hardware- concerning element such as inference latency, energy consumption, and memory utilization into the NAS. To decrease evaluation costs, these methods concentrate on training a single network that can adapt to various hardware architectures, reaching a balance between high performance and cost-effectiveness.

3.2.5.4 EEG-based Convolutional Neural Networks Hardware Acceleration

In recent years, more and more CNNs have been deployed on portable or wearable EEG devices. Many Hardware solutions mainly focus on architecture (CPU and GPU [160]) and spatial architecture (FPGA [52, 161] and ASIC [162, 163]). This section focuses on reviewing CNN accelerators applied to EEG signal processing.

Paper [161] deployed a VGG model accelerator on the PYNQ Z1 development board, optimized for motor imagery tasks. Figure 3.1 shows the CNN accelerator architecture. The paper highlights that using a 16-bit fixed-point format reduces DSP

and BRAM resource usage by 25% compared to the 32-bit floating-point format, without significantly impacting accuracy. However, using an 8-bit fixed-point format results in a notable accuracy drop of approximately 7%.

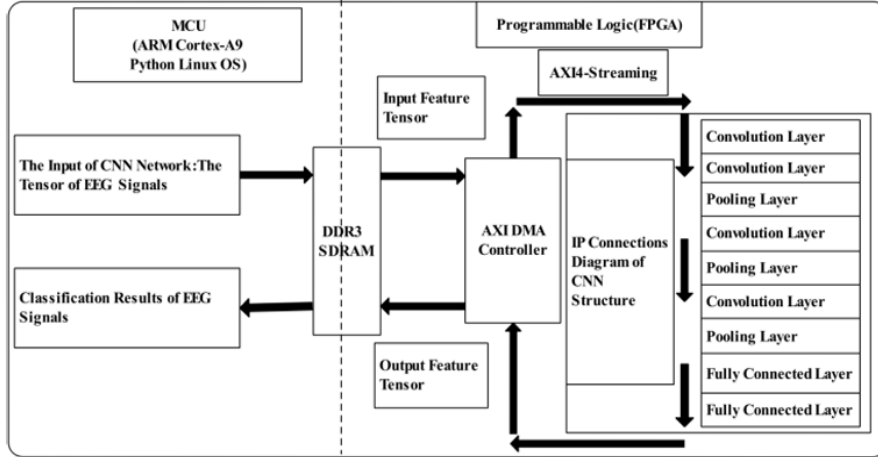


Figure 3.1 FPGA Block diagram of [161]

Paper [163] designed a CNN and STFT accelerator, deploying and testing it for an emotion recognition task using the DEAP dataset on a TSMC 28nm chip. Paper [162] deploys CNN accelerated inference on a 180nm 1P6M CMOS coprocessor. They highlight that the design achieves 97.8% accuracy with floating-point operations and 93.5% with fixed-point operations. The authors also customized the coprocessor interface (CCI) for communication between the coprocessor and RISC-V. Figure 3.2 show the system diagram of their work.

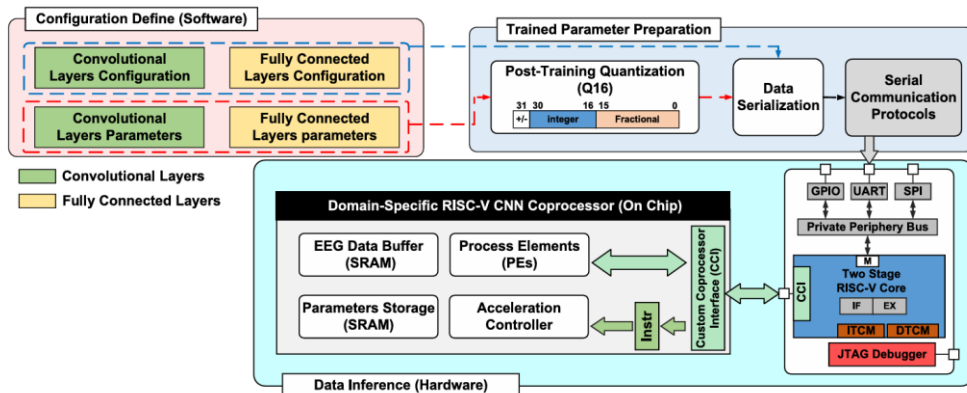


Figure 3.2 The block diagram of [162].

Due to the FPGA's parallel processing capabilities, programmable flexibility, low power consumption, and high efficiency, it has become increasingly popular for developing portable EEG processing devices, attracting more researchers to this field. Paper [160] evaluated the deployment of the motor imagery task across various platforms (CPU, GPU, FPGA) and concludes that FPGA is the optimal solution in terms of figure of merit (FOM) and energy consumption.

	CPU	FPGA	Jetson Nano	
Power consumption (W)	13.22	1.47	5	10
Inference time (ms)	121.57	19.97	15.85	12.21
FOM (W × s)	1.61	29.36×10^{-3}	79.25×10^{-3}	122.1×10^{-3}
Memory footprint (Mb)	476	6.37	415	
Test accuracy	81%	77%	81%	

Figure 3.3 Comparison of the different platform for [160].

Paper [52] presents a specialized design for emotion classification tasks, BioCNN, which is implemented on a low-cost Spartan-6 FPGA using the Digilent Atlys board. Tested with the FEXD and DEAP datasets, it achieved 11 GOPS/W energy efficiency and a latency of under 1ms, far below the 150ms human-computer interaction threshold.

Although ASICs offer exceptional performance and efficiency for specific applications, their high development costs and lack of flexibility make them suitable only for mass production in specialized use cases. On the other hand, while GPUs perform well in general computing, the literature we reviewed shows they are typically implemented in high-power PCs, with their limited customization capabilities restricting their use in portable devices. In comparison, FPGAs not only accelerate CNN inference by executing numerous multiply-accumulate (MAC) operations in parallel but also optimize memory access and data flow through customized hardware logic designs. The local storage and control logic of FPGA processing elements (PEs) allow direct communication with minimal memory bandwidth, significantly improving data reuse, making FPGAs ideal for low-power, efficient portable EEG devices with high computational demands.

The chapter presents an in-depth analysis of hardware implementation techniques for EEG signal pre-processing and five types of feature extraction: frequency-based, time-frequency, spatial features, and raw signals. Following this, it researches efficient CNN solutions for EEG signals, covering methods such as pruning, quantization, tensor decomposition, knowledge distillation, and NAS. Finally, this chapter discusses existing CNN-based EEG accelerators.

Table 3.3 List of Collected CNN Hardware Techniques

Method	Authors	Work Description
Weight Pruning	[82], [83]	Hessian matrix of the loss function
	[84], [85]	Three-step method
	[88]	Frequency-domain approach
	[89]	Variable compression rates
	[90]	GMM
	[91]	Energy reduction with compression ratios and accuracy loss
Structural Pruning	[92], [93], [94]	Criterion-Based Pruning
	[95]	Ranking-Based Pruning
	[96], [97]	Optimization-Based Pruning
	[98], [99], [100]	Whole-layer pruning methods
	[101], [102], [103]	Automatically identify and prune
	[104]	Characterized by even step sizes
Quantization	[105]	Represented by logarithmic distributions
	[107], [108]	Post-training quantization
	[109], [110], [111]	Quantization-aware training
Low-Rank Matrix Decomposition	[120]	Decompose the product of weight matrices and input data
	[121]	Sparsity: Maintaining a lower rank for less critical neurons
	[122]	Channel-wise SVD decomposition by dividing kernels into two layers
	[123]	Low-rank approximation without invoking SVD at every training step
	[124]	Joint matrix factorization scheme
	[125]	Tucker decomposition
Tensorized Decomposition	[126], [127]	CANDECOMP/PARAFAC (CP)
	[128], [129], [130]	Tensor Train decomposition
	[131]	Tensor Ring decomposition
Knowledge Distillation	[135], [136], [137], [138]	Offline distillation

	[139], [140], [141]	Online distillation
	[142], [143], [144]	Self-knowledge distillation
Neural Architecture Search Space	[146] [147] [148] [149]	NASNet ENAS AutoDispNet MnasNet
Neural Architecture Search Algorithm	[145] [150] [152]	NAS-RL MetaQnn-RL Optimizing neural structures
Neural Architecture Search Performance	[153] [154] [155]	Progressive NAS ReNAS One-Shot NAS
Evaluation Strategy	[156], [157], [158], [159]	Hardware-aware NAS

The results presented here provide the basis for the second thesis point, which is belong to the thesis group 1 and it has been published to Sensors Journal with title “A Comprehensive Review of Hardware Acceleration Techniques and Convolutional Neural Networks for EEG Signals[J5].”

Thesis 2: I provided a comprehensive review of various state-of-the-art EEG-based hardware solutions, emphasizing pre-processing and feature extraction stages, and hardware convolutional neural network acceleration techniques for EEG signal processing, focusing on efficiency and performance improvements.

4. Research on EEG Signal Algorithm of Eye State

In this chapter, we will utilize some previously discussed EEG signal processing and classification methods for detecting eye states. We evaluate the efficiency of multilayer perceptron (MLP) and CNN models with the PhysioNet dataset and compare their performances. Additionally, we will test CNN on Eye EEG data we have collected ourselves.

4.1 Eye State Research on PhysioNet data

4.1.1 PhysioNet EEG Dataset

The PhysioNet EEG dataset consists of over 1500 EEG recordings, each lasting one or two minutes, from 109 subjects [164]. During these recordings, the subjects performed six different tasks while EEG data were captured from 64 electrodes base on the BCI2000 system. Each volunteer completed 14 trials. Trials 1 and 2: Subjects performed eye opening and closing tasks. Trials 3, 7, and 11: Subjects performed real motor imagery tasks including left and right hand fist movements. Trials 4, 8, and 12: Subjects performed imagined motor imagery tasks including left and right hand fist movements. Trials 5, 9, and 13: Subjects performed real motor imagery tasks including both hand and foot opening and closing. Trials 6, 10, and 14: Subjects performed imagined motor imagery tasks including both hand and foot opening and closing. The EEG recordings, taken from 64 channels at a sampling rate of 160 samples per second, adhered to the international 10-10 system, excluding electrodes Nz, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10, as shown in Figure 4.1. We will focus on the segments involving two one-minute eye open (EO) and eye close (EC) tasks in this study. For our experiments, we will use data from eight channels: C3, C4, Fp1, Fp2, P7, P8, O1, and O2.

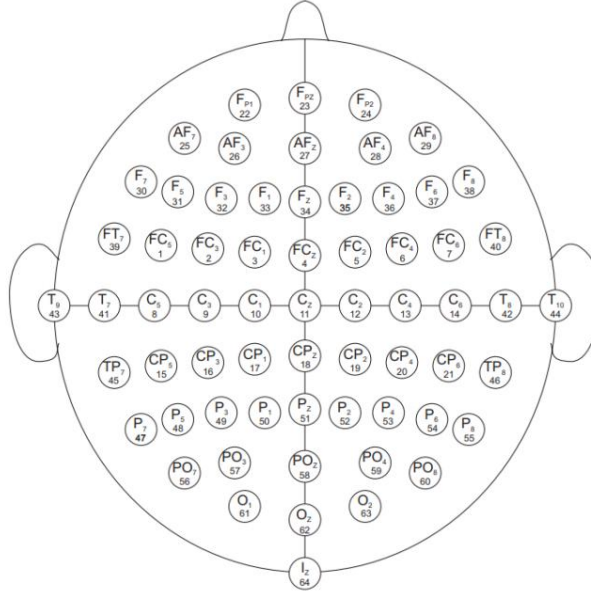


Figure 4.1 10-10 system for electrode placement.

4.1.2 Data analysis and processing

The experiment environment was:

CPU: Intel Core i5-7300HQ, RAM: 16 GB, GPU: GeForce GTX 1050 Ti) with Window 10, 64 bit operating system and MATLAB R2020a. The data sets in this chapter have all been preprocessed by bandpass filtering and normalization.

MATLAB is a high-level programming environment that is widely used in scientific computing, signal processing, and biomedical engineering research. MATLAB was developed by MathWorks and is particularly suitable for analyzing (EEG). In EEG-based BCI research, MATLAB is often used for preprocessing, feature extraction, and model prototyping due to its extensive library support and compatibility with signal processing toolboxes. EEGLAB is an open-source toolbox running under MATLAB, specifically designed for analyzing electrophysiological signals such as EEG and MEG. The main functions of EEGLAB include ICA, artifact removal, time-frequency decomposition, and event-related potential (ERP) analysis. The EEG preprocessing in this chapter uses the EEGLAB toolbox, and the EEG training uses the MATLAB toolkit, Deep Learning Toolbox.

The initial data from the eight scalp channels are roughly two minutes in duration, sampled at a frequency of 160 samples per second. To augment the number of samples, we segmented the data into four-second intervals (640 points each).

The first eight seconds of PSD of EC and EO obtained by first volunteer is illustrated in Figure 4.2 and Figure 4.3. Each colored line represents the spectral activity of an individual data channel. The leftmost scalp map shows the power distribution at 10.3 Hz, concentrated in the occipital regions and the parietal midline. The other scalp maps display the power distribution at 15.5 Hz and 20.6 Hz. It is clearly observable that the alpha wave power is markedly higher in the EC state compared to the EO state, with significant differences in the frontal, parietal, and occipital regions.

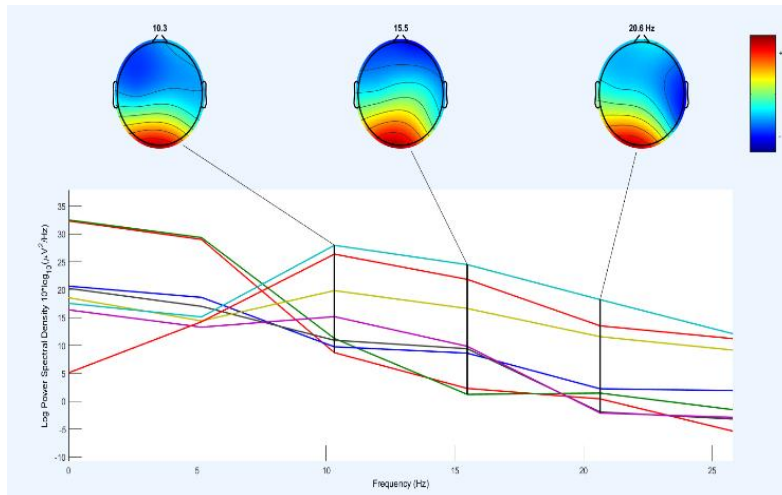


Figure 4.2 Eight Seconds Window PSD of EC

Figure 4.4 and Figure 4.5 show the scalp location of O2 (channel 8), its activity spectrum, and an ERP-image plot of EC and Eye Open EO activities in single epochs, respectively. A wave peak at 8-14 Hz is visible in Figure 4 compared to Figure 5. Alpha waves emerge when individuals are awake, serene, and with their eyes closed. However, upon opening their eyes, engaging in cognitive tasks, or receiving stimuli, alpha waves fall and transition into faster waves. It reappears when the individual regains calmness and closes their eyes again. This phenomenon, termed “alpha blocking [165]”, highlights alpha waves as the principal manifestation of cerebral cortex electrical activity during wakefulness, calmness, and the EC state.

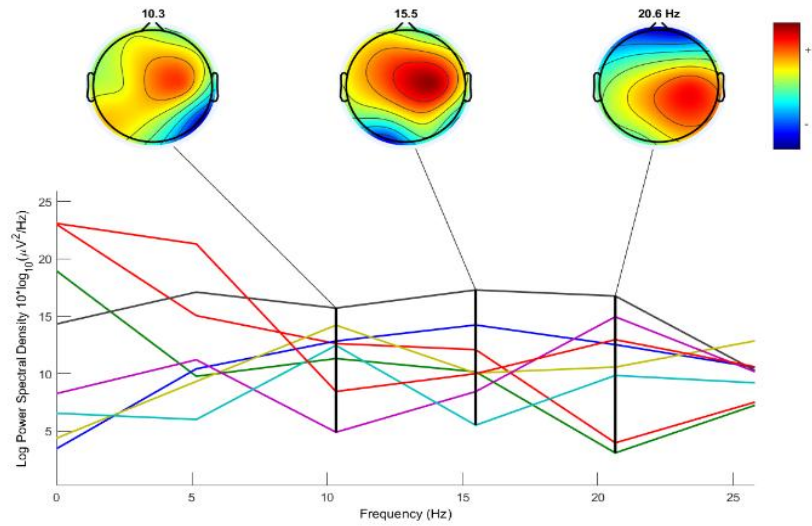


Figure 4.3 Eight Seconds Window PSD of EO

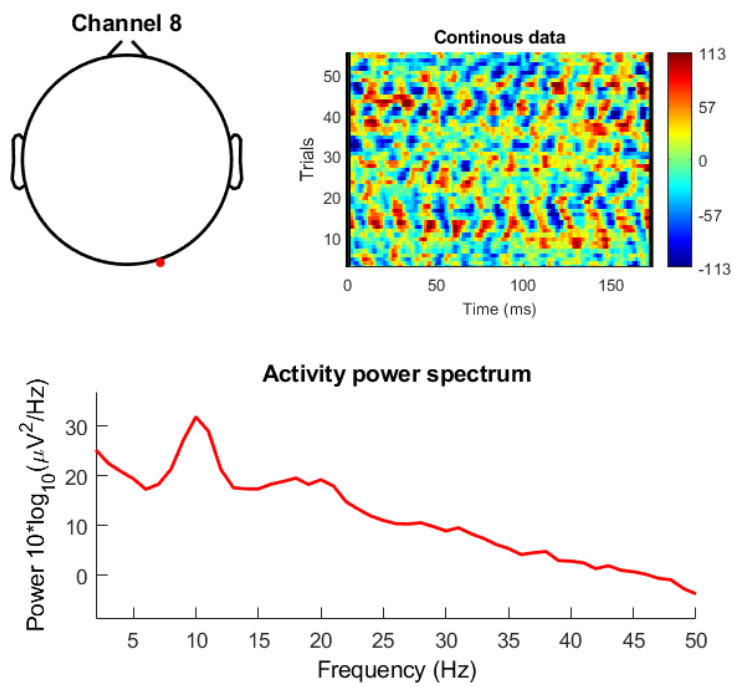


Figure 4.4 Power spectrum of channel 8 of EO

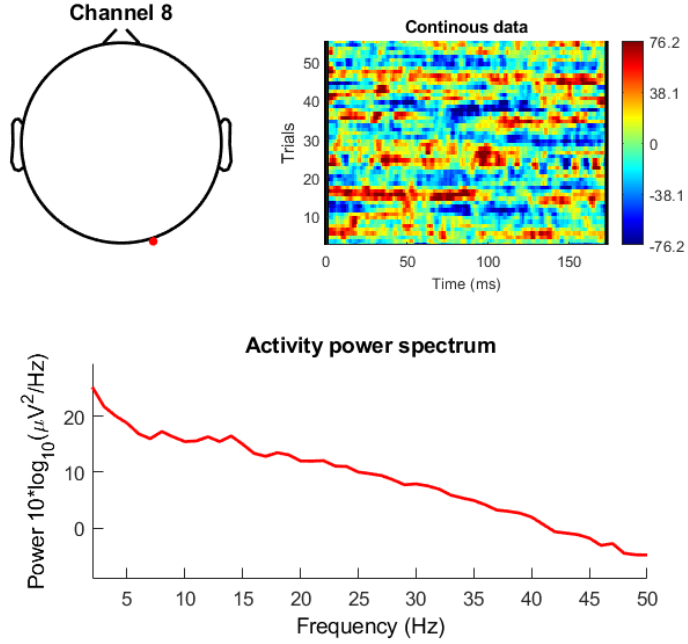


Figure 4.5 Power spectrum of channel 8 of EC

Beta wave activity demonstrates more stability compared to alpha wave activity. Because the amplitude of beta waves is notably lower than that of alpha waves, as shown in Figure 4.4 and Figure 4.5. Consequently, it might be more logical to utilize beta waves as an analytical indicator for eye state recognition. Although alpha and beta waves have been used in research in the field of EEG, most of the research does not include correlating and mutually comparing the two types EEG waves. Consequently, beta waves have been included as additional features for comparison in subsequent experiments.

4.1.3 Construction of a multilayer perceptron (MLP) network

The MLP neural network is structured as a multi-layer feedforward neural network, comprising an input layer, hidden layers, and an output layer. Each layer is composed of numerous neurons, with each neuron receiving input from the preceding layer and transmitting it to the subsequent layer. We denote i, j, k as the indices representing neurons in the upper layer, current layer, and next layer, respectively.

The formula for MLP forward propagation is:

$$h_j = \sum_{i=0}^M w_{ij} x_{ij} \quad (4.1)$$

Where h_j represents the weighted sum of all inputs to the current node, w_{ij} represents the weight from each neural network in the previous layer to the current neural network, x_{ij} is the current input.

$$a_j = f(h_j) = f\left(\sum_{i=0}^M w_{ij} x_{ij}\right) \quad (4.2)$$

Where a_j represents the output value of the hidden layer neuron. $f()$ represents the activation function, usually used Rectified Linear Unit

$$y = \max(0, x) \quad (4.3)$$

(ReLU), tangent hyperbola (tanh) or sigmoid function. The expressions of the three functions are:

ReLU:

$$y = \max(0, x) \quad (4.3)$$

Tanh:

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4.4)$$

Relu:

$$y = \frac{1}{1 + e^{-x}} \quad (4.5)$$

The expression of the output value of the output layer is:

$$y = a_k = f(h_k) = f\left(\sum_{i=0}^M w_{jk} x_{jk}\right) \quad (4.6)$$

where w_{jk} is the weight from the current neuron to the next layer neuron, h_k represents the sum of the input weights of the output layer neurons.

The backpropagation algorithm of MLP computes the error between the output layer and the target output, then adjusts the weights along the gradient direction to systematically minimize the error. This process can be formulated as follows:

$$\Delta w = lr(h - y)f'(h_k) \quad (4.7)$$

where Δw is the increment of weights, lr is the learning rate.

The structure of MLP used in this study is illustrated in Figure 4.6. We employed the Levenberg-Marquardt training function (trainlm), which yielded optimal results with a network comprising two hidden layers, containing 12 and 7 neurons respectively, utilizing log-sigmoid transfer functions. The output layer consisted of one neurons, corresponding to two eye activities to be recognized. Initial weights on all layers were initialized from a normal Gaussian distribution. We set the epoch limit to 1000 cycles for training.

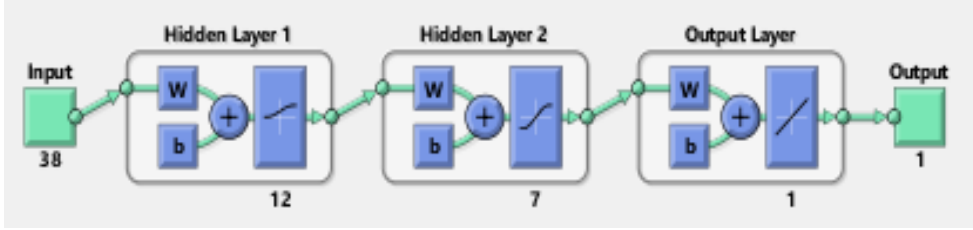


Figure 4.6 MLP Network Construction

4.1.4 Construction of Simplified Convolutional Neural Network (SCNN)

Deep learning is a fundamental aspect of machine learning, representing a rapid field that continues to progress. It has become the focus of scholars in various fields. [166-169]. Within deep learning, Convolutional Neural Networks (CNNs) are a prevalent neural network model with extensive applications across diverse fields [170-172], and it is also have notable research progress in the field of BCI [173, 174].

At the core of CNNs lies the convolutional layer, tasked primarily with extracting features from the input signal. Convolutional kernels, also referred to as filters, execute pertinent convolutional operations and they may exist in multiple instances within a single convolutional layer. During the training of the neural network, the weight parameters and biases of the convolutional kernels are subject to modification. Convolutional operations utilize principles of matrix multiplication to map the features from input to output. The position of the neural element in the feature map produced by the k th convolution kernel is denoted as (m, n) . Equation (4.8) illustrates the network.

$$y_{m,n} = f(w(i)_k * I(m, n) + b) \quad (4.8)$$

where $f()$ represents the activation function, commonly found in forms such as tanh, sigmoid, or ReLU, k signifies the first convolution kernel of the k th layer, $I(m, n)$ represents the input data, b denotes the bias,

Additionally, successive connections to deeper layers facilitate the extraction of increasingly abstract features by combining multiple convolutional layers. Apart from the convolution operation, the convolutional layer also encompasses operations such as padding and stride, rendering the associated computational process more complex.

Pooling, also recognized as the sampling layer, entails reducing the length and width dimensions of the input image. Situated subsequent to the convolutional layer, the pooling layer executes down-sampling procedures to extract local features. This layer assists in diminishing the network's parameter, thus lowering computational workload and model intricacy. Concurrently, pooling can enhance the network model's robustness to minor data discrepancies and reduce overfitting concerns. At the same time, it may cause the lost of some important features. Pooling can also enhance the network model's robustness against minor data errors and alleviate overfitting issues.

The pooling layer includes two pooling operations. Through maximum pooling, the feature value of the target region is determined by the highest element value within the region. This operation accentuates local features and mitigates minor errors caused by the convolution layer. Similarly, with average pooling, the feature value of the target region is determined by the average of its constituent elements. Besides preserving more adjacency information, average pooling aids in mitigating errors within the region stemming from the size of the convolution kernel. The expressions for maximum and average pooling are presented in Equations (4.8) and (4.9) respectively. We supposed that the size of the pooling kernel is $(N * N)$.

$$\text{Max pooling: } f(x) = \max(x_{[m,m+N],[n,n+N]}) \quad (4.9)$$

$$\text{Average pooling: } f(x) = \frac{1}{N * N} \sum_{m=m_1, n=n_1}^{m=m_1+N, n=n_1+N} x_{m,n} \quad (4.10)$$

Serving as the concluding component of the network, fully connected layer performs several functions for the final classification or prediction task. The fully connected layer takes the multidimensional feature maps generated by the preceding convolutional and pooling layers and transforms them into a one-dimensional vector. This process also known as flattening. Once the feature maps are flattened into a vector, they are fed into the fully connected layer where matrix multiplication occurs. This operation involves multiplying the input vector by a weight matrix, adding a bias vector, and applying an activation function to produce the output. The fully connected

layer applies non-linear activation functions such as ReLU, sigmoid, or softmax. Generally, the ReLU function is employed after the convolution operation of the intermediate layer to enhance the nonlinearity of the network. The softmax function is used in the output layer to calculate the probability distribution of k output classes. Due to the large number of parameters, fully connected layers can lead to overfitting, especially with limited training data. Regularization techniques such as dropout are often employed to reduce this risk.

Design of sparse connections, weight sharing, sampling, and nonlinear transformation make the CNN network model more robust. Manual feature processing of a large number of samples is required to achieve high-accuracy classification in traditional classification research. In contrast, the CNN model automatically learns the intrinsic characteristics of the signal through convolution operations and related processes [175]. It not only saves the trouble of manual feature extraction but also has excellent classification capabilities.

The advantages of CNN accuracy typically become obvious when the number of sample sets is sufficiently large. Another way to improve accuracy is to increase the number of layers in the network. The more complex the network, the more parameters it needs to train, and the more likely it is that overfitting will occur if there are not enough samples. Therefore, high-complexity network models will not be used in our subsequent studies. On the contrary, we will also simplify the structure of CNN. Some scholars have proposed that the pooling layer can be streamlined to improve the accuracy of CNN [176].

In this chapter, we enhance the network structure by eliminating the pooling layer from the conventional CNN to avoid losing valuable features. We refer to as simplified CNN (SCNN). The layers used in the SCNN are outlined in Table 4.1. The input data were transformed into grayscale images with the size of $64 \times 1 \times 8$ (8 channels, 64 measurement points).

Table 4.1 Outline of SCNN

Layer number	Layer type
1	Image input layer(64x1x8)
2	Convolution layer (13x1 64)
3	Batch normalization layer
4	ReLU layer
5	Convolution layer (13x1 128)
6	Batch normalization layer
7	ReLU layer
8	Convolution layer (39x1 256)
9	Batch normalization layer
10	ReLU layer
11	Fully connected layer(2)
12	Softmax layer
13	Classification layer

Before training SCNN, many hyperparameters need to be set. We will explain in detail the role of hyperparameters and specific values.

- **Optimizer:** The training algorithm used to update the weights based on the gradient descent algorithm (Momentum, SGD, Adam etc.). We use momentum in our network because it provides faster convergence, better handling of local minima, reduces oscillations and smooths the noise of the data.
- **Batch Size:** Batch Size is used to estimate the gradient in each iteration. A larger batch size leads to a more accurate estimate of the gradient and faster computation due to parallel processing but requires more memory. On the other side, it can offer regularization benefits but might result in noisy gradient estimates and slower training with smaller batch sizes. The value of 64 is set for our case.
- **Learning rate:** Learning rate is the step size at each iteration while moving toward a minimum of the loss function. It can speed up training but may overshoot the optimal solution with a high value, leading to divergence. A low learning rate ensures more precise convergence but can result in slow

training and getting stuck in local minima. We set a decreasing rate, reduced by a factor of 0.1 every 10 epochs with 0.2 initial learning rate.

- **Number of epochs:** We set the number of epochs for training to 50. Because more epochs can make the model learn better, but easy lead to overfitting.
- **Regularization Techniques:** Regularization adds penalties to the loss function or randomly drops connections, helping to generalize the model better. We set L2 regularization to 0.0005 to prevent overfitting.

4.1.5 Comparison results of MLP and SCNN

Segmentation and overlap represent common strategies to augment data volume, a practice known to markedly enhance classification accuracy [177]. Moreover, it proves highly advantageous in terms of curtailing computational resource utilization within network operations. By diminishing input size and the volume of transmitted and stored data during training and inference, these methods not only optimize performance but also slash associated costs. To understand the effect of window size on band power, we experimented with various window sizes and repeated training and testing 1000 times to reduce errors. The neural network used is a 2-layer MLP with 12 and 7 neurons in the hidden layer. Table 4.2 summarizes the average and best accuracy on test data using different window sizes. The window size of 40 sample size has the highest best accuracy and well mean accuracy, which is the best choice for our next experiment.

Table 4.2 Comparison of different window size

Window size	Mean accuracy	Best accuracy
20	84.46%	91.92%
40	84.23%	94.23%
60	81.00%	90.00%
80	80.87%	90.00%
100	80.26%	88.85%
120	79.66%	86.54%
140	78.71%	84.62%
160	78.94%	85.00%
180	78.57%	86.54%
200	78.50%	85.38%

We mentioned earlier that increasing the sample size might improve accuracy. Therefore, we extended the dataset by creating a segment to include half of the sampling points from the previous segment. Table 4.3 shows a comparison of training time and average accuracy. In Table 4.3, the suffix “A” indicates that only α waves are considered as features, “B” indicates that only β waves are considered, “A and B” indicates features of a combination of α waves and β waves, and “L” indicates that the extended data set use. All accuracy results are averages of 100 times 10-fold cross-validation.

Table 4.3 Comparison of training time and Accuracy

Type of neural network	Training time	Mean Accuracy
MLP_A (13 x 1300)	0.76s	81.95%
MLP_A_L (13 x 22400)	14.09s	88.47%
MLP_B (24 x 1300)	0.64s	80.46%
MLP_B_L (24 x 22400)	12.03s	92.66%
MLP_A_B (38 x 1300)	1.10s	84.23%
MLP_A_B_L (38 x 22400)	14.25s	94.85%
CNN	4m50s	88%
CNN_L	7m14s	91%

As far as MLP is concerned, MLP_A and MLP_B do not show significant differences. However, MLP_A_L and MLP_B_L yields a notable difference. At the same time, the results show that MLP_A_B_L reaches a best accuracy (94.85%). The CNN network achieved an accuracy of 91%, but the training time was significantly higher than that of the MLP.

The results presented here provide the basis for the third thesis point, which has belonged to the thesis group 2. The result has been published to 1st Conference on Information Technology and Data Science with title “Comparison of EEG Data Processing Using Feedforward and Convolutional Neural Network[C2].”

Thesis 3: I found that it would be reasonable to use the beta wave as an EEG analytical indicator. There was no significant difference in the training results when using alpha or beta features individually. And combining both features simultaneously improved accuracy.

4.1.6 Conclusion

The first paragraph of this chapter presents the well-known public dataset PhysioNet EEG. Then we use the EEGLAB toolbox to analyze the spectrum of EO and EC states. We introduce the construction of a multi-layer perception (MLP) network and propose a two-layer MLP model. At the same time, we introduce the SCNN architecture in detail, including the configuration of convolutional layers, pooling layers, and fully connected layers, its advantages in classifying EEG data, and the specific hyperparameters used in the training process. Comparing the two networks, we find that classifying alpha or beta waves alone as features does not show significant differences. But it raises a notable difference employing the extended dataset yields. Consequently, it appears more reasonable to utilize beta waves as an analytical indicator rather than alpha waves for this purpose. While we found higher accuracy and shorter training times from MLP instead of CNN, it is important to note that CNN combines signal preprocessing, feature extraction, and classification to keep away from the blindness associated with EEG signal processing. While CNN is still maintaining good accuracy rates. Each approach—MLP and CNN—has its advantages and trade-offs, and the choice between them depends on the specific requirements of the task and the characteristics of the dataset. In the next section we will use EEG signal acquisition equipment to collect and analyze our own data and then compare and discuss several different preprocessing methods and segment sizes.

4.2 CNN efficiency test on own data

In this section, a consumer-grade OpenBCI device is used to collect our own data, and then we compare and discuss several different preprocessing methods and segment sizes.

4.2.1 Data collection and preprocessing

The collection equipment used in the collection experiment is OpenBCI Cyton Biosensing Board amplifiers and OpenBCI EEG electrode cap which is a 21-channel electrode cap with coated electrodes[178]. Figure 4.7 shows the photo of OpenBCI 21-channel electrode cap using in this study.



Figure 4.7 Photo of the OpenBCI device

The Cyton board supports two recording modes: 8-channel and 16-channel. The 8-channel mode supports a sampling frequency of 250 Hz. The 16-channel mode supports a sampling frequency of 125 Hz, and an add-on board (Daisy) is also required. We utilized 8 following labels according to the internationally 10–20 system: C3, C4, Fp1, Fp2, O1, O2, P7, and P8. These electrodes' location corresponds to the labels on the electrodes shown in Figure 4.8. T3, T4, T5, and T6 have been updated to T7, T8, P7, and P8, respectively, as the older labeling were applied by OpenBCI [12]. These placements are interchangeable, and although the labels on the cap use the older nomenclature, they match the electrodes depicted in Figure 4.8.

This acquisition experiment recorded the EEG signals of 10 volunteers. Each participant completed 10 trials, where each trial included one minute of eye-open recording and one minute of eye-closed recording. We developed our own software to provide visual and audio cues to the volunteers and to add time stamps to the recorded EEG signals. Because the EEG data collected from scalp electrodes are often affected by noise, the signals captured directly do not accurately reflect the brain's neural activity. Therefore, it is essential to preprocess and de-noise the raw EEG data to minimize or eliminate the impact of artifacts as much as possible.

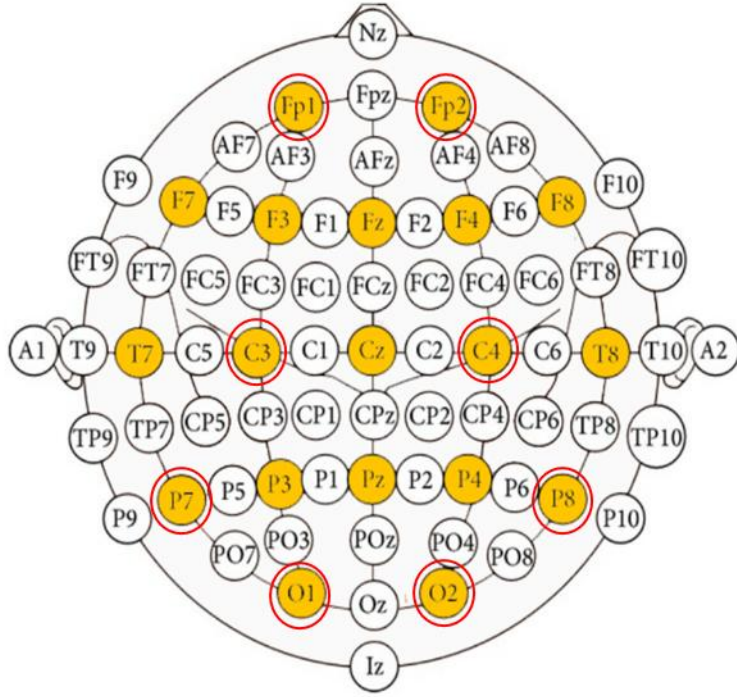


Figure 4.8 The placement of OpenBCI cap electrodes. Where the electrodes on the OpenBCI EEG Electrode Cap are indicated by yellow circles. The electrodes of Our project are represented in red circles

We applied a bandpass FIR filter spanning from 8 to 30 Hz on the gathered data. As we mentioned in section 4.1 that this frequency band, namely alpha and beta waves, is most closely related to eye activity. Figure 4.9 presents a comparison of partial EEG signals pre and post filtering, showcasing a reduction in clutter following the filtering process.

FFT is the frequently utilized analysis tool of EEG data. But it cannot address the issue of information localization in the time and frequency domains. Indeed, EEG signals include features in both the time and frequency domains. While some researchers concentrate solely on either time or frequency domain signals, possibly overlooking vital information, others aim to combine diverse features to improve classification performance. To overcome this limitation, FFT has been extended, resulting in the creation of various methods based on time-frequency domain analysis, such as the Short-Time Fourier Transform (STFT) and Continuous Wavelet Transform (CWT). Which are used in subsequent research.

Gabor Dennis initially proposed STFT in 1946. He used a time window to identify the frequency information at a specific moment. It addressed the issue of localizing information in time-frequency domain. The STFT is expressed as Equation 4.11:

$$STFT_x(w, \tau) = \int_{-\infty}^{\infty} x(t) w^*(t - \tau) e^{-jw\tau} dt \quad (4.11)$$

where $x(t)$ is EEG signal and $w(t)$ is a temporal window, $w^*(t - \tau)$ is the complex conjugate of $w(t)$. The size of the time window determines the relationship between frequency and time domain. A shorter time window enhances time resolution, while a narrower frequency window boosts frequency resolution. [13].

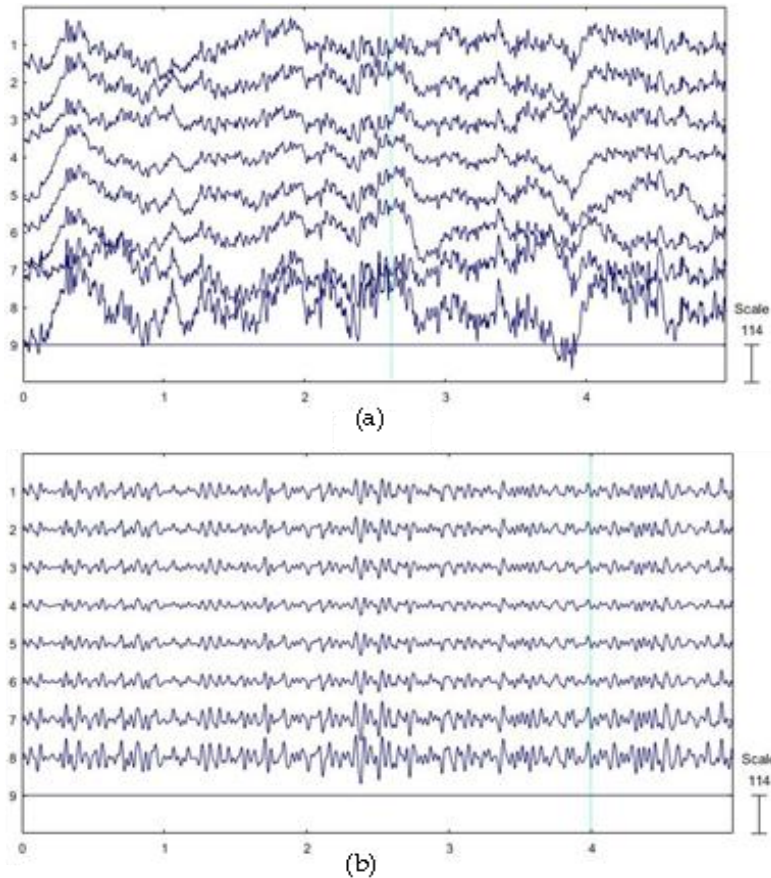


Figure 4.9 Comparison of 8-channels 4 seconds EEG signal before and after filtering. (a) is the EEG signal before filtering and (b) is the EEG signal after filtering.

In 1984, CWT was introduced to address the challenge that the STFT could not consider both global and local features due to the fixed window size [14]. The CWT is expressed in Equation 4.12.

$$w_s(a, \tau) = a^{\frac{1}{2}} \int s(t) \phi^* \left(\frac{t - \tau}{a} \right) dt \quad (4.12)$$

where $s(t)$ is the input signal, a is the scale of the wavelet transform, ϕ is the wavelet basis function, and τ is the time shift. Morlet, Mexican Hat, Harr, Daubechies, and SymN clusters are frequently employed wavelet basis functions. After testing, the Morlet wavelet is most suitable for our purpose. The time and frequency domain of Morlet wavelet is expressed in Equation 4.13 and Equation 4.14.

$$\phi(t) = \left(\frac{2}{\pi T^2} \right)^{\frac{1}{4}} \exp \left(-\frac{t^2}{T^2} + j w_c t \right) \quad (4.13)$$

$$\phi(w) = \left(\frac{T^2}{2\pi} \right)^{\frac{1}{4}} \exp \left(-\frac{(w - w_c)^2}{4T^2} \right) \quad (4.14)$$

Figure 4.10 and Figure 4.11 illustrates the comparison of filtered signals during EO and EC states, alongside their corresponding CWT and STFT feature images. It was apparent that energy is predominantly concentrated and higher amplitude in the 8-13 Hz frequency band on the Figure 4.10 (a), which is not shown on the Figure 4.10 (b). The same characteristic can also be clearly seen from Figure 4.11. This is the alpha block phenomenon, which we mentioned in section 4.1.2. It shows that the data we collected is a valid EEG signal, which has obvious eye state characteristics.

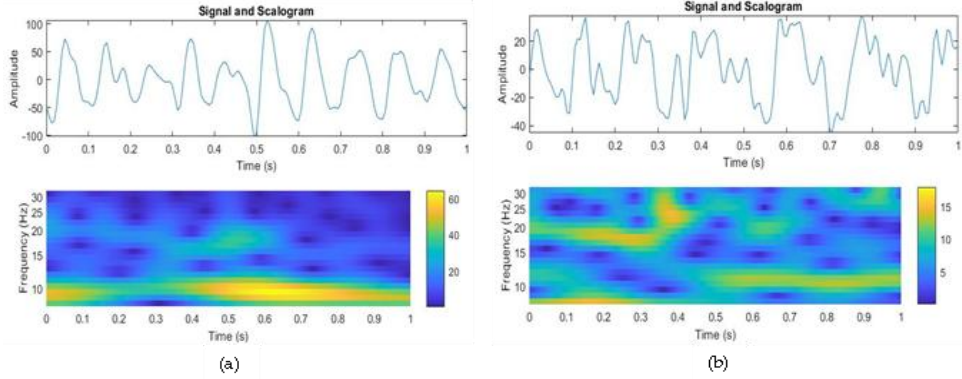


Figure 4.10 Time-frequency signal analysis diagram by CWT. where (a) shows the signal when the eyes are closed while plots the original signal in the upper subplot and the CWT scalogram in the lower subplot, and (b) shows the signal when the eyes are open.

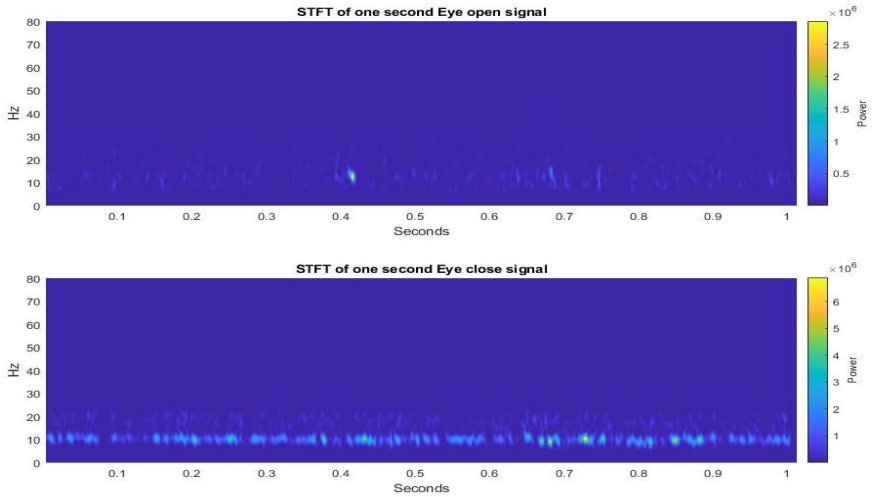


Figure 4.11 The STFT map of EO and EC signals

4.2.2 The proposed network structure

The block diagram in this study is illustrated in . The collected signal is first filtered by FIR to extract the signal in the frequency band we are interested in. In order to compare the effects of different preprocessing methods on CNN feature classification, we process the filtered signal by STFT and CWT respectively and compare it with the signal without any processing. The structures of the CNNs used for comparison are shown in Figure 4.13. We try to keep the trained networks with different preprocessing similar structure and same training hyperparameters to reduce

experimental errors. All data are normalized with center 0 and standard deviation 1 as the input of CNN.

- **Optimizer:** Stochastic gradient descent with momentum.
- **Batch Size:** 64 per iteration.
- **Learning rate:** The learning rate was reduced by a factor of 0.05 every 10 epochs, starting with an initial learning rate of 0.2.
- **Number of epochs:** The maximum number of training epochs was set to 500.
- **Regularization Techniques:** L2 regularization was 0.0005.

The purpose and description of each hyperparameter are detailed in Section 4.1.4.

To prevent overfitting, we implemented an “early stop” mechanism, which terminates training before the test error increases, even if the training process has not fully converged. That is, training will be stopped if the accuracy does not increase within 10 epoch. Similar to the network mentioned in 4.1.4, we removed the pooling layer to prevent the loss of important features.

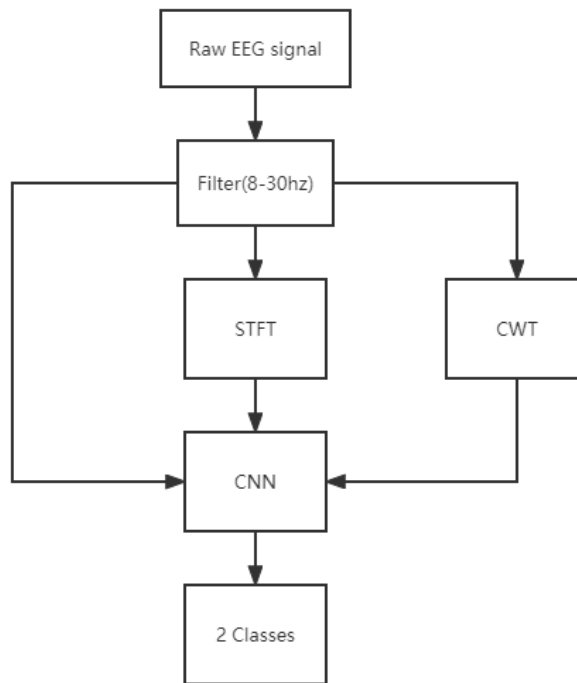


Figure 4.12 The flowchart of the EEG data processing

Layer Number	CWT-CNN	STFT-CNN	Raw-CNN
1	Image input layer	Image input layer	Image input layer
2	Convolution layer ($32 \times 5 \times 5 \times 1$)	Convolution layer ($32 \times 5 \times 5 \times 1$)	Convolution layer ($32 \times 1 \times 8 \times 1$)
3	Batch normalization layer (32 channels)	Batch normalization layer (32 channels)	Batch normalization layer (32 channels)
4	ReLU layer	ReLU layer	ReLU layer
5	Convolution layer ($64 \times 5 \times 5 \times 32$)	Convolution layer ($64 \times 5 \times 5 \times 32$)	Convolution layer ($64 \times 10 \times 1 \times 32$)
6	Batch normalization layer (64 channels)	Batch normalization layer (64 channels)	Batch normalization layer (64 channels)
7	ReLU layer	ReLU layer	ReLU layer
8	Fully connected layer	Fully connected layer	Fully connected layer
9	Softmax layer	Softmax layer	Softmax layer
10	Classification layer	Classification layer	Classification layer

Figure 4.13 The framework of CNN structure

4.2.3 Comparison results of CWT-CNN, STFT-CNN, RAW-CNN

In order to test the effect of different input lengths on the classification, we divided the EEG data into various time windows. The sampling rate of the signal is 250, that is, 1 second contains 250 sampling points. For example, the feature map size of a 3-second window in CWT-CNN is (18, 750), where 18 is the frequency sampling point, including the frequency information of 8-30hz; 750 is the time sampling point, including 3 seconds of time information. We also use overlap technology to increase the number of samples, that is, 50% of the data in the latter segment is the same as that in the previous segment. To ensure random mixing of the training and testing datasets before training and inference to enhance the network's generalization ability, we applied a five-fold cross-validation.

As shown in Figure 4.14, for all three preprocessing types (Raw, STFT, and CWT), the mean accuracy generally increases as the segment size increases. This trend suggests that utilizing longer segments of EEG data may provide more information for classification tasks, leading to improved performance. However, the best accuracy does not consistently increase with segment size for all preprocessing types. For some segment sizes, the best accuracy peaks and then decreases slightly or remains relatively stable (less than a 1%). This suggests that there may be an optimal segment size beyond which further increases do not significantly improve classification performance. The CWT preprocessing type consistently achieves the highest mean and best accuracy among the three preprocessing types across various segment sizes. The STFT

preprocessing type follows closely behind CWT in terms of performance, with slightly lower mean and best accuracy values.

Preprocessing Type	Segment Size (Seconds)	Mean Accuracy	Best Accuracy
Raw	0.25	74.87%	78.30%
	0.5	78.26%	84.00%
	1	83.23%	89.92%
	1.5	83.46%	88.36%
	2	84.33%	89.21%
	2.5	83.25%	89.76%
	3	84.72%	89.17%
STFT	0.25	74.91%	78.12%
	0.5	79.33%	84.92%
	1	85.12%	87.33%
	1.5	84.20%	88.16%
	2	85.72%	88.99%
	2.5	85.39%	89.23%
	3	85.52%	90.21%
CWT	0.25	75.22%	80.54%
	0.5	80.21%	86.06%
	1	85.55%	89.92%
	1.5	85.76%	89.28%
	2	86.33%	91.31%
	2.5	86.25%	91.65%
	3	86.72%	91.23%

Figure 4.14 The comparison performance of different input lengths

The results presented here provide the basis for the fourth thesis point, which is belong to the thesis group 2. The result has been published to Electronics Journal with title “FPGA-Based Hardware Accelerator on Portable Equipment for EEG Signal Patterns Recognition[J1].”

Thesis 4: I have shown that using manually extracted features as input to convolutional neural networks can steadily improve the accuracy of the algorithms in all time windows. Among them, continuous wavelet transform has the best effect, with an accuracy of 91.23%.

4.2.4 Conclusion

In this section, we collected eye EEG data by the OpenBCI device and self-developed software. Then, we introduced the CWT and STFT techniques and compared the classification accuracy of the two preprocessing techniques and different segment sizes. The superior performance of CWT and STFT preprocessing compared

to raw EEG signals highlights the effectiveness of feature extraction in EEG-based classification tasks.

Given the variability in performance across different preprocessing techniques and segment sizes, there may be opportunities to leverage branch learning approaches to further improve classification accuracy. Branch methods can combine multiple CNN models trained with different preprocessing techniques or segment sizes to enhance overall performance and robustness, potentially achieving even higher accuracy than any single model alone. In Chapter 5, we will further study Multibranch CNN architecture.

We will uniformly adopt the 1-second time window segmentation method in the experiments of Chapter 6. Because it offers a good balance of accuracy and latency in three CNNs compared to other time window sizes.

5. Research on EEG Signal Algorithm of Motor Imagery

In Chapter 4, we compared the effects of different preprocessing techniques on accuracy, this established the foundation for algorithm optimization. We considered combining multi-branch technology with previous research. Extending the research on eye state algorithms, this chapter focuses on developing and testing EEG signal algorithms for motor imagery (MI). We believe that MI tasks are more practical in BCI applications. MI tasks involve mentally simulating the movement of body parts without actual physical movement, making them particularly relevant for BCI applications aimed at controlling external devices through thought alone. At the same time, activities such as the BCI competition provide a valuable platform for testing and comparing algorithm performance.

Python has become one of the mainstream high-level programming languages due to its simplicity, rich libraries, and active ecosystem. In BCI-based research, PyTorch, developed by Meta AI, is a widely used open-source deep learning framework that enables dynamic computational graph construction and GPU acceleration. PyTorch is particularly suitable for the development and research of CNNs. It supports seamless integration with Python libraries. In the chapter 5 and 6, data processing will be carried out in the form of Python and Pytorch, where CWT processing uses standard “Scipy” library functions.

In section 5.1, we defined two transfer learning models and evaluated their performance in transferring from eye movement recognition to MI tasks using the PhysioNet dataset. In section 5.2, we proposed Multi-branch CNN architecture based on different frequency bands and test on the Competition IV Dataset 2b [179]. In section 5.3, we proposed Multi-branch CNN architecture based on frequency band and time domain and test on the Competition IV Dataset 2a [179].

5.1 EEG signal recognition based on transfer learning

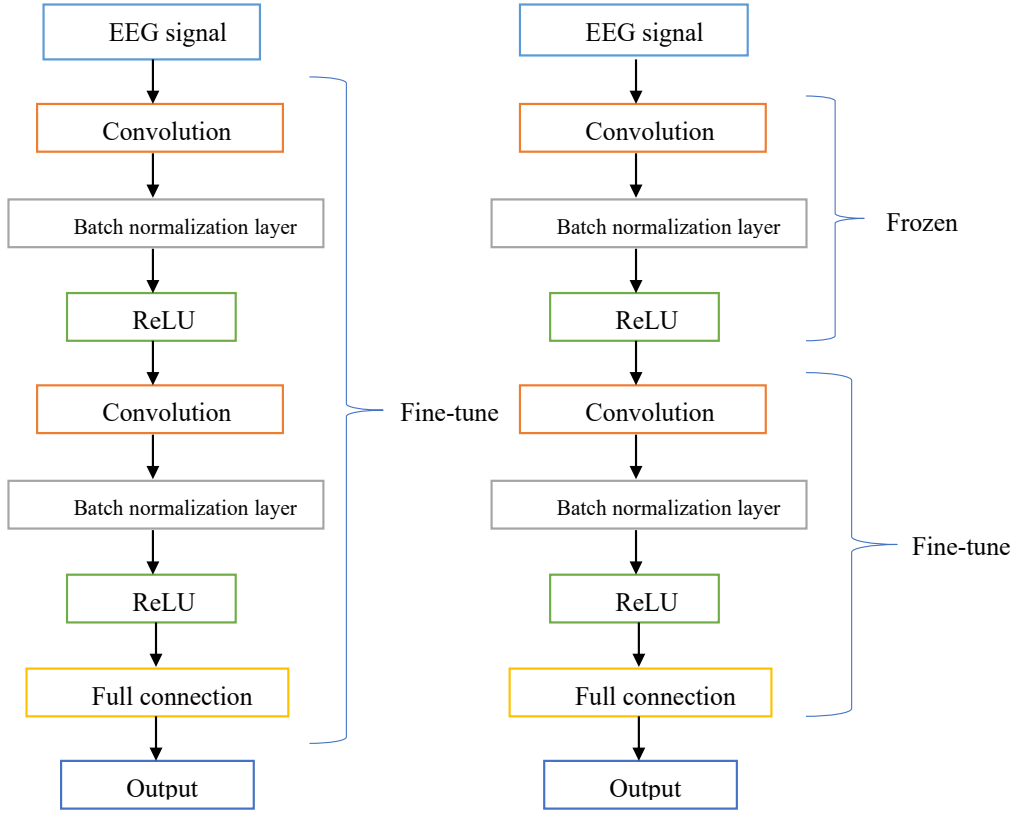
Transfer learning is a method of pre-training a network using prior knowledge and then fine-tuning it. It has been applied to many fields to solve new tasks, such as translation, data mining, and computer vision [180, 181]. Because the model has learned knowledge from other tasks, the transfer model can still maintain accuracy

with only a small amount of data. MI data requires subjects to concentrate and repeat specific brain activities many times, which can easily lead to experimental tiredness and unstable signal quality. Secondly, due to significant individual differences, the collected MI data often requires large-scale annotation to meet the training requirements of CNN, which are time-consuming and costly. To solve these problems, this study proposed a method based on transfer learning, which uses relatively simple and easy-to-collect eye state data to assist in training the model, and then transfer it to the more complex motor imagery EEG signal recognition task.

5.1.1 Fine-tuning transfer learning model

In Section 4.2, we demonstrated that the CWT-CNN achieved the best accuracy. In this subsection, we will use the CWT-CNN mentioned in Section 4.2 as a pretrained model and then fine-tune it. The structure of the CWT-CNN is shown in Figure 4.13. Fine-tuning a network typically involves two approaches: fine-tuning the entire network or freezing certain layers while fine-tuning others[182]. The frozen layers are usually the earlier layers, which capture general features with good transferability. In the subsequent experiments, we will adopt both fine-tuning approaches for comparison. CNN1 represents fine-tuning all layers, while CNN2 represents fine-tuning only certain layers, as shown in the Figure 5.1.

The dataset used in this study is the PhysioNet dataset mentioned in Section 4.1. We will create two datasets: the dataset 1 consists of eye movement data, specifically trials 1 and 2. The dataset 2 consists of data related to imagining the fist and relax of the left and right hands, specifically trials 4, 8, and 12. All data from the dataset1 will be used for pretraining the CNN network. The dataset 2 will be divided into two groups, with one-tenth of the data used as the validation set, and the remaining data used for fine-tuning the CNN in proportion to the training ratios.



CNN1: Fine-tune the whole net

CNN2: Fine-tune certain net

Figure 5.1 The structure of transfer learning model

5.1.2 Performance of transfer learning model

The experiment environment is:

CPU: Intel(R) Core (TM) i5-7300HQ @2.50 GHz,

GPU: NVIDIA GeForce GTX 1050 Ti,

Operating System: Windows 10 64 bits,

Python Version: 3.6.

Because of the annotation errors, the subjects of 88, 89, 92, 100 and 104 were removed from the experiment. The hyperparameters of CNN training methods are consistent with those in section 4.2.2, as follows:

- Batch Size: 64 per iteration.
- Learning rate: The learning rate was reduced by a factor of 0.05 every 10 epochs, starting with an initial learning rate of 0.2.
- Number of epochs: The maximum number of training epochs was set to 500.
- Regularization Techniques: L2 regularization was 0.0005.

The purpose and description of each hyperparameter are detailed in Section 4.1.4.

In the Table 5.1, training data ratio represents the proportion of Dataset 2 used for training the network. A ratio of 100% indicates that all available data is used for training. CNN1 represents fine-tuning all layers, CNN2 represents fine-tuning specific layers, and CNN3 represents training from scratch without using a pretrained model. After adding training data at each step, the network is trained starting from its pretrained state, and each network is trained 10 times to obtain the average accuracy.

Table 5.1 Comparison of Transfer Learning model

Training data ratio	CNN1(All layers)	CNN2(Some layers)	CNN3(Scratch)
10%	65.2%	63.1%	54.7%
20%	70.5%	68.2%	61.3%
30%	76.4%	73.5%	66.2%
40%	79.3%	77.9%	72.7%
50%	81.5%	80.3%	73.4%
60%	83.1%	81.4%	75.5%
70%	83.8%	82.1%	79.7%
80%	84.5%	82.8%	83.2%
90%	84.9%	83.0%	85.3%
100%	85.1%	82.9%	85.4%

The experimental results show that the accuracy of CNN1 and CNN2 increases rapidly when the data volume is 10%-50%, indicating that the pre-trained features are fully utilized. However, as the data volume increases to more than 70%, the

performance improvement rate of the transfer learning model decreases, which mean the features learned from the source task have limited effect under large data volumes. Compared with CNN1 and CNN2, CNN1 always maintains its advantage, indicating that fine-tuning all layers can make more full use of the features learned from the source task to adapt to the target task.

In contrast, CNN3 performs unsatisfactorily when the data volume is small, but its feature learning ability gradually increases with the increase of data volume. At a data volume of 90%-100%, the accuracy of CNN3 surpasses the transfer learning model and finally reaches the highest level. This phenomenon shows that training from scratch can learn MI-EEG specific features more stably when there is sufficient data.

5.2 Multi-branch CNN architecture based on different frequency bands

5.2.1 BCI Competition IV Dataset 2b

As one of the famous public MI datasets, 2008 BCI Competition IV Data Set 2b provides reliable data for EEG researchers. On the other hand, choosing this dataset can also better evaluate the effectiveness of our research. This dataset collected motor imagery signals from the left and right hands of nine volunteers and only contained data from three bipolar electrodes (C3, Cz, and C4). All data are divided into the training section and validation section.

The EEG acquisition process illustrated on the Figure 5.2. Initially, directional arrows appear on the screen, accompanied by an audible alert to remind the volunteers to start preparing for the experiment. This process lasts for 3.5 seconds. For the next 3.5 to 7 seconds, the volunteers imagined moving their left or right hands according to the direction of the arrow on the screen. After this, the subjects rested and waited for the next arrow and sound cue to appear. Data collection time ranged from 4 to 7 seconds per round to prevent participants from delaying their responses to the prompts, thereby collecting irrelevant data. This process yielded a total of 750 data points per round. Each raw data segment is formatted as 750×3 , where 750 represents 3 seconds of signal points (sampled at 250 Hz), and 3 corresponds to the number of electrodes used (C3, Cz, and C4). The time-frequency diagrams of the alpha and beta bands are shown in the Figure 5.3. We transform the data of the C3, Cz, and C4 channels through

CWT to obtain a time-frequency image of size (40, 750). We will average sample every 5 points along the time axis, and finally, the size of a sample is (40, 150, 3).

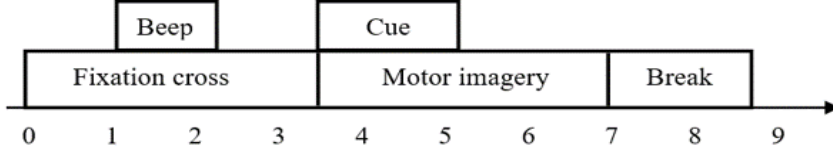


Figure 5.2 Timing scheme of the paradigm.

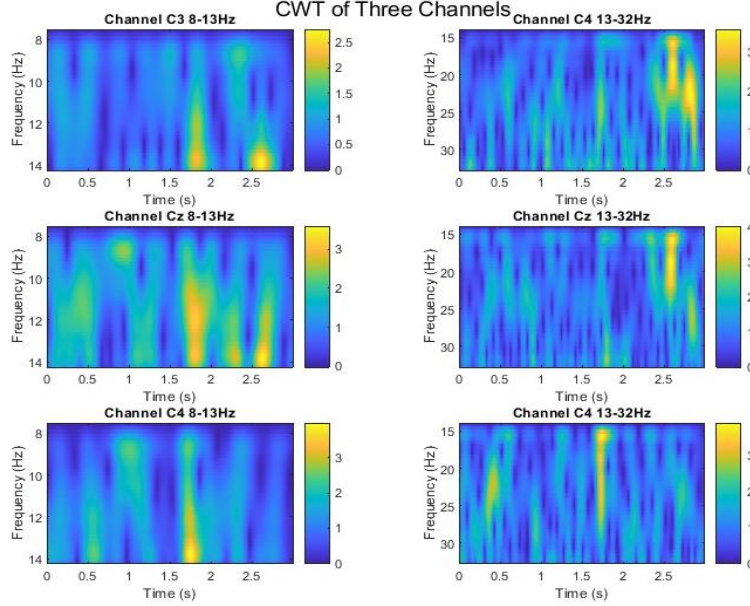


Figure 5.3 Three seconds CWT maps.

5.2.2 Proposed Multi-branch CNN Structure

There are already predecessors of multi-branch CNN architecture. Paper [183] proposed a Three-branch CNN. This approach involves combining three distinct frequency bands as input. Each frequency band is assigned to a corresponding convolutional layer, each with distinct convolutional kernel scales. This method focuses on extracting information from different frequency bands. Paper [184] introduces a Four-branch CNN model. They used four different time-domain images as inputs to effectively extract features.

The previous researchers mentioned above only focused on extracting features from a single domain (time domain or frequency domain). However, MI images are continuous and complex and contain features from both the time and frequency

domains. Therefore, we developed a two-branch time-frequency CNN architecture, hoping to extract more useful information from MI images. Our CNN is shown in Figure 5.4. The proposed CNN consists of two separate CNNs. The original data is mapped into two time-frequency feature maps through CWT, containing the frequency band information of 8-14 Hz and 15-30 Hz respectively. Then they are used as input to the two branch CNNs respectively. The two branches adopt the same structure, consisting of two convolutional layers, each layer contains 32 and 64 convolutional kernels of 5x5 size with a stride of 2x2 respectively. The feature maps generated by the two minor CNN branches are input to a fully connected layer, transforming them into 1-D vectors. These 1-D features from both branches are then combined into a single 1-D vector, which serves as input for the classifier. We utilized three popular classifiers: decision tree (DT), SoftMax, and support vector machine (SVM). Also, the pooling layer was removed to simplify the model architecture and avoid the loss of sampling data.

The detailed hyperparameter information is as follows (see section 4.1.4 and 4.2.3 for details on the role of hyperparameters):

- **Activation function:** ReLU
- **Optimizer:** Adam
- **Batch Size:** 64
- **Learning rate:** Initially 0.1 with an automatically adjusted function
- **Number of epochs:** 100
- **Regularization Techniques:** 0.01
- **Early stop:** On.

The input image signals were trained and tested using the proposed CNN model with 10-fold cross-validation.

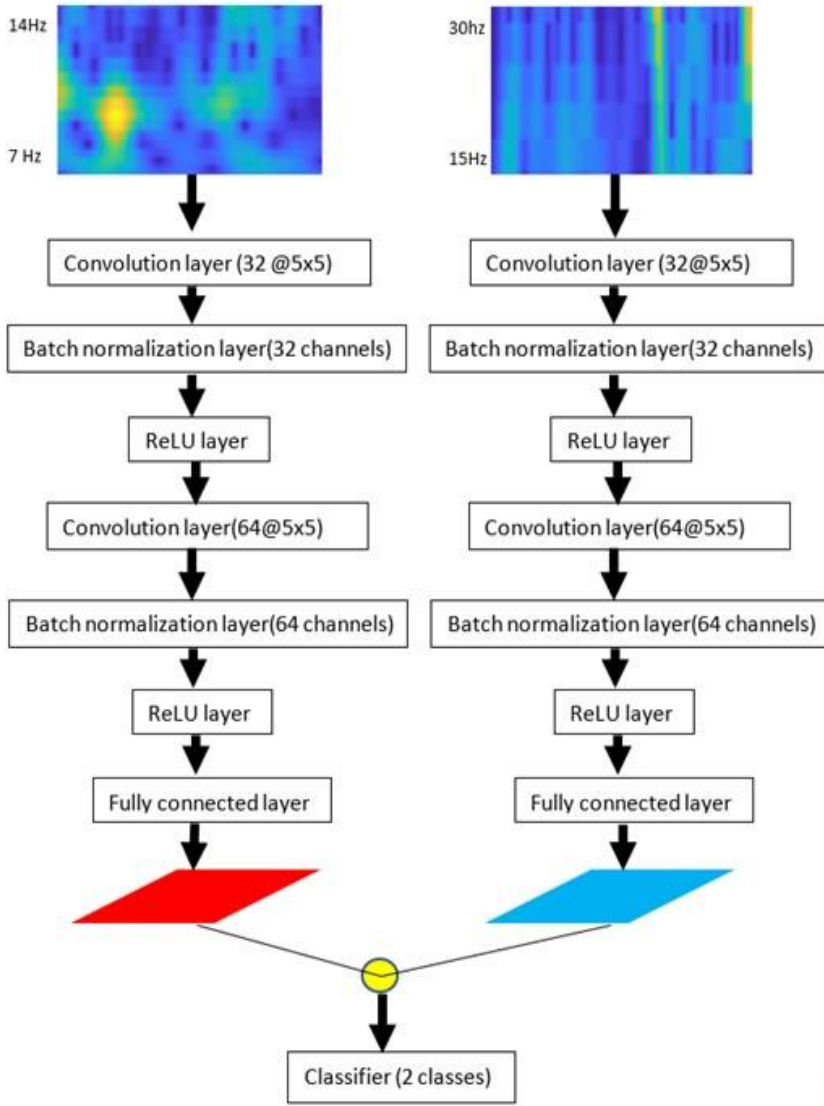


Figure 5.4 The proposed CNN framework

5.2.3 Performance of the Proposed Model

We outline several practical metrics employed to evaluate the effectiveness of our approach. The Accuracy is calculated as Equation 5.1.

$$Accuracy = \frac{T_p + T_N}{T_p + T_N + F_p + F_N}, Accuracy \in R[0,1] \quad (5.1)$$

where T_p , T_N , F_p and F_N represent true positives, true negatives, false positives and false negatives, respectively.

Precision, Recall, and F1 score, are determined as:

$$Precision = \frac{T_p}{T_p + F_p} \quad (5.2)$$

$$Recall = \frac{T_p}{T_p + F_N}, a \in R[0,1] \quad (5.3)$$

$$F1 = 2 * \frac{Recall + Precision}{Recall * Precision}, a \in R[0,1] \quad (5.4)$$

The experiment environment is:

CPU: Intel(R) Core (TM) i5-7300HQ 2.50 GHz,

GPU: NVIDIA GeForce GTX 1050 Ti,

Operating System: Windows 10 64 bits,

Python Version: 3.6.

Figure 5.5 presents the comparative outcomes of three classifiers. SVM reaches the best accuracy of 84.1%. Softmax exhibited an accuracy of 80.6, 3.5% lower than SVM. DT only reached 75.3%. EEG signals are often affected by considerable noise and interference, but CNN can effectively extract relevant data through convolution layers. These layers enable the abstraction of feature representations from classification tasks. In contrast, decision trees may be limited to utilizing original features or a restricted set of manual features, leading to comparatively weaker expressive capabilities [45]. Furthermore, we showcase the confusion matrices, Receiver operating characteristic curve (ROC) and the comprehensive results of precision and recall values on Figure 5.7, Figure 5.6 and Table 5.2 respectively. These optional evaluation metrics are also employed to indicate the efficacy of the proposed model.

One disadvantage of this dataset is that the number of samples is small, and the number of sampling channels is only 3. This is not friendly to deep models that require big data training. On the other hand, our study also proves that the traditional method SVM still has strong classification ability in this small dataset. In section 5.2 investigations, we intend to mitigate this limitation by employing suitable data augmentation techniques to expand the dataset size.

subject	Accuracy (%)		
	DT	Softmax	SVM
S1	71.2	77.8	75.1
S2	60.7	65.4	69.8
S3	66.2	70.7	70.5
S4	89.5	97.1	92.6
S5	85.2	86.4	91.7
S6	80.4	72.1	87.2
S7	70.1	80.5	81.5
S8	73.9	89.6	93.3
S9	80.5	86.2	95.2
Average	75.3	80.6	84.1

Figure 5.5 Comparison of our work and different classifiers.

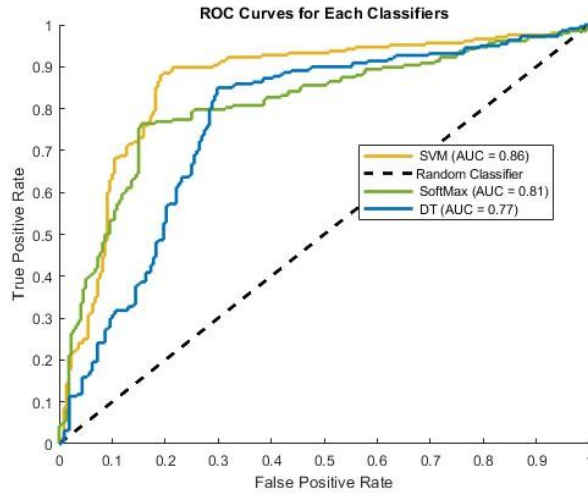


Figure 5.6 ROC for three classifiers.

Table 5.2 Performance Metrics in our CNN

Metrics	Our
Precision (%)	85.71
Recall (%)	81.41
F1-Score (%)	83.5

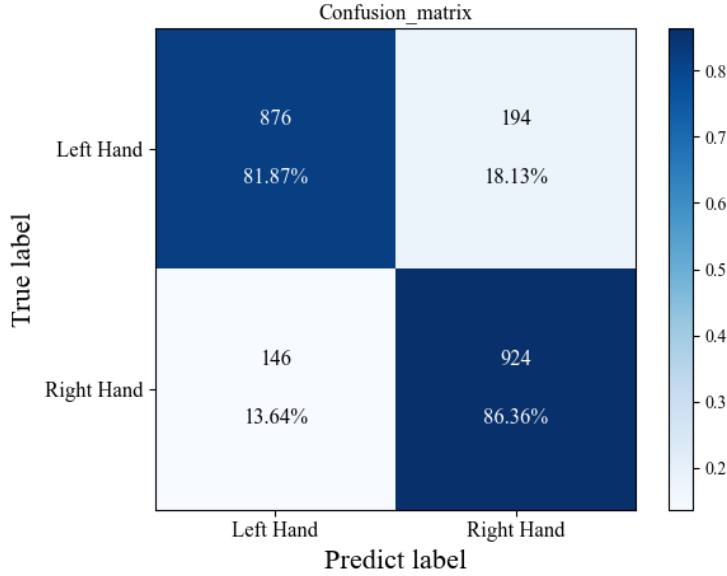


Figure 5.7 Confusion matrix for the proposed model with SVM

To demonstrate the effectiveness of the proposed approach, we conducted a comparative analysis with several other models. Paper [2] employs Conditional empirical mode decomposition (CEMD) preprocessing, ERS/ERD encoding, and a 1D multi-scale CNN model for effective classification of EEG signals related to MI tasks. Paper [185] introduces a Covariate Shift Estimation and Unsupervised Adaptive Ensemble Learning (CSE-UAEL) method. It integrates covariate shift detection with dynamic ensemble updating and adds classifiers based on detected shifts to address non-stationarity in EEG-based BCI systems. Paper [186] extracted features from time–frequency energy, maximum entropy power spectral estimation, and Hjorth parameters by sparse representation (SR). Paper [183] decompose the raw signal into signals in three frequency bands, and input them to three small CNNs with the same parameters. Paper [187] introduces an one convolutional layer CNN with 30x30 convolution kernel. Paper [188] designs a two-branch CNN network (TBTF-CNN) to extract features from

both the raw EEG signals and CWT map. The Contribution to the comparative literature is summarized in Table 5.3.

Table 5.3 Contribution to the comparative literature

Literature	Contribution	Hardware complexity
[2]	● Conditional empirical mode decomposition (CEMD) ● 1D multi-scale CNN: consists of three 1D CNN layers of different sizes	● CEMD involves iterative processes that do not fully leverage hardware feature
[185]	● Covariate shift estimation (CSE) ● Unsupervised adaptive ensemble learning (UAEL)	● Requires moderate computational resources ● Integrating multiple classifiers increases memory requirement
[186]	● Multi-domain feature extraction and sparse feature fusion (MFESF), such as the power spectrum estimation, time-frequency energy	● Mentioned Feature extraction techniques require relatively low hardware implementation complexity
[183]	● Temporal two 1D Convolutional layers CNN: three different kernel sizes (1x45, 1x65, 1x85) ● Data Augmentation (DA)	● 1D structure reduces computational parameters ● 9 branches increase implementation complexity
[187]	● CWT ● 30×30 kernel CNN	● 30×30 kernel significantly increases computational complexity

[188]	● Two-Branch CNN fusing temporal and frequency features	● Single Convolutional Layer: Facilitates hardware implementation ● 64x1 Kernel: Too large for efficient hardware implementation
Our	● CWT ● Two-Branch lightweight CNN	● 5x5 kernel CNN small kernel good for reduce hardware memory and computing resource usage ● Many hardware acceleration techniques for CNNs and CWTs

The comparative performance outcomes are outlined in Table 5.4. Our method outperforms many current state-of-the-art models. The mean accuracy of our model is 84.1%, with 25.4% accuracies gap, from 69.8% to 95.2%. Additionally, for subjects 6 and 9, we reach the best accuracy among all the models, which are 91.7% and 95.2%, respectively. Among them, DA-CNN has the highest average accuracy of 87.6%. We note that it uses data augmentation technology to increase the number of training samples by three times. For deep learning networks, increasing the sample size is a proven method to improve training accuracy and is widely practiced in engineering. However, since EEG data is biological, we must carefully validate the use and reliability of artificially generated biological data. The applicability of conventional data enhancement methods, such as inversion, resizing, and noise addition, to biological data needs further exploration. Therefore, in section 5.2.2, we focus on developing data augmentation methods suitable for EEG data that meet both engineering and biological standards to optimize our model's performance. Notably, individual subject analysis shows varied accuracy to different models, highlighting the impact of individual variability on model efficacy. Our method exhibits minimal accuracy discrepancies among subjects compared to other techniques. Although CWT

and CNN have been applied to the MI field in prior research, our network still outperforms models employing similar CWT-CNN technology in terms of accuracy.

Table 5.4 Compare the result with advanced CNNs

Accuracy (%)							
Subject No.	CWT-CNN			Other Method			
	Our	[187]	[188]	[2]	[185]	[186]	[183]
1	75.1	84.5	85.6	80.56	78.13	75.3	80.5
2	69.8	63.3	72.8	65.44	54.69	65.4	70.6
3	70.5	62.3	78.0	65.97	53.13	62.2	85.6
4	92.6	98.1	95.4	99.32	94.38	97.8	94.6
5	87.2	89.7	82.6	89.19	85.31	88.8	86.6
6	91.7	85.0	79.8	86.11	80.31	87.2	86.6
7	81.5	79.5	82.9	81.25	72.81	73.4	89.6
8	93.3	84.7	85.0	88.82	78.75	91.9	95.6
9	95.2	84.5	85.3	86.81	74.38	87.5	87.4
Mean	84.1	81.3	83.0	82.61	74.65	81.1	87.6

The classification accuracy using the feature maps mapped by CSP, FFT, STFT, and CWT as input is shown on Table 5.5. Among the evaluated technology, the CWT-CNN achieves the best performance, outperforming the CSP-CNN by 4.1%, the FFT-CNN by 4.5%, and the STFT-CNN by 2.2%. As mentioned in section 2.2.4, CSP requires multiple electrodes to avoid missing short-term signal variations. However, the dataset used in this section only contains two electrodes. Additionally, FFT has difficulty effectively capturing the local features of MI-EEG signals. The fixed window size of STFT can result in an insufficient representation of both global and local features. Conversely, CWT provides a balance between global and local features by decomposing the signal and offering time-varying windows with high temporal

resolution. Although the CWT-CNN achieves the highest average accuracy overall, for subjects 4, 7, and 9, its accuracy is lower than that of other models.

This section presents a dual-branch lightweight CNN. Additionally, we examine the impact of various preprocessing methods (STFT, CSP) and classifiers (SVM, DT) on the branch structure. The results demonstrate that the CWT-CNN architecture is more effective for MI-related tasks, achieving an average accuracy of 84.1% on the BCI Competition IV 2B dataset.

Table 5.5 Comparison of the result obtained with different input

Accuracy (%)				
Subject No.	CSP-CNN	FFT-CNN	STFT-CNN	CWT-CNN
1	65.9	68.2	73.5	75.1
2	65.1	63.5	68.1	69.8
3	65.2	67.9	65.2	70.5
4	91.8	92.1	93.1	92.6
5	81.7	84.2	85.7	91.7
6	80.3	78.4	84.3	87.2
7	82.1	73.7	78.1	81.5
8	92.4	90.0	91.9	93.3
9	95.3	90.1	92.7	95.2
Mean	79.9	79.6	81.9	84.1

5.3 Multi-branch CNN architecture based on frequency band and time domain

5.3.1 BCI Competition IV Data Set 2a

In 5.2 section, we chosen BCI Competition IV Data Set 2a [179] for analysis. Each subject performed four different MI tasks: imagining movement of the left hand, imagining movement of the right hand, imagining movement of both feet, imagining movement of the tongue. The EEG signals were recorded using 22 Ag/AgCl electrodes placed according to the international 10-20 system. Each subject participated in two sessions on different days. Each session contains six runs. The EEG acquisition process

illustrated on the Figure 5.8. Similar to BCI Competition IV Data Set 2b, directional arrows appear on the screen, accompanied by an audible alert to remind the volunteers to start preparing for the experiment. This process lasts for 3 seconds. For the next 3 to 6 seconds, the volunteers imagined the motor according to the prompts on the screen. After this, the subjects rested and waited for the next arrow and sound cue to appear. Data collection time ranged from 3.5 to 6 seconds per round to prevent participants from delaying their responses to the prompts, thereby collecting irrelevant data. This process yielded a total of 625 data points per round. Each raw data segment is formatted as 625×22 , where 625 represents 2.5 seconds of signal points (sampled at 250 Hz), and 22 corresponds to the number of electrodes used. Each volunteer included 576 sample of experiments

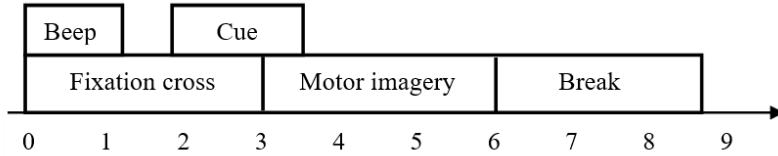


Figure 5.8 Timing scheme of the paradigm

5.3.2 Data Augmentation (DA)

In section 5.2, we mentioned the generalization problem of SoftMax caused by too small data sets. In this section, we try to solve this problem by increasing the number of samples. In the field of computer vision, there are already some common DA methods, such as rotation, cropping, deformation, and scaling. Based on our previous analysis of the basic properties of EEG signals, we can find that EEG signals exhibit different characteristics in both the time domain and the frequency domain. Therefore, simple data enhancement methods may not be applicable to EEG signals. Current EEG signal data augmentation techniques can be broadly classified into two main categories: data transformation [189-191] and noise addition [192, 193]. Data transformation involves modifying the existing signals to create new samples. This can include techniques such as time-warping, window slicing, or signal filtering. These methods aim to simulate variations in the signal that might naturally occur, enhancing the model's ability to generalize from the training data. Noise addition, on the other hand, involves incorporating various types of noise into the EEG signals to make the model more robust to real-world conditions. This can include adding Gaussian noise, salt-and-pepper noise, or other perturbations that mimic potential interferences during

signal acquisition. By employing these DA techniques, the dataset can be effectively expanded, thus improving the training process, and boosting the performance and generalization capabilities of models designed to work with EEG data.

In this study, the DA algorithm operates data in both the time and frequency domains, aiming to preserve the intrinsic characteristics of the original EEG signals to the fullest extent. The proposed method involves two primary steps: time domain transformation and frequency domain transformation, as illustrated in Figure 5.9 and Figure 5.10, respectively.

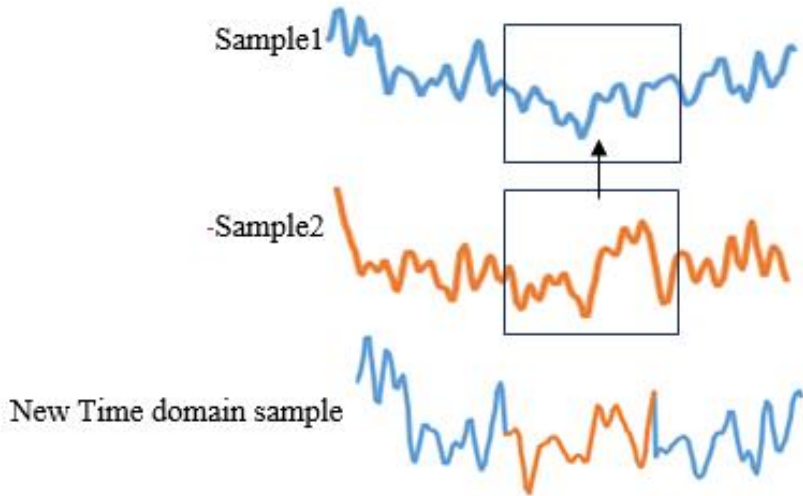


Figure 5.9 An example of Time domain sample generation

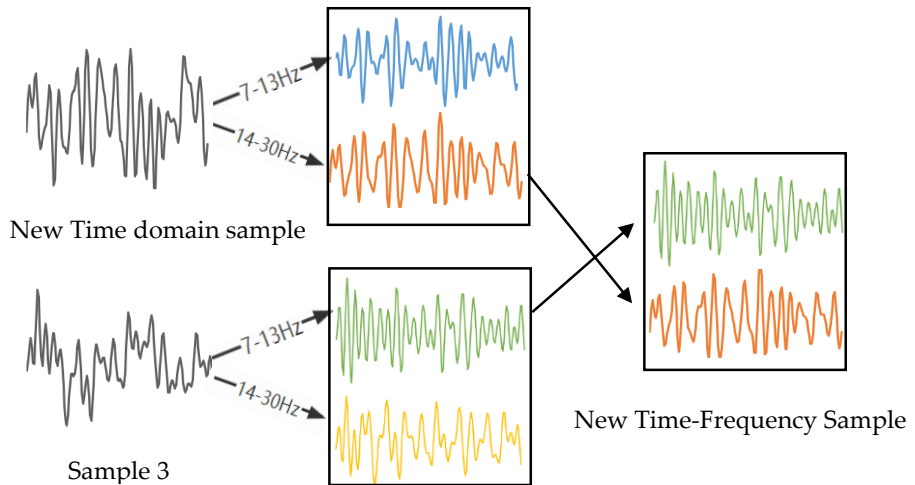


Figure 5.10 An example of Time-Frequency domain sample generation.

We randomly selected three samples from the same category and named them sample 1, sample 2, and sample 3.

- A segment of 1 second from sample 2 is randomly chosen and substituted for the data at the corresponding time position in sample 1 to generate a new sample, namely new time domain sample.
- All samples were band-pass filtered at 7-13 Hz and 14-30 Hz, respectively. Subsequently, the frequency band from sample 3 is interchanged with the corresponding frequency band of the new time domain sample to reconstruct the artificial time-frequency EEG signal.

Our DA process was iterated 1000 times to generate 1000 fresh samples. Consequently, the total number of samples for each subject rose from the initial 576 to 1576.

5.3.3 Multibranch Convolutional Networks

We optimized the two-branch CNN mentioned in section 5.1. As depicted in Figure 5.11, the input to the left segment of the model comprises a time-domain image. The initial and secondary convolutional layers of this segment are 1-D convolutional layers with kernel sizes of 1x3 and 10x1, respectively, employing a stride step size of 2x1 for both. These segments aim to capture spatial information and temporal features among each channel. On the other hand, the input to the right segment of the model is the image post CWT transformation. Here, the two convolutional layers consist of 32 and 64 convolutional kernels, each with a size of 5x5 and a stride step size of 2x2. Following this, the 2-D feature maps obtained from both segments are transformed into a 1-D vector through a fully connected layer. Subsequently, all 1-D features from the two branches are concatenated into a single 1-D vector, serving as the input for the classifier. The Softmax function will be the classifier in this section. Also, the pooling layer was removed to simplify the model architecture and avoid the loss of sampling data.

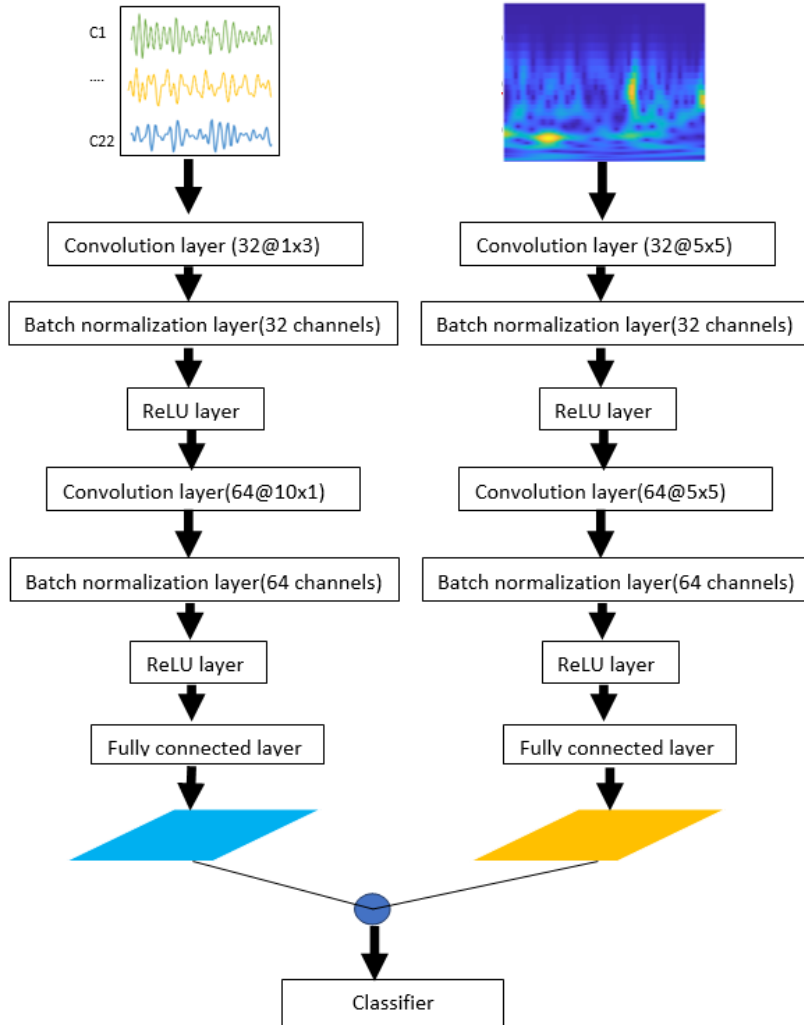


Figure 5.11 The proposed CNN framework.

The detailed hyperparameter information is as follows (see section 4.1.4 and 4.2.3 for details on the role of hyperparameters):

- **Activation function:** ReLU
- **Optimizer:** Adam
- **Batch Size:** 64
- **Learning rate:** Initially 0.1 with an automatically adjusted function
- **Number of epochs:** 100
- **Regularization Techniques:** 0.01
- **Early stop:** On.

5.3.4 Performance of DA and Multi-branch CNN

The spectral power comparison plots pre- and post-signal augmentation of C3, Cz, C4 channels were displayed on Figure 5.12 and Figure 5.13. A comparison shows that the original EEG signal's characteristics are basely preserved pre- and post-augmentation. Our proposed DA method emphasizes augmenting the energy of the substituted band while maintaining a relatively consistent spectral ratio between the signal and its replaced band.

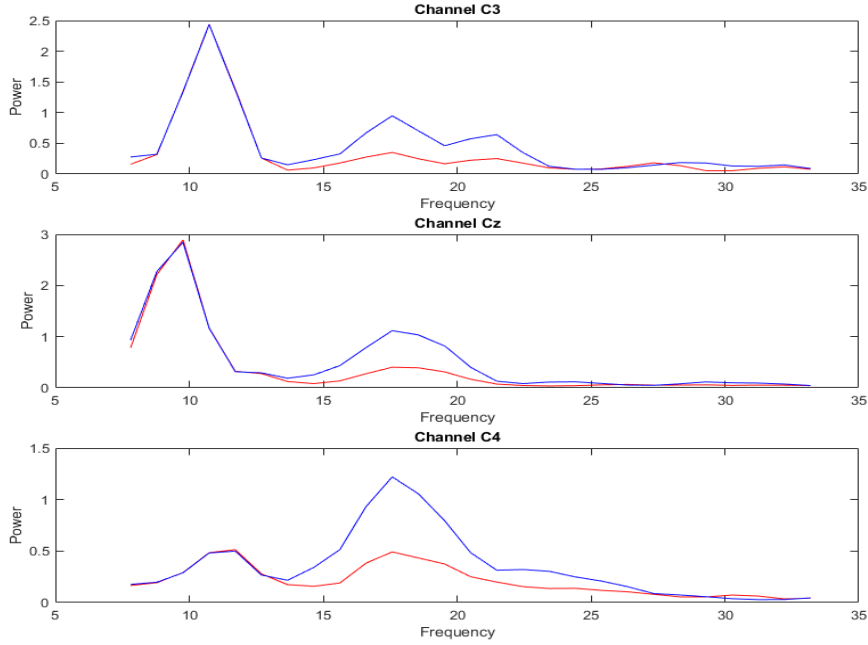


Figure 5.12 An example of left-hand motor imagery where the beta band is displaced. The red line represents the spectrum of the original data, while the blue line represents the spectrum post-augmentation.

In addition to the DA technology mentioned above, we also used other methods to increase the number of samples, such as 50% overlap and segmentation. In order to study the impact of segment size on accuracy, we divided the data into four segments: 0.5 seconds, 1 second, 2 seconds, and 3 seconds. Figure 5.14 illustrates the accuracy comparison across different segment sizes, showing that the 0.5s window exhibits the lowest classification accuracy, while the 2s and 3s windows showcase more comparable accuracy. The 2s segment data already include enough MI information. Consequently, all subsequent experiments adopt a 2s time window to optimize resource utilization.

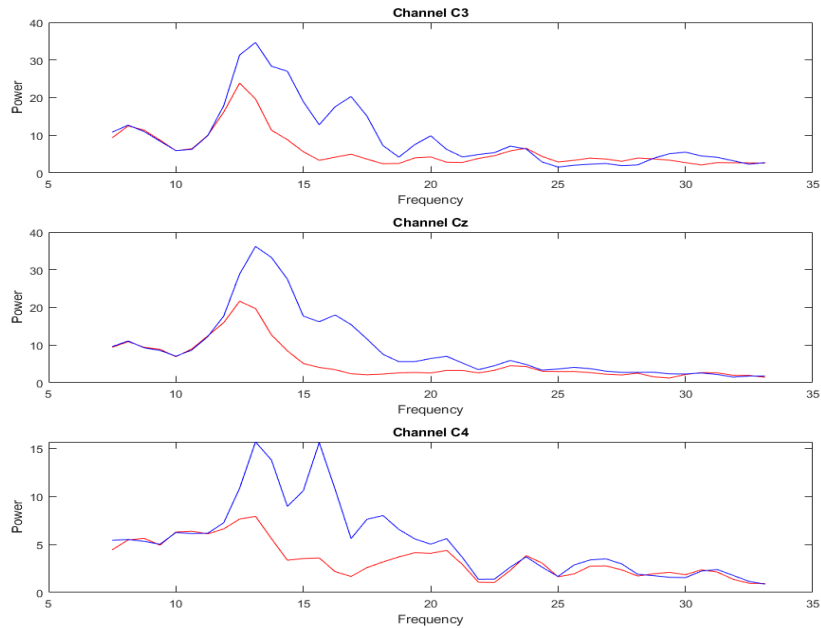


Figure 5.13 An example of left-hand motor imagery where the alpha band is displaced. The red line represents the spectrum of the original data, while the blue line represents the spectrum post-augmentation.

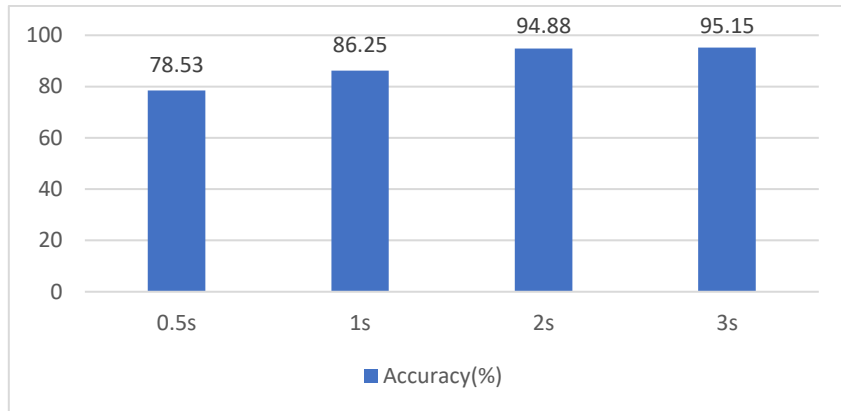


Figure 5.14 Comparison of different length segments

Table 5.6 provides a comparative synthesis of the reviewed literature. We have reviewed the literature and provided an overview of their methodologies and contributions.

Table 5.6 Contributions of the reviewed works

literature	Classifier	Contribution
VGGNet[194]	CNN	- Using VGGNet for MI dataset training. - Better efficiency and accuracy compared to SVM, ANN and traditional CNN.
EEGNet[168]	CNN	- Introduce EEGNet, a compact convolutional neural network for EEG-based BCIs. - Demonstrate three different approaches to visualize the contents of a trained EEGNet model to enable interpretation of the learned features.
HS-CNN[183]	CNN	- Different convolution kernel scales are designed for EEG signals in different frequency bands, and convolution operations are performed in the temporal and spatial dimensions using one-dimensional convolution kernels. - Identically labeled samples are segmented and then swapped and recombined, and then divided by frequency bands, and different segments of the same frequency band are swapped and recombined.
PSO-CNN[195]	CNN	- The signal is input to the CSP filter, while the PSD of different frequency bands is calculated and the particle swarm optimization algorithm is introduced for feature optimization. - CNN has only one convolutional layer.
CNN-LSTM[196]	CNN+LSTM	- Using the Filter Bank Common Spatial Pattern technique - Hybrid network to extract multimodal features.

DWT-CNN[197]	CNN	- Sliding window for data enhancement, calculating PSD for optimal frequency bands. - Decompose 3×3 convolution kernels into 4×1 or 2×1.
--------------	-----	---

To further investigate the capabilities of our proposed DA, two popular models (VGGNet and EEGNet) are reproduced and retrained with the same dataset. Table 5.6 presents the comparison of the results. The mean accuracy is reached to 97.61%. It is worth noting that this does not only refer to the average accuracy but is the highest for each subject. Additionally, our DA method shows significant accuracy improvements for all models, with average enhancements of up to 4.41% and 11.15% for individual subjects.

Table 5.7 Compared results with EEGNet and VGGNet, marked with the suffix (_A) indicate the application of DA techniques

Subject No.	Acc(%)					
	VGGNet	VGGNet_A	EEGNet	EEGNet_A	Our	Our_A
1	79.32	85.36	88.21	91.33	97.23	98.61
2	69.01	77.22	70.35	72.58	94.77	98.29
3	91.11	92.67	92.40	93.14	94.12	97.99
4	68.32	66.12	81.26	85.65	95.22	97.1
5	59.00	70.15	85.77	83.85	93.25	96.72
6	70.32	71.39	73.12	76.98	92.36	96.37
7	82.55	85.73	93.10	98.54	94.8	97.69
8	79.33	88.52	90.73	89.32	95.75	97.22
9	70.74	72.29	91.05	89.99	96.38	98.51
Mean	74.41	78.82	85.11	86.82	94.88	97.61

The results presented here provide the basis for the fifth thesis point which is belong to the thesis group 2. The result has been published to Sensors Journal with title “Classification of Motor Imagery EEG Signals Based on Data Augmentation and Convolutional Neural Networks.[J3]“

Thesis 5: I demonstrated that our data augmentation method shows significant accuracy improvements for multiple models, including VGGNet, EEGNet, and two branch CNN architectures, with average enhancements of up to 4.41% and 11.15% for individual subjects.

Table 5.7 Comparison of the proposed method with state-of-the-art methods. presented the comparison with more studies [195-198]. It is apparent that our method achieves the best classification accuracy among state-of-the-art techniques. The proposed approach demonstrates significant accuracy improvements, particularly for subject 2 (an increase of up to 18.01%) and subject 6 (an increase of up to 19.45%). Furthermore, comparing Table 5.7 and Table 5.8, our method exhibits consistent classification performance across 9 subjects, with a maximum difference of only 5% between subjects, which is 32% in VGGNet and 12% in DWT-CNN. Our method effectively mitigates the impact of individual EEG signal variability and can autonomously learn the unique EEG signal activity patterns of different subjects across four imaginary states.

Table 5.7 Comparison of the proposed method with state-of-the-art methods.

Subject No.	Acc(%)					
	HS-CNN	PSO-CNN	CNN-LSTM	DWT-CNN	Our	Our_A
1	90.07	93.30	98.82	98	97.23	98.61
2	80.28	84.59	98.64	98	94.77	98.29
3	97.08	91.68	96.92	95	94.12	97.99
4	89.66	84.55	96.50	96	95.22	97.1
5	97.04	86.54	92.75	86	93.25	96.72
6	87.04	76.92	91.84	92	92.36	96.37
7	92.14	94.03	95.07	94	94.8	97.69
8	98.51	93.20	95.25	93	95.75	97.22
9	92.31	92.24	99.23	98	96.38	98.51
Mean	91.57	85.56	96.13	94	94.88	97.61

Additional, various metrics can be employed to evaluate the effectiveness of our work. Figure 5.15 and Figure 5.16 show the confusion matrices of our binary branch model before and after using DA technology. Table 5.8 provides an exhaustive analysis of the Precision and Recall values.

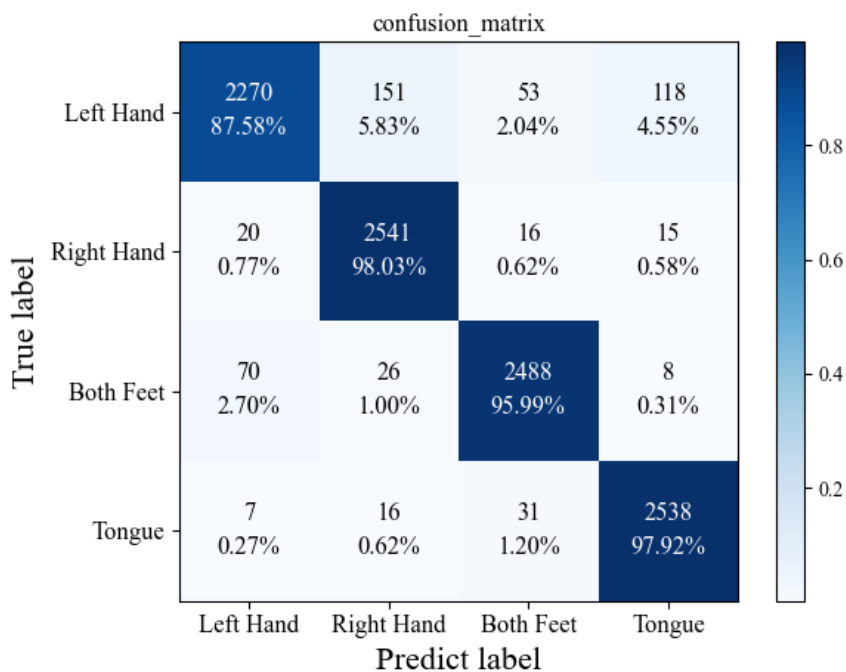


Figure 5.15 Confusion matrix for our binary branch CNN

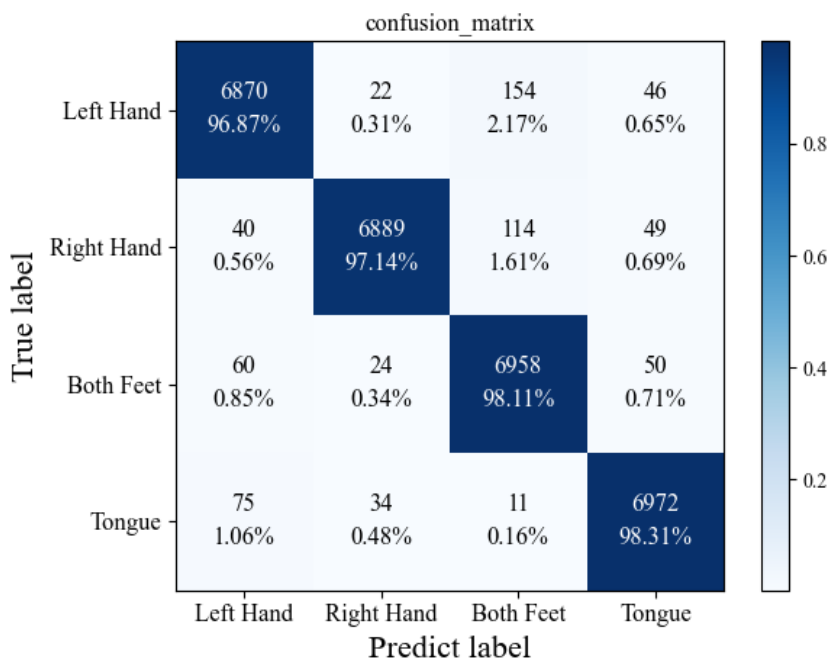


Figure 5.16 Confusion matrix for the binary branch CNN with DA.

Table 5.8 Precision and recall scores for our binary branch model.

Metrics	Our	Our_A
Precision(%)	94.93	97.62
Recall(%)	94.88	97.61
F1-Score	0.95	0.98

The results presented in this chapter provide the basis for the sixth thesis point. The result has been published to Sensors Journal with title “Classification of Motor Imagery EEG Signals Based on Data Augmentation and Convolutional Neural Networks[J3],” and “Enhancing Motor Imagery Classification in Brain–Computer Interfaces Using Deep Learning and Continuous Wavelet Transform [J4].”

Thesis 6: I demonstrated that the proposed two-branch CNN structure achieves the 97.61% accuracy, which is the best classification accuracy in comparison with popular EEG models (VGGNet and EEGNet) and other state of art models. Meanwhile, support vector machine with convolutional neural networks is the best classification method with a small amount of training sample data, outperforming decision trees and Softmax.

5.4 Conclusion

Due to the complexity and significant individual differences of EEG signals, achieving high automatic classification accuracy is challenging, especially when training data size is limited. Another challenge is that existing CNN-based EEG MI classification methods typically rely on a single domain to extract EEG features. This results in limited classification effectiveness because it necessitates decoding the time, frequency, and spatial information of the EEG signal simultaneously. To address this challenge, we introduced two parallel structures that utilize CNN and CWT techniques in Section 5 to achieve more comprehensive feature extraction and improve classification accuracy.

In Section 5.1, We studied the transfer learning method from eye state recognition to MI task. We created two transfer learning models and compared them

with the new model trained from scratch. Fine-tuning all layers is more adaptable to the learning of new MI task objectives than fine-tuning certain layers. While the difference between fine-tuned models and models trained from scratch is minimal in large data scenarios, transfer learning models demonstrate a significant advantage when training with small datasets. In Section 5.2, we analyzed left and right-hand motor imagery data from the BCI Competition IV 2b dataset, achieving an average accuracy of 84.1%. By comparing the classification results obtained from various preprocessing methods, we found that CWT is more suitable for integration with CNN to analyze MI-EEG signals compared to CSP, FFT, and STFT. Additionally, our model's performance surpassed that of most existing studies. Recognizing that the small dataset limits CNN performance, we proposed a novel data enhancement method based on transforming time and frequency domain data in Section 5.2. Experiments demonstrated that this DA method improves the performance of deep learning models to varying degrees. Concurrently, we enhanced the binary branch CNN mentioned in Section 5.2 by designing two CNNs of different scales for time domain and CWT mapping to achieve more comprehensive feature extraction. The average accuracy of this method reached 97.61%, the highest among all models. We believe this technology has the potential to be applied to other machine learning and MI-EEG analysis tasks, and we look forward to further exploring its capabilities.

In the 5.3 section, our goal is to design an inference accelerator for CNN that can be integrated into FPGA. However, this objective presents several design challenges, particularly because CNNs require significant computing resources, which may lead to insufficient memory or slow inference speed on the devices used. To achieve our goal, we need to overcome these difficulties by finding innovative solutions.

6. Hardware implementation

This chapter transitions from software-based algorithm development to practical hardware implementation. Hardware implementation of CNNs presents numerous challenges due to their computational intensity and complexity. In Section 3, we introduced hardware-based solutions to accelerate CNN computations. GPU platforms are often the preferred choice for this purpose because of their excellent computing performance and user-friendly development environment. However, GPUs have limited universal applicability due to their high cost, high power consumption, and large physical size. Instead, FPGA-based inference accelerators have the potential to surpass GPUs in terms of both CNN computing performance and power efficiency. This aligns with our goals for improving the practical implementation of CNNs. Traditional FPGA development includes several meticulous and time-consuming steps, including HDL coding, simulation, synthesis, implementation, and bitstream generation. These stages require significant effort and can be particularly cumbersome due to their iterative nature. Furthermore, with the rapid advancements and frequent updates in CNN models, researchers often need to test new models repeatedly, necessitating a re-execution of these development steps each time. This repetitive process leads to substantial time wastage for researchers.

To address these challenges, we aim to develop a design framework that allows for the modification of the CNN network structure at the software level. By enabling network reconfiguration through software, we can streamline the development process, significantly reducing the time and effort required to test new CNN models on FPGA platforms. We will introduce the development platforms and describe the acceleration circuit hardware frameworks designed in this chapter.

6.1 Development Platform

The Zynq [199] developed by Xilinx, is a programmable SoC that integrates both an FPGA chip and an ARM processor. Unlike other FPGA devices, Zynq combines the flexibility of an FPGA with the processing power of an ARM-based architecture. This integration is facilitated by interfaces like the AXI bus, allowing for seamless communication between the two components. The Zynq platform's ARM-based integrated architecture leverages the FPGA's parallelism to accelerate specific algorithms, providing benefits such as low energy consumption, easy deployment, and

mobility. Given the linear structures in CNNs, the ARM processor on the Zynq platform can achieve high operational efficiency, aligning well with the platform's strengths. PYNQ platform [200] short for Python on Zynq, aims to enable embedded developers to fully utilize the capabilities of Zynq without needing to work directly with programmable logic circuits. By using Python-based programming techniques, engineers can leverage Zynq's combination of programmable logic and microprocessors to develop advanced devices more efficiently. This makes the PYNQ platform an ideal choice for building a hardware CNN accelerator, as it allows developers to quickly prototype and deploy sophisticated designs using Python, capitalizing on Zynq's unique architecture. Therefore, we chose PYNQ-Z2 platform for building a hardware CNN accelerator.

6.2 Hardware circuit module design

The proposed framework of the CNN inference unit is shown in Figure 6.1. The well-trained CNN parameters and test samples will be uploaded from PC to PYNQ-Z2. Signal preprocessing and displaying are managed by ARM processing system (PS) part.

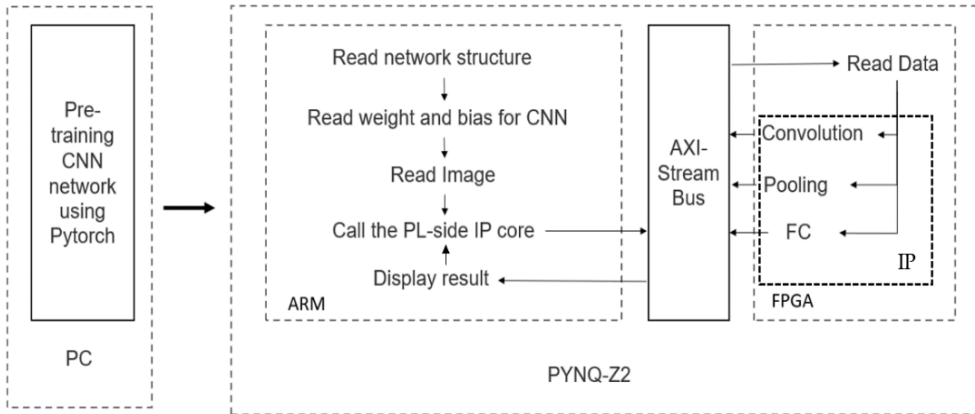


Figure 6.1 The overall design of the framework

The Programmable Logic (PL) part is responsible for the computation of convolutional, pooling and fully connection. The Xilinx High-Level Synthesis (HLS) Suite allows conversion of C, C++ or SystemC to RTL implementation. Compared to traditional hardware description languages (VHDL or Verilog), HLS significantly reduces the required code and enhances efficiency of intellectual property (IP) design. Therefore, we used HLS instead of Verilog to program the CNN circuit. It is important to note that not all C and C++ syntax is supported by HLS [201].

Since the convolution layer is the most computationally intensive block in a CNN, our main work is to optimize the calculation of the convolution layer. In order to improve the computational efficiency of the convolutional layer, it is very important to utilize the parallel computing capabilities of FPGA. Unrolled loops facilitate multiple independent operations simultaneously rather than sequentially processing a single set of operations [202]. In general, loops in C/C++ functions remain rolled. So, the synthesis creates the logic for one iteration of the loop, and the RTL design executes this logic for each iteration in sequence by default. **UNROLL** pragma transforms loops by generating multiple copies of the loop body in the RTL design, allowing several or all loop iterations to occur in parallel.

For convolution operations, not all loops can be unrolled. The independence and correlation of the loops need to be considered. Data dependency relationships in computing can be understood in three main classes [203]:

- **Irrelevant Dependencies:** This occurs when a variable does not interact with any function within an array. The variable is independent and does not influence the array's operations.
- **Independent Dependencies:** When an array can be completely divided along a specific loop dimension without affecting its operations, this dimension is considered independent. The computations along this dimension can occur simultaneously without interference.
- **Dependent Dependencies:** If an array cannot be divided along a particular loop dimension, that dimension is said to be dependent. The computations along this dimension are interlinked and must be executed sequentially.

Table 6.1 illustrates the data dependency relationships depicted in Figure 6.2. Figure 6.2 illustrates the positions of seven variables related to convolution computation within the loop. Here, R represents the height of the output feature map, C represents the width of the output feature map, M represents the number of output feature maps, N represents the depth of the input image, H represents the height of the convolution kernel, and W represents the width of the convolution kernel. The loop dimensions of “Number of output feature map” and “Input image depth” are suitable for unrolling because they are independent with another variables. Figure 6.3 presents a block of the convolutional computing function. The data is distributed to the parallel computing unit through the input register, weight register and crossbar to complete the

efficient convolution operation. The calculation result is stored in the output register after multiplication and addition operation.

Table 6.1 Data dependencies relationship of CNN

	Input Array	Weight Array
Output feature map height	Dependent	Irrelevant
Output feature map width	Dependent	Irrelevant
Number of output feature map	Irrelevant	Independent
Input image depth	Independent	Independent
Convolution kernel height	Dependent	Independent
Convolution kernel width	Dependent	Independent

```

for(row = 0; idx1 < R; row++) // Output feature map height
{
    for(col = 0; idx2 < C; col++) //Output feature map width
    {
        for(to = 0; idx3 < M; to++) //Number of output feature map
        {
            for(ti = 0; idx4 < N; ti++) //Input image depth
            {
                for(i = 0; i < H; i++) //Convolution kernel height
                {
                    for(j = 0; j < W; j++) //Convolution kernel width
                    {
                        output [idx3] [idx1] [idx2] += weights[idx3] [idx4] [i] [j] * input[idx4] [S * idx1 + i] [S * idx2 + j];
                    }
                }
            }
        }
    }
}

```

Figure 6.2 Pseudo code of convolution computing

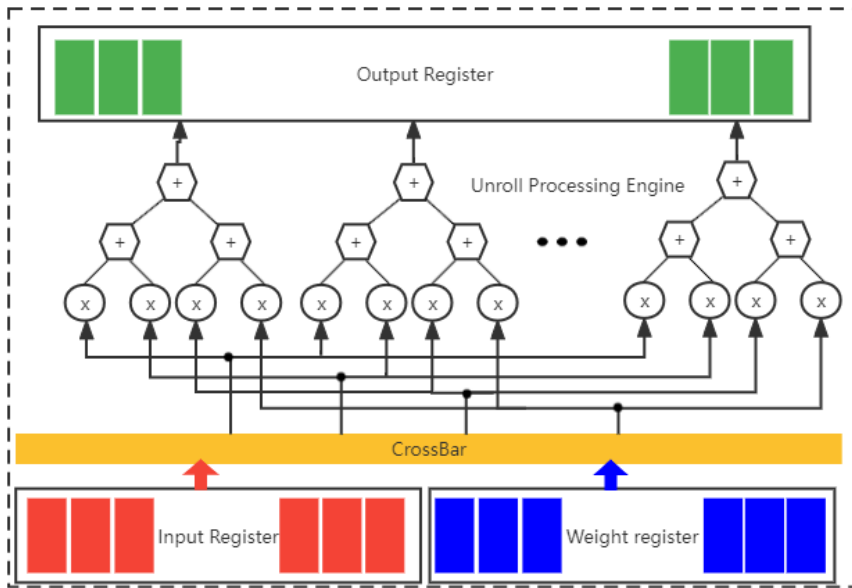


Figure 6.3 The convolution function block of the proposed accelerator structure

Table 6.2 presents the optimized resource usage. The on-chip resource usage is improved, and the latency is reduced 7.7 times after optimization. The digital signal processing (DSP) slices, specifically DSP48E slices, are versatile components within certain generations of FPGA devices, capable of performing various calculations such as multiply-accumulators, multiply-adders, and one- or n-step counters. In our case, the DSP48E slices are primarily utilized as multiply-accumulators to execute convolution operations.

Table 6.2 The utilization of convolution computing function before and after optimizing.

Name	After	Before
BRAM	8	5
DSP48E	20	5
Flip-flops	3209	623
Look-up tables	3573	1364
Latency	48,237	374,368
Interval	48,237	374,368

The results presented here provide the basis for the seventh thesis point, which has belonged to the thesis group 3. The result has been published to Electronics Journal with title “FPGA-Based Hardware Accelerator on Portable Equipment for EEG Signal Patterns Recognition[J1].”

Thesis 7: I have shown that the latency after the Optimization was reduced by 7.7 times. By comparing with other FPGA -based CNN design, the inference speed of our design is faster and has less resources consumed.

The PL system design is shown in Figure 6.4. We integrate the blocks for convolution, pooling, and fully connected functions into a single IP core. The system utilizes the AXI4-Lite bus for controlling signal communication and the AXI4 bus for data transfer.

- **IP Core Functions:** The combined IP core handles convolution, pooling, and fully connected functions. PS calls these functions based on the model's requirements. The output from each function is sent back to the PS, which then calls the next layer's function accordingly.
- **Communication Interface:** The ZYNQ7 processing system IP provides a logical connection between the PS and PL. It facilitates the interaction and coordination between the two subsystems.
- **Instruction and Control Signal Management:** The proposed accelerator functions as an IP core on the AXI4 bus. It interacts with the Direct Memory Access (DMA) IP to receive instructions and control signals from the ZYNQ7 processing system. The DMA IP allows for efficient data transfer between the IP core and the PS.
- **Memory Access:** Through the AXI4 bus, the DMA can directly access the DDR3 memory, enabling high-speed data transfer required for convolution, pooling, and fully connected operations.

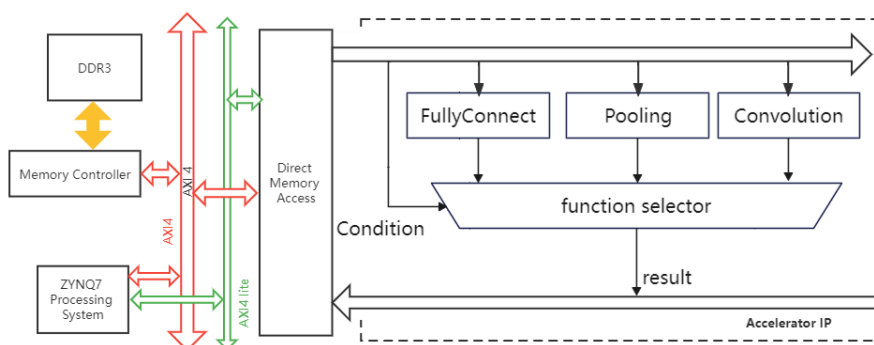


Figure 6.4 The overview PL system design.

The proposal aim is to create a flexible CNN accelerator. The structure can be adjusted as needed, including the window size, the arrangement of network layers, and the number of layers, to meet the specific requirements of various networks.

Another optimization method is to fuse the Batch normalization layer and convolutional layer, which can reduce the complexity of calculation [204]. At the same time, for this accelerator architecture, reducing the number of network layers also means reducing the amount of data transmission and inference time. The hardware platform structure is presented in the Figure 6.5. CNN_0 is the accelerator. This IP is connected to the DMA controller via the AXI bus and is responsible for network calculations. The DMA controller is used to efficiently manage data transfer. The entire system is coordinated by the ZYNQ processor, and data flow and inter-IP communication are achieved through AXI interconnection.

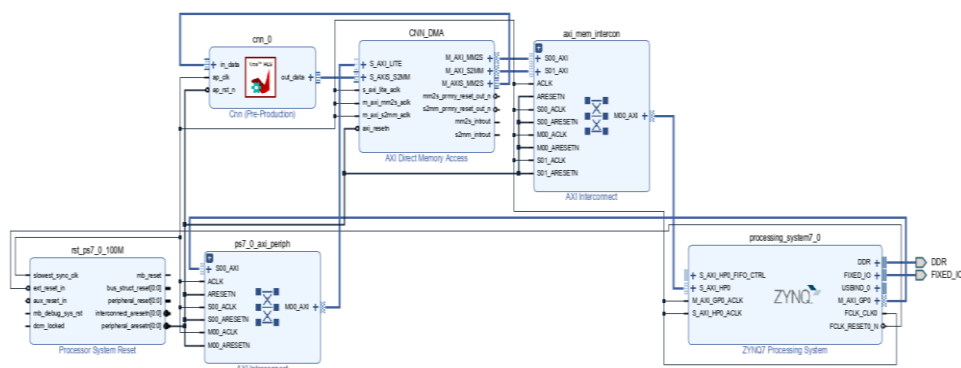


Figure 6.5 The hardware platform structure on Vivado 2021.1

6.3 Experimental evaluation

The hardware and software environment we used is presented below:

Personal computer:

CPU: Intel Core i5-7300HQ,

GPU: GeForce GTX 1050Ti,

OS: Window 10 64-bit, 21H2.

Software: Vivado 2021.1; Vitis HLS 2021.1; Python 3.9.

PYNQ-Z2 platform:

CPU: dual-core ARM Cortex-A9,

FPGA: XC7Z020-1CLG400C,

OS: Ubuntu 20.04,

Software: Python 3.8; JupyterLab 3.0.16.

The CNN and dataset introduced in section 4.2 were used for test performance of our accelerator. In section 4.2 (Figure 4.14), we mentioned 1-second time window is enough for detecting EEG tasks. It offers a reasonable accuracy rate and acceptable latency compared to other time window sizes. While larger segment sizes typically result in higher accuracy, they also raise the computational time and complexity of the accelerator due to the need to process a bigger size of image. Thus, there is a trade-off between achieving higher accuracy and managing computational resources efficiently. The following experiments on PYNQ-Z2 are based on one-second window size.

We chose CWT-CNN in section 4.2 as the first network to test. Table 6.3 lists the comprehensive performance comparison of accelerators on different platforms. Power consumption for the ARM Cortex-A9 was estimated based on information from the Xilinx official website. Accurately estimating power consumption for only the GPU or a single inference program on PC or PYNQ is challenging. Thus, we calculate the total energy usage of the OS when running an inference program. It was measured using a socket wattmeter. The power shown by the socket wattmeter is not the actual power consumption because the power supply has conversion efficiency. The actual power consumption must be adjusted by considering the power supply conversion efficiency.

The evaluation criterion for power efficiency is calculated as follows:

$$\text{Power efficiency} = \frac{\text{Throughput}}{\text{Power}} = \frac{\text{Power}}{\text{Inference Time}} \quad (6.1)$$

Table 6.3 The comparative performance of different platform

Hardware Platform	CPU	Accelerator	Memory	Inference Time	Accuracy	Power	Power Efficiency	Price	Weight
Laptop	Intel Core i5-7300HQ @ 2.50 GHz	GTX 1050Ti @ 1620 Mhz	8 GB DDR4 @2666 MHz	0.003 s	89.92%	131.1 W	2.5	~\$750	3.61 kg
PYNQ-Z2	ARM Dual-core Cortex A9 @ 650 MHz	-	512 MB DDR3 @525 MHz	5.1 s	89.85%	1.9 W	0.1	~\$177	0.22 kg
PYNQ-Z2	ARM Dual-core Cortex A9 @ 650 MHz	FPGA @ 100 Mhz	512 MB DDR3 @525 MHz	0.22 s	89.69%	3.1 W	1.46	~\$177	0.22 kg

The accuracy of the three platforms is basically the same, and the maximum does not exceed 0.23%. It shows that our accelerator has completed its basic functions. We focus on analyzing other indicators. The result shows that the accelerator on the ARM operates approximately 16 times slower than the accelerator running on the FPGA. However, the GPU platform performance period is about 71 times faster than that of the FPGAs. The GPU is significantly consuming power 37 times higher than the FPGA in terms of power consumption. Regarding power efficiency, the FPGA demonstrates a substantial advantage over the ARM CPU, being 14.6 times more efficient, but it only reaches 0.58 times the efficiency of the laptop platform. The proposed framework is highly dependent on memory frequency, as computation results for each layer are stored in external DDR3 memory and then re-read for subsequent layer calculations. The memory frequency of GPU is 16 times that of FPGA. The complexity of the CNN model amplifies the impact of memory frequency on inference time. This is one of the reasons why FPGAs are not efficient enough. Despite this, FPGA platforms offer advantages in terms of cost and weight compared to GPUs.

Due to different goals and methods, it is difficult to compare with other designs. Therefore, we chose the LeNet-5 [205] model to facilitate comparison with other designs. The test data is MNIST handwritten digits. Table 6 shows the comparison result between different designs.

Our work demonstrates shorter inference time and uses fewer resources, specifically Look-Up Tables (LUTs) and Flip-Flops, compared to the results presented in [32]. When compared to [33], their implementation was based on 12-bit fixed-point numbers, and our applied 32 bits float point numbers. Shorter data types contribute to shorter inference time and less on-chip resource consumption for their models, at the expense of reduced accuracy. Unlike other models, we use Vivado HLS as our programming language, which is more accessible than hardware description languages. The most important thing is the convenience of our design. The other work is focus on specific CNN framework, whereas our approach allows for easy modification of the CNN model through a Python application, such as Jupyter Notebook, to meet the developer's needs. It can modify network framework at the software level. Our approach allows for changes to the network framework without significantly impacting resource usage and operational efficiency compared to traditional design methods. Consequently, it saves developers considerable time otherwise spent on redesigning, synthesis, placement, and routing processes.

Table 6.4 The comparison with some predecessors.

Metric	Our Design	[206]	[207]
Model	LeNet-5	LeNet-5	LeNet-5
Programming language	HLS	Verilog	Verilog
Data type	32 bits float	32 bits float	12 bits fixed
Operation frequency	100 MHz	100 MHz	100 MHz
BRAM	264	2	45
DSP48E	62	23	158
Flip-flops	8947	11962	2772
Look-up tables	9334	9893	4254
Accuracy	98.34%	98.2%	97.59%
Inference time	0.17 s	0.653 s	0.143 s

The results presented here provide the basis for the eighth thesis point, which is belong to the thesis group 3. The result has been published to International Conference on e-Health and Bioengineering (EHB) with title “Hardware Implementation of CNN Based on FPGA for EEG Signal Patterns Recognition[C1].”

Thesis 8: I demonstrated that Our accelerator is not only 16 times faster than traditional CPU-based inference but also achieves a remarkable energy efficiency ratio of 14.6 times that of CPU. Moreover, our accelerator is highly flexible, allowing researchers to modify the CNN structure at the software level without requiring any hardware changes.

6.4 Conclusion

This chapter explores a lightweight CNN structure tested on the PYNQ-Z2 FPGA development board. This framework enables easy reconfiguration of the network model, allowing it to adapt swiftly to the evolving landscape of deep learning algorithms. Compared to ARM processors, the FPGA on PYNQ demonstrates significant acceleration. Our findings highlight time savings and development simplification for researchers. Comparative analysis of the three platforms shows that while the GPU outperforms the FPGA on PYNQ in terms of operating speed and power consumption efficiency, the FPGA excels in low power consumption, high portability, and cost-effectiveness. These advantages are particularly beneficial for real-time inference systems on portable devices.

7. Summary and future directions

Our work aims to address the challenges of EEG processing by exploring new methods for EEG signal processing, feature extraction, classification, and hardware acceleration. The contribution of thesis is following:

1. We bring together the latest advancements in EEG signal processing and classification, offering a comprehensive and insightful review of both traditional and state-of-the-art techniques. By providing a detailed comparison of different approaches, including EEG signal pre-processing, feature extraction, classification, and their hardware solution, we help researchers and practitioners navigate the complex landscape of EEG analysis and choose the most suitable methods for their specific applications. At the same time, it also provides a theoretical basis for further exploration and development of EEG related algorithms for our study.

2. We analyze the basic characteristics of EEG Signals in detail and describe the construction of an MLP network and SCNN specifically designed for EEG data classification. Our comparative analysis of the MLP and SCNN shows that classifying alpha or beta waves alone does not yield significant performance differences. However, an extended dataset reveals that beta waves are more reliable indicators than alpha waves for eye state classification. While the MLP achieves higher accuracy and shorter training times, CNN's integrated approach to preprocessing, feature extraction, and classification helps avoid common issues in EEG signal processing, maintaining competitive accuracy.

3. We collect eye EEG data using the OpenBCI device and self-developed software, introducing CWT-CNN and STFT-CNN network. We compare their classification accuracy to raw EEG signals, demonstrating the effectiveness of feature extraction in improving classification tasks. We noticed that the pre-processing of the time -frequency analysis technology can significantly improve the accuracy of classification. Given the variability in performance across different preprocessing techniques and segment sizes, there may be opportunities to leverage branch learning approaches to further improve classification accuracy. Branch methods can combine multiple CNN models trained with different preprocessing techniques or segment sizes to enhance overall performance and robustness, potentially achieving even higher accuracy than any single model alone.

4. We introduce parallel structures using CNN and different techniques, such as CWT, CSP, STFT for comprehensive feature extraction and improved classification accuracy. We found that our model's performance surpasses that of most existing studies.

5. Realizing that the lack of data has a great impact on the network, we have developed special data enhanced technology based on the characteristics of EEG. This technology has an accurate effect in multiple data sets in our experiments. Our binary branch CWT-CNN has the highest accuracy compared to other state-of-the-art models.

6. Our study switch from software-based algorithm development to practical hardware implementation. Hardware implementation of CNNs presents numerous challenges due to their computational intensity and complexity. We explored a lightweight CNN structure for EEG signal and tested on the PYNQ-Z2 FPGA development board. This framework allows easy reconfiguration of the network model, enabling quick adaptation to evolving deep learning algorithms. Compared to ARM processors, the FPGA on PYNQ demonstrates significant acceleration, highlighting time savings and development simplification for researchers. Comparative analysis of the three platforms—GPU, ARM, and FPGA on PYNQ—reveals that while the GPU outperforms the FPGA in terms of operating speed, the FPGA excels in low power consumption, high portability, and cost-effectiveness. These advantages make the FPGA on PYNQ particularly beneficial for real-time MI inference systems on portable devices for edge computing.

Future research directions in EEG processing could explore several promising directions to build upon our work.

1. Exploring more complex deep learning architectures, such as combining CNN with Recurrent Neural Network or Graph Neural Network, may provide new ways for more efficient and accurate EEG analysis. These models have performed well in many fields. By adjusting their structure, they can better capture the spatial characteristics and temporal dynamics of EEG signals at the same time, thereby further improving the analysis performance.

2. Integrating multimodal data, such as combining EEG with other physiological signals (such as electromyography, electrocardiography) or environmental data, can more comprehensively analyze brain activity and improve the robustness and

adaptability of the system in different application scenarios. At the same time, developing automated and adaptive processing techniques can effectively manage with the differences in EEG signals among different subjects and environments, reducing the need for manual intervention.

3. In real-time and embedded systems, further optimize hardware implementation and explore energy-efficient architectures to adapt to more resource-efficient and lower-latency FPGA platforms, and implement real-time EEG signal analysis on wearable devices and embedded systems. This can create new possibilities for innovative applications in healthcare, gaming, and human-computer interaction.

Acknowledgment

Time flies, and my doctoral journey is nearing its end. Reflecting on the past years, I realize that every bit of progress I have made is inseparable from the earnest guidance of my professors, the enthusiastic support of my friends, and the silent backing of my family. As I prepare to graduate, I would like to express my heartfelt gratitude to all my teachers, family, and friends for their invaluable support and encouragement.

Frist, I would like to express my deepest gratitude to my supervisor, Istvan Oniga, for his unwavering support, invaluable guidance, and profound knowledge throughout my doctoral journey. Without his mentorship, this accomplishment would not have been possible.

I am also profoundly thankful to my wife and my parents for their enduring love, constant encouragement, and unwavering support throughout the years. Their understanding, patience, and sacrifices have been the cornerstone of my success. Their belief in me has been a constant source of motivation, and I am forever appreciative of their sacrifices and encouragement.

Furthermore, I wish to express my sincere appreciation to all my friends and colleagues who have offered their encouragement, shared insights, and stood by me during challenging times. Your help has enriched my academic experience and made this journey more rewarding.

Finally, I would like to extend my heartfelt wishes for the continued success and good health of all those who have supported me on this journey. May your lives be filled with happiness, prosperity, and fulfillment. Thank you once again for your unwavering support and encouragement.

Bibliography

1. Nicolas-Alonso, L.F. and J. Gomez-Gil, *Brain computer interfaces, a review*. *sensors*, 2012. **12**(2): p. 1211-1279.
2. Tang, X., et al., *Motor imagery EEG recognition based on conditional optimization empirical mode decomposition and multi-scale convolutional neural network*. *Expert Systems with Applications*, 2020. **149**: p. 113285.
3. van de Laar, B., et al., *Experiencing BCI control in a popular computer game*. *IEEE Transactions on Computational Intelligence and AI in Games*, 2013. **5**(2): p. 176-184.
4. Carabalona, R., et al., *Light on! Real world evaluation of a P300-based brain-computer interface (BCI) for environment control in a smart home*. *Ergonomics*, 2012. **55**(5): p. 552-563.
5. Kotchetkov, I.S., et al., *Brain-computer interfaces: military, neurosurgical, and ethical perspective*. *Neurosurgical focus*, 2010. **28**(5): p. E25.
6. Korats, G., et al. *Applying ICA in EEG: choice of the window length and of the decorrelation method*. in *Biomedical Engineering Systems and Technologies: 5th International Joint Conference, BIOSTEC 2012, Vilamoura, Portugal, February 1-4, 2012, Revised Selected Papers 5*. 2013. Springer.
7. Liu, T. and D. Yao, *Removal of the ocular artifacts from EEG data using a cascaded spatio-temporal processing*. *Computer methods and programs in biomedicine*, 2006. **83**(2): p. 95-103.
8. Vorobyov, S. and A. Cichocki, *Blind noise reduction for multisensory signals using ICA and subspace filtering, with application to EEG analysis*. *Biological Cybernetics*, 2002. **86**(4): p. 293-303.
9. Lakshmi, M.R., T. Prasad, and D.V.C. Prakash, *Survey on EEG signal processing methods*. *International journal of advanced research in computer science and software engineering*, 2014. **4**(1).
10. Yu, X., P. Chum, and K.-B. Sim, *Analysis the effect of PCA for feature reduction in non-stationary EEG based motor imagery of BCI system*. *Optik*, 2014. **125**(3): p. 1498-1502.
11. Moctezuma, L.A., et al., *Subjects identification using EEG-recorded imagined speech*. *Expert Systems with Applications*, 2019. **118**: p. 201-208.
12. Velásquez-Martínez, L.F., A.M. Álvarez-Meza, and C.G. Castellanos-Domínguez. *Motor imagery classification for BCI using common spatial patterns and feature relevance analysis*. in *Natural and Artificial Computation in Engineering and Medical Applications: 5th International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2013, Mallorca, Spain, June 10-14, 2013. Proceedings, Part II 5*. 2013. Springer.
13. Kim, C., et al., *An effective feature extraction method by power spectral density of EEG signal for 2-class motor imagery-based BCI*.

- Medical & biological engineering & computing, 2018. **56**: p. 1645-1658.
14. Kundu, S. and S. Ari, *P300 detection with brain-computer interface application using PCA and ensemble of weighted SVMs*. IETE Journal of Research, 2018. **64**(3): p. 406-414.
 15. Dimigen, O., *Optimizing the ICA-based removal of ocular EEG artifacts from free viewing experiments*. NeuroImage, 2020. **207**: p. 116117.
 16. Lawhern, V., et al., *Detection and classification of subject-generated artifacts in EEG signals using autoregressive models*. Journal of neuroscience methods, 2012. **208**(2): p. 181-189.
 17. Edla, D.R., et al., *Classification of facial expressions from eeg signals using wavelet packet transform and svm for wheelchair control operations*. Procedia computer science, 2018. **132**: p. 1467-1476.
 18. Suto, J. and S. Oniga, *Efficiency investigation of artificial neural networks in human activity recognition*. Journal of Ambient Intelligence and Humanized Computing, 2018. **9**: p. 1049-1060.
 19. Suto, J., et al., *Comparison of offline and real-time human activity recognition results using machine learning techniques*. Neural computing and applications, 2020. **32**: p. 15673-15686.
 20. Ventouras, E.M., et al., *Sleep spindle detection using artificial neural networks trained with filtered time-domain EEG: a feasibility study*. Computer methods and programs in biomedicine, 2005. **78**(3): p. 191-207.
 21. Dande, P. and P. Samant, *Acquaintance to artificial neural networks and use of artificial intelligence as a diagnostic tool for tuberculosis: a review*. Tuberculosis, 2018. **108**: p. 1-9.
 22. Grossi, E., C. Olivieri, and M. Buscema, *Diagnosis of autism through EEG processed by advanced computational algorithms: A pilot study*. Computer methods and programs in biomedicine, 2017. **142**: p. 73-79.
 23. Craik, A., Y. He, and J.L. Contreras-Vidal, *Deep learning for electroencephalogram (EEG) classification tasks: a review*. Journal of neural engineering, 2019. **16**(3): p. 031001.
 24. Soucy, P. and G.W. Mineau. *A simple KNN algorithm for text categorization*. in *Proceedings 2001 IEEE international conference on data mining*. 2001. IEEE.
 25. Murphy, K., et al., *A large-scale evaluation of automatic pulmonary nodule detection in chest CT using local image features and k-nearest-neighbour classification*. Medical image analysis, 2009. **13**(5): p. 757-770.
 26. Ilyas, M.Z., P. Saad, and M.I. Ahmad. *A survey of analysis and classification of EEG signals for brain-computer interfaces*. in *2015 2nd International Conference on Biomedical Engineering (ICoBE)*. 2015. IEEE.

27. Narayan, Y., *Motor-Imagery EEG Signals Classification using SVM, MLP and LDA Classifiers*. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021. **12**(2): p. 3339-3344.
28. Cortes, C. and V. Vapnik, *Support-vector networks*. Machine learning, 1995. **20**: p. 273-297.
29. Li, Y., *Recognition algorithm of driving fatigue related problems based on EEG signals*. NeuroQuantology, 2018. **16**(6).
30. Siuly, S. and Y. Li, *Designing a robust feature extraction method based on optimum allocation and principal component analysis for epileptic EEG signal classification*. Computer methods and programs in biomedicine, 2015. **119**(1): p. 29-42.
31. Tahernezhad-Javazm, F., V. Azimirad, and M. Shoaran, *A review and experimental study on the application of classifiers and evolutionary algorithms in EEG-based brain-machine interface systems*. Journal of neural engineering, 2018. **15**(2): p. 021007.
32. Machado, J. and A. Balbinot, *Executed movement using EEG signals through a Naive Bayes classifier*. Micromachines, 2014. **5**(4): p. 1082-1105.
33. Su, H. and C. Jung, *Perceptual enhancement of low light images based on two-step noise suppression*. IEEE Access, 2018. **6**: p. 7005-7018.
34. Obeidat, M.A. and A.M. Mansour, *EEG Based Epilepsy Diagnosis System using Reconstruction Phase Space and Naïve Bayes Classifier*. WSEAS Transactions on Circuits and Systems, 2018. **17**.
35. Katkar, V.D. and S.V. Kulkarni. *A novel parallel implementation of Naive Bayesian classifier for Big Data*. in *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*. 2013. IEEE.
36. Larose, D.T. and C.D. Larose, *Discovering knowledge in data: an introduction to data mining*. Vol. 4. 2014: John Wiley & Sons.
37. Amedi, A., et al., *Cortical activity during tactile exploration of objects in blind and sighted humans*. Restorative neurology and neuroscience, 2010. **28**(2): p. 143-156.
38. Librelotto, S.R. and P.M. Mozzaquatro, *Análise dos algoritmos de mineração J48 e Apriori aplicados na detecção de indicadores da qualidade de vida e saúde*. Revista Interdisciplinar de Ensino, Pesquisa e Extensão-RevInt, 2014. **1**(1).
39. Alvarenga, M.T., *Utilização da ferramenta j48 para descoberta do conhecimento em bases de dados fitossanitários, climáticos e espectrais*. 2014, Ph. D. Dissertation. Master thesis, Universidade Federal de Lavras, Minas
40. Narayan, R., *Encyclopedia of biomedical engineering*. 2018: Elsevier.
41. Stone, D.B., et al., *Automatic removal of physiological artifacts in EEG: the optimized fingerprint method for sports science applications*. Frontiers in human neuroscience, 2018. **12**: p. 96.

42. Bhuvaneswari, P. and J.S. Kumar. *Methods used for identifying EEG signal artifacts*. in *Proc. of the International Conference on Computational Intelligence and Information Technology*. 2012.
43. Khatter, A., D. Bansal, and R. Mahajan, *Study of various automatic eeg artifact removal techniques*. International Journal for Research in Applied Science & Engineering Technology, 2017. **5**(10): p. 1027-1037.
44. Saltuklaroglu, T., et al., *EEG mu rhythms: Rich sources of sensorimotor information in speech processing*. Brain and language, 2018. **187**: p. 41-61.
45. Albaqami, H., et al., *Automatic detection of abnormal EEG signals using wavelet feature extraction and gradient boosting decision tree*. Biomedical Signal Processing and Control, 2021. **70**: p. 102957.
46. Yang, C.-J., et al. *An ai-edge platform with multimodal wearable physiological signals monitoring sensors for affective computing applications*. in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020. IEEE.
47. Aslam, A.R. and M.A.B. Altaf. *An 8 channel patient specific neuromorphic processor for the early screening of autistic children through emotion detection*. in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2019. IEEE.
48. Aslam, A.R., et al. *A10. 13uJ/classification 2-channel deep neural network-based SoC for emotion detection of autistic children*. in *2020 IEEE Custom Integrated Circuits Conference (CICC)*. 2020. IEEE.
49. Yarahuaman, J.C.R. and P.F. Huamani-Navarrete. *Design and Simulation of a Digital Filter in Hardware for EEG Signals Based on FPGA*. in *2020 IEEE ANDESCON*. 2020. IEEE.
50. Aslam, A.R. and M.A.B. Altaf, *An on-chip processor for chronic neurological disorders assistance using negative affectivity classification*. IEEE Transactions on Biomedical Circuits and Systems, 2020. **14**(4): p. 838-851.
51. Nunez, P.L. and R. Srinivasan, *Electric fields of the brain: the neurophysics of EEG*. 2006: Oxford University Press, USA.
52. Gonzalez, H.A., et al. *An inference hardware accelerator for EEG-based emotion detection*. in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020. IEEE.
53. Yang, C.-H., Y.-H. Shih, and H. Chiueh, *An $81.6\mu\text{W}$ FastICA Processor for Epileptic Seizure Detection*. IEEE Transactions on Biomedical circuits and Systems, 2014. **9**(1): p. 60-71.
54. Wu, D., et al., *Exploiting transfer learning for emotion recognition under cloud-edge-client collaborations*. IEEE Journal on Selected Areas in Communications, 2020. **39**(2): p. 479-490.
55. Tao, X., et al., *Rebuffering optimization for DASH via pricing and EEG-based QoE modeling*. IEEE Journal on Selected Areas in Communications, 2019. **37**(7): p. 1549-1565.

56. Chen, Y.-H., S.-W. Chen, and M.-X. Wei, *A VLSI implementation of independent component analysis for biomedical signal separation using CORDIC engine*. IEEE Transactions on Biomedical Circuits and Systems, 2020. **14**(2): p. 373-381.
57. Chen, Y.-H. and S.-P. Wang, *Low-cost implementation of independent component analysis for biomedical signal separation using very-large-scale integration*. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020. **67**(12): p. 3437-3441.
58. Altaheri, H., et al., *Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: A review*. Neural Computing and Applications, 2023. **35**(20): p. 14681-14722.
59. Aggarwal, S. and N. Chugh, *Review of machine learning techniques for EEG based brain computer interface*. Archives of Computational Methods in Engineering, 2022: p. 1-20.
60. Knyazev, G.G., et al., *Cross-frequency coupling in developmental perspective*. Frontiers in human neuroscience, 2019. **13**: p. 158.
61. Yang, J. and M. Sawan, *From seizure detection to smart and fully embedded seizure prediction engine: A review*. IEEE Transactions on Biomedical Circuits and Systems, 2020. **14**(5): p. 1008-1023.
62. P. Davis, C.D.C.a.W.T., *ASIC implementation of the cross frequency coupling algorithm for EEG signal processing*. 2014 International Symposium on Integrated Circuits (ISIC), 2014. pp. **248-251**.
63. Rizal, A., S. Hadiyoso, and A.Z. Ramdani, *FPGA-Based Implementation for Real-Time Epileptic EEG Classification Using Hjorth Descriptor and KNN*. Electronics, 2022. **11**(19): p. 3026.
64. Richman, J.S. and J.R. Moorman, *Physiological time-series analysis using approximate entropy and sample entropy*. American journal of physiology-heart and circulatory physiology, 2000. **278**(6): p. H2039-H2049.
65. Chen, C., et al. *Acceleration of Fast Sample Entropy Towards Biomedical Applications on FPGAs*. in *2022 International Conference on Field-Programmable Technology (ICFPT)*. 2022. IEEE.
66. Wickramasuriya, D.S., L.P. Wijesinghe, and S. Mallawaarachchi. *Seizure prediction using Hilbert Huang Transform on field programmable gate array*. in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 2015. IEEE.
67. Zhang, L. *Real-time feature extraction for multi-channel EEG signals time-frequency analysis*. in *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*. 2017. IEEE.
68. Beeraka, S.M., et al., *Accuracy enhancement of epileptic seizure detection: a deep learning approach with hardware realization of STFT*. Circuits, Systems, and Signal Processing, 2022. **41**: p. 461-484.
69. Fang, W.-C., et al., *Development and validation of an EEG-based real-time emotion recognition system using edge AI computing platform with convolutional neural network system-on-chip design*.

- IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019. **9**(4): p. 645-657.
70. Ellawala, N. and S. Thayaparan. *Hardware implementation of eeg classifier using lda*. in *2019 2nd International Conference on Bioinformatics, Biotechnology and Biomedical Engineering (BioMIC)-Bioinformatics and Biomedical Engineering*. 2019. IEEE.
 71. Qassim, Y.T., et al. *FPGA implementation of Morlet continuous wavelet transform for EEG analysis*. in *2012 International Conference on Computer and Communication Engineering (ICCCE)*. 2012. IEEE.
 72. Belwafi, K., et al., *A hardware/software prototype of EEG-based BCI system for home device control*. Journal of Signal Processing Systems, 2017. **89**: p. 263-279.
 73. Kais, B., et al. *An embedded implementation of home devices control system based on brain computer interface*. in *2014 26th International Conference on Microelectronics (ICM)*. 2014. IEEE.
 74. Malekmohammadi, A., et al., *An efficient hardware implementation for a motor imagery brain computer interface system*. Scientia Iranica, 2019. **26**(Special Issue on: Socio-Cognitive Engineering): p. 72-94.
 75. Duan, R.-N., J.-Y. Zhu, and B.-L. Lu. *Differential entropy feature for EEG-based emotion classification*. in *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*. 2013. IEEE.
 76. Kaminski, M.J. and K.J. Blinowska, *A new method of the description of the information flow in the brain structures*. Biological cybernetics, 1991. **65**(3): p. 203-210.
 77. Wang, Y., et al. *EmotioNet: A 3-D convolutional neural network for EEG-based emotion recognition*. in *2018 international joint conference on neural networks (IJCNN)*. 2018. IEEE.
 78. Salama, E.S., et al., *EEG-based emotion recognition using 3D convolutional neural networks*. International Journal of Advanced Computer Science and Applications, 2018. **9**(8).
 79. Singh, K. and J.P. de Gyvez, *Twenty years of near/sub-threshold design trends and enablement*. IEEE Transactions on Circuits and Systems II: Express Briefs, 2020. **68**(1): p. 5-11.
 80. Höppner, S., et al., *Adaptive body bias aware implementation for ultra-low-voltage designs in 22FDX technology*. IEEE Transactions on Circuits and Systems II: Express Briefs, 2019. **67**(10): p. 2159-2163.
 81. Gao, Z., et al., *A coincidence-filtering-based approach for CNNs in EEG-based recognition*. IEEE Transactions on Industrial Informatics, 2019. **16**(11): p. 7159-7167.
 82. LeCun, Y., J. Denker, and S. Solla, *Optimal brain damage*. Advances in neural information processing systems, 1989. **2**.
 83. Hassibi, B. and D. Stork, *Second order derivatives for network pruning: Optimal brain surgeon*. Advances in neural information processing systems, 1992. **5**.

84. Han, S., et al., *Learning both weights and connections for efficient neural network*. Advances in neural information processing systems, 2015. **28**.
85. Han, S., H. Mao, and W.J. Dally, *Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding*. arXiv preprint arXiv:1510.00149, 2015.
86. Zhu, M. and S. Gupta, *To prune, or not to prune: exploring the efficacy of pruning for model compression*. arXiv preprint arXiv:1710.01878, 2017.
87. Alford, S., et al. *Training behavior of sparse neural network topologies*. in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. 2019. IEEE.
88. Liu, Z., et al., *Frequency-domain dynamic pruning for convolutional neural networks*. Advances in neural information processing systems, 2018. **31**.
89. Ding, X., et al., *Global sparse momentum sgd for pruning very deep neural networks*. Advances in Neural Information Processing Systems, 2019. **32**.
90. Lee, E. and Y. Hwang, *Layer-Wise Network Compression Using Gaussian Mixture Model*. Electronics, 2021. **10**(1): p. 72.
91. Yang, T.-J., Y.-H. Chen, and V. Sze. *Designing energy-efficient convolutional neural networks using energy-aware pruning*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
92. Li, H., et al., *Pruning filters for efficient convnets*. arXiv preprint arXiv:1608.08710, 2016.
93. He, Y., et al., *Soft filter pruning for accelerating deep convolutional neural networks*. arXiv preprint arXiv:1808.06866, 2018.
94. Hu, H., et al., *Network trimming: A data-driven neuron pruning approach towards efficient deep architectures*. arXiv preprint arXiv:1607.03250, 2016.
95. Lin, M., et al. *Hrank: Filter pruning using high-rank feature map*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.
96. Luo, J.-H., J. Wu, and W. Lin. *Thinet: A filter level pruning method for deep neural network compression*. in *Proceedings of the IEEE international conference on computer vision*. 2017.
97. Yu, R., et al. *Nisp: Pruning networks using neuron importance score propagation*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
98. Chen, S. and Q. Zhao, *Shallowing deep networks: Layer-wise pruning based on feature representations*. IEEE transactions on pattern analysis and machine intelligence, 2018. **41**(12): p. 3048-3056.
99. Elkerdawy, S., et al. *To filter prune, or to layer prune, that is the question*. in *Proceedings of the Asian Conference on Computer Vision*. 2020.

100. Xu, P., et al., *Layer pruning via fusible residual convolutional block for deep neural networks*. arXiv preprint arXiv:2011.14356, 2020.
101. Liu, Z., et al., *Rethinking the value of network pruning*. arXiv preprint arXiv:1810.05270, 2018.
102. Liu, Z., et al. *Learning efficient convolutional networks through network slimming*. in *Proceedings of the IEEE international conference on computer vision*. 2017.
103. Wang, Y., et al. *Pruning from scratch*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.
104. Zhou, S., et al., *Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients*. arXiv preprint arXiv:1606.06160, 2016.
105. Miyashita, D., E.H. Lee, and B. Murmann, *Convolutional neural networks using logarithmic data representation*. arXiv preprint arXiv:1603.01025, 2016.
106. Fiesler, E., A. Choudry, and H.J. Caulfield. *Weight discretization paradigm for optical neural networks*. in *Optical interconnections and networks*. 1990. SPIE.
107. Wu, H., et al., *Integer quantization for deep learning inference: Principles and empirical evaluation*. arXiv preprint arXiv:2004.09602, 2020.
108. Banner, R., Y. Nahshan, and D. Soudry, *Post training 4-bit quantization of convolutional networks for rapid-deployment*. Advances in Neural Information Processing Systems, 2019. **32**.
109. Jacob, B., et al. *Quantization and training of neural networks for efficient integer-arithmetic-only inference*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
110. Hubara, I., et al., *Quantized neural networks: Training neural networks with low precision weights and activations*. The Journal of Machine Learning Research, 2017. **18**(1): p. 6869-6898.
111. Zhou, Q., et al. *Octo:{INT8} training with loss-aware compensation and backward quantization for tiny on-device learning*. in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 2021.
112. Courbariaux, M., Y. Bengio, and J.-P. David, *Binaryconnect: Training deep neural networks with binary weights during propagations*. Advances in neural information processing systems, 2015. **28**.
113. Courbariaux, M., et al., *Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1*. arXiv preprint arXiv:1602.02830, 2016.
114. Rastegari, M., et al. *Xnor-net: Imagenet classification using binary convolutional neural networks*. in *European conference on computer vision*. 2016. Springer.
115. Hou, L., Q. Yao, and J.T. Kwok, *Loss-aware binarization of deep networks*. arXiv preprint arXiv:1611.01600, 2016.

116. Zhou, A., et al., *Incremental network quantization: Towards lossless cnns with low-precision weights*. arXiv preprint arXiv:1702.03044, 2017.
117. Qin, H., et al. *Forward and backward information retention for accurate binary neural networks*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.
118. Denil, M., et al., *Predicting parameters in deep learning*. Advances in neural information processing systems, 2013. **26**.
119. Klema, V. and A. Laub, *The singular value decomposition: Its computation and some applications*. IEEE Transactions on automatic control, 1980. **25**(2): p. 164-176.
120. Masana, M., et al. *Domain-adaptive deep network compression*. in *Proceedings of the IEEE International Conference on Computer Vision*. 2017.
121. Swaminathan, S., et al., *Sparse low rank factorization for deep neural network compression*. Neurocomputing, 2020. **398**: p. 185-196.
122. Zhang, X., et al., *Accelerating very deep convolutional networks for classification and detection*. IEEE transactions on pattern analysis and machine intelligence, 2015. **38**(10): p. 1943-1955.
123. Yang, H., et al. *Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020.
124. Chen, S., et al., *Joint matrix decomposition for deep convolutional neural networks compression*. Neurocomputing, 2023. **516**: p. 11-26.
125. Kim, Y.-D., et al., *Compression of deep convolutional neural networks for fast and low power mobile applications*. arXiv preprint arXiv:1511.06530, 2015.
126. Lebedev, V., et al., *Speeding-up convolutional neural networks using fine-tuned cp-decomposition*. arXiv preprint arXiv:1412.6553, 2014.
127. Astrid, M. and S.-I. Lee. *Cp-decomposition with tensor power method for convolutional neural networks compression*. in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2017. IEEE.
128. Yang, Y., D. Krompass, and V. Tresp. *Tensor-train recurrent neural networks for video classification*. in *International Conference on Machine Learning*. 2017. PMLR.
129. Yin, M., et al. *Towards efficient tensor decomposition-based dnn model compression with optimization framework*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.
130. Wang, D., et al., *Compressing 3DCNNs based on tensor train decomposition*. Neural Networks, 2020. **131**: p. 215-230.
131. Li, N., et al., *Heuristic rank selection with progressively searching tensor ring network*. Complex & Intelligent Systems, 2021: p. 1-15.

132. Phan, A.-H., et al. *Stable low-rank tensor decomposition for compression of convolutional neural network*. in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX* 16. 2020. Springer.
133. Bucilua, C., R. Caruana, and A. Niculescu-Mizil, *Model compression*, in *proceedings of the 12 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA, 2006. 4.
134. Hinton, G., O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*. arXiv preprint arXiv:1503.02531, 2015.
135. Fukuda, T., et al. *Efficient Knowledge Distillation from an Ensemble of Teachers*. in *Interspeech*. 2017.
136. Polino, A., R. Pascanu, and D. Alistarh, *Model compression via distillation and quantization*. arXiv preprint arXiv:1802.05668, 2018.
137. Nayak, G.K., et al. *Zero-shot knowledge distillation in deep networks*. in *International Conference on Machine Learning*. 2019. PMLR.
138. Li, H.-T., et al., *Layer-level knowledge distillation for deep neural network learning*. Applied Sciences, 2019. 9(10): p. 1966.
139. Jin, X., et al. *Knowledge distillation via route constrained optimization*. in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.
140. Guo, Q., et al. *Online knowledge distillation via collaborative learning*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.
141. Walawalkar, D., Z. Shen, and M. Savvides. *Online ensemble model compression using knowledge distillation*. in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX* 16. 2020. Springer.
142. Ji, M., et al. *Refine myself by teaching myself: Feature refinement via self-knowledge distillation*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.
143. Vu, D.-Q., N. Le, and J.-C. Wang, *Teaching yourself: A self-knowledge distillation approach to action recognition*. IEEE Access, 2021. 9: p. 105711-105723.
144. Stewart, R., et al., *Optimising Hardware Accelerated Neural Networks with Quantisation and a Knowledge Distillation Evolutionary Algorithm*. Electronics, 2021. 10(4): p. 396.
145. Zoph, B. and Q.V. Le, *Neural architecture search with reinforcement learning*. arXiv preprint arXiv:1611.01578, 2016.
146. Zoph, B., et al. *Learning transferable architectures for scalable image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
147. Pham, H., et al. *Efficient neural architecture search via parameters sharing*. in *International conference on machine learning*. 2018. PMLR.

148. Saikia, T., et al. *Autodispnet: Improving disparity estimation with automl*. in *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
149. Tan, M., et al. *Mnasnet: Platform-aware neural architecture search for mobile*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.
150. Baker, B., et al., *Designing neural network architectures using reinforcement learning*. arXiv preprint arXiv:1611.02167, 2016.
151. Stanley, K.O. and R. Miikkulainen, *Evolving neural networks through augmenting topologies*. *Evolutionary computation*, 2002. **10**(2): p. 99-127.
152. Real, E., et al. *Aging evolution for image classifier architecture search*. in *AAAI conference on artificial intelligence*. 2019.
153. Liu, C., et al. *Progressive neural architecture search*. in *Proceedings of the European conference on computer vision (ECCV)*. 2018.
154. Xu, Y., et al. *Renas: Relativistic evaluation of neural architecture search*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.
155. Brock, A., et al., *Smash: one-shot model architecture search through hypernetworks*. arXiv preprint arXiv:1708.05344, 2017.
156. Cai, H., et al., *Once-for-all: Train one network and specialize it for efficient deployment*. arXiv preprint arXiv:1908.09791, 2019.
157. Xia, X., et al. *Progressive automatic design of search space for one-shot neural architecture search*. in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022.
158. Zhang, L.L., et al. *Fast hardware-aware neural architecture search*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020.
159. Liu, P., et al. *MemNAS: Memory-efficient neural architecture search with grow-trim learning*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.
160. Pacini, F., et al., *Design and Evaluation of CPU-, GPU-, and FPGA-Based Deployment of a CNN for Motor Imagery Classification in Brain-Computer Interfaces*. *Electronics*, 2024. **13**(9): p. 1646.
161. Ma, X., et al. *Fpga-based rapid electroencephalography signal classification system*. in *2019 IEEE 11th International Conference on Advanced Infocomm Technology (ICAIT)*. 2019. IEEE.
162. Lee, S.-Y., et al., *RISC-V CNN coprocessor for real-time epilepsy detection in wearable application*. *IEEE transactions on biomedical circuits and systems*, 2021. **15**(4): p. 679-691.
163. Huang, Y.-D., et al. *An edge AI system-on-chip design with customized convolutional-neural-network architecture for real-time EEG-based affective computing system*. in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 2019. IEEE.

164. Goldberger, A.L., et al., *PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals*. *circulation*, 2000. **101**(23): p. e215-e220.
165. Kawabata, N., *Nonstationary power spectrum analysis of the photic alpha blocking*. *Kybernetik*, 1972. **12**(1): p. 40-44.
166. Roy, Y., et al., *Deep learning-based electroencephalography analysis: a systematic review*. *Journal of neural engineering*, 2019. **16**(5): p. 051001.
167. León, J., et al., *Deep learning for EEG-based Motor Imagery classification: Accuracy-cost trade-off*. *Plos one*, 2020. **15**(6): p. e0234178.
168. Lawhern, V.J., et al., *EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces*. *Journal of neural engineering*, 2018. **15**(5): p. 056013.
169. Xie, Y. and S. Oniga, *A Review of Processing Methods and Classification Algorithm for EEG Signal*. *Carpathian Journal of Electronic & Computer Engineering*, 2020. **12**(3).
170. Liu, Y., et al., *Automatic building extraction on high-resolution remote sensing imagery using deep convolutional encoder-decoder with spatial pyramid pooling*. *IEEE Access*, 2019. **7**: p. 128774-128786.
171. Ouyang, Z., et al., *Deep CNN-based real-time traffic light detector for self-driving vehicles*. *IEEE transactions on Mobile Computing*, 2019. **19**(2): p. 300-313.
172. Yu, Y., et al., *Vision-based concrete crack detection using a hybrid framework considering noise effect*. *Journal of Building Engineering*, 2022. **61**: p. 105246.
173. Yu Xie, S.O., Tamás Majoros, *Comparison of EEG Data Processing Using Feedforward and Convolutional Neural Network*, in *Conference on Information Technology and Data Science 2020*, A.H. István Fazekas, Tibor Tómacs, Editor. 2020: Debrecen, Hungary. p. 279-289.
174. Lotte, F., et al., *A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update*. *Journal of neural engineering*, 2018. **15**(3): p. 031005.
175. Xie, Y., T. Majoros, and S. Oniga, *FPGA-based hardware accelerator on portable equipment for EEG signal patterns recognition*. *Electronics*, 2022. **11**(15): p. 2410.
176. Li, F., et al., *A novel simplified convolutional neural network classification algorithm of motor imagery EEG signals based on deep learning*. *Applied Sciences*, 2020. **10**(5): p. 1605.
177. Freeman, W.J., *Origin, structure, and role of background EEG activity. Part 2. Analytic phase*. *Clinical Neurophysiology*, 2004. **115**(9): p. 2089-2107.
178. Peterson, V., et al., *A feasibility study of a complete low-cost consumer-grade brain-computer interface system*. *Heliyon*, 2020. **6**(3).

179. Tangermann, M., et al., *Review of the BCI competition IV*. Frontiers in neuroscience, 2012. **6**: p. 55.
180. Khademi, Z., F. Ebrahimi, and H.M. Kordy, *A transfer learning-based CNN and LSTM hybrid deep learning model to classify motor imagery EEG signals*. Computers in biology and medicine, 2022. **143**: p. 105288.
181. Zoph, B., et al., *Transfer learning for low-resource neural machine translation*. arXiv preprint arXiv:1604.02201, 2016.
182. Wan, Z., et al., *A review on transfer learning in EEG signal analysis*. Neurocomputing, 2021. **421**: p. 1-14.
183. Dai, G., et al., *A CNN with hybrid convolution scale for EEG motor imagery classification.*, 2020, 17. DOI: <https://doi.org/10.1088/1741-2552/ab405f>. PMID: <https://www.ncbi.nlm.nih.gov/pubmed/31476743>: p. 016025.
184. Wu, H., et al., *A parallel multiscale filter bank convolutional neural networks for motor imagery EEG classification*. Frontiers in neuroscience, 2019. **13**: p. 1275.
185. Raza, H., et al., *Covariate shift estimation based adaptive ensemble learning for handling non-stationarity in motor imagery related EEG-based brain-computer interface*. Neurocomputing, 2019. **343**: p. 154-166.
186. Xu, C., et al., *Two-level multi-domain feature extraction on sparse representation for motor imagery classification*. Biomedical Signal Processing and Control, 2020. **62**: p. 102160.
187. Lee, H.K. and Y.-S. Choi, *Application of continuous wavelet transform and convolutional neural network in decoding motor imagery brain-computer interface*. Entropy, 2019. **21**(12): p. 1199.
188. Yang, J., S. Gao, and T. Shen, *A two-branch CNN fusing temporal and frequency features for motor imagery EEG decoding*. Entropy, 2022. **24**(3): p. 376.
189. Pei, Y., et al., *Data Augmentation: Using Channel-Level Recombination to Improve Classification Performance for Motor Imagery EEG*. Front Hum Neurosci, 2021. **15**: p. 645952.
190. Kim, S.K., et al., *Intrinsic interactive reinforcement learning—Using error-related potentials for real world human-robot interaction*. Scientific reports, 2017. **7**(1): p. 1-16.
191. Le Guennec, A., S. Malinowski, and R. Tavenard. *Data augmentation for time series classification using convolutional neural networks*. in *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2016.
192. Li, Y., et al., *Human emotion recognition with electroencephalographic multidimensional features by hybrid deep neural networks*. Applied Sciences, 2017. **7**(10): p. 1060.
193. Wang, F., et al. *Data augmentation for eeg-based emotion recognition with deep convolutional neural networks*. in *International conference on multimedia modeling*. 2018. Springer.

194. Xu, G., et al., *A deep transfer convolutional neural network framework for EEG signal classification*. IEEE Access, 2019. 7: p. 112767-112776.
195. Majidov, I. and T. Whangbo, *Efficient classification of motor imagery electroencephalography signals using deep learning methods*. Sensors, 2019. **19**(7): p. 1736.
196. Zhang, R., et al., *A novel hybrid deep learning scheme for four-class motor imagery classification*. Journal of neural engineering, 2019. **16**(6): p. 066004.
197. Ma, X., et al., *DWT and CNN based multi-class motor imagery electroencephalographic signal recognition*. J Neural Eng, 2020. **17**(1): p. 016073.
198. Dai, G., et al., *HS-CNN: a CNN with hybrid convolution scale for EEG motor imagery classification*. J Neural Eng, 2020. **17**(1): p. 016025.
199. Crockett, L.H., et al., *The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC*. 2014: Strathclyde Academic Media.
200. Wang, E., J.J. Davis, and P.Y. Cheung. *A PYNQ-based framework for rapid CNN prototyping*. in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 2018. IEEE.
201. Nane, R., et al., *A survey and evaluation of FPGA high-level synthesis tools*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015. **35**(10): p. 1591-1604.
202. Zhong, G., et al. *Lin-analyzer: A high-level performance analysis tool for FPGA-based accelerators*. in *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2016. IEEE.
203. Alle, M., A. Morvan, and S. Derrien. *Runtime dependency analysis for loop pipelining in high-level synthesis*. in *Proceedings of the 50th Annual Design Automation Conference*. 2013.
204. Sledevic, T. *Adaptation of convolution and batch normalization layer for CNN implementation on FPGA*. in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. 2019. IEEE.
205. El-Sawy, A., H. El-Bakry, and M. Loey. *CNN for handwritten arabic digits recognition based on LeNet-5*. in *International conference on advanced intelligent systems and informatics*. 2016. Springer.
206. YIN Zhen-yu et al., *Design and implementation of Convolution Neural Network Unit Based on Zynq Platform*. Journal of Chinese Computer Systems, 2021. **1-6**.
207. González, E., W.D. Villamizar Luna, and C.A. Fajardo Ariza, *A Hardware Accelerator for the Inference of a Convolutional Neural network*. Ciencia E Ingenieria Neogranadina, 2020. **30**(1): p. 107-116.

Appendix: Publication list

- [J1] Xie, Yu, Tamás Majoros, and Stefan Oniga. “FPGA-Based Hardware Accelerator on Portable Equipment for EEG Signal Patterns Recognition.” *Electronics* 11.15 (2022): 2410.
- [J2] Xie, Yu, and Stefan Oniga. “A Review of Processing Methods and Classification Algorithm for EEG Signal.” *Carpathian Journal of Electronic & Computer Engineering* 12.3 (2020).
- [J3] Xie, Y.; Oniga, S. Classification of Motor Imagery EEG Signals Based on Data Augmentation and Convolutional Neural Networks. *Sensors* 2023, 23, 1932. <https://doi.org/10.3390/s23041932>.
- [J4] Xie, Yu, and Stefan Oniga. 2024. “Enhancing Motor Imagery Classification in Brain–Computer Interfaces Using Deep Learning and Continuous Wavelet Transform” *Applied Sciences* 14, no. 19: 8828. <https://doi.org/10.3390/app14198828>.
- [J5] Xie, Yu, and Stefan Oniga. 2024. “A Comprehensive Review of Hardware Acceleration Techniques and Convolutional Neural Networks for EEG Signals” *Sensors* 24, no. 17: 5813. <https://doi.org/10.3390/s24175813>.
- [C1] X. Yu, T. Majoros and S. Oniga, “Hardware Implementation of CNN Based on FPGA for EEG Signal Patterns Recognition,” 2021 International Conference on e-Health and Bioengineering (EHB), 2021, pp. 1-4, doi: 10.1109/EHB52898.2021.9657679.
- [C2] Yu Xie, Stefan Oniga, Tamás Majoros, Comparison of EEG Data Processing Using Feedforward and Convolutional Neural Network, 1st Conference on Information Technology and Data Science, Debrecen, Hungary, November 6–8, 2020, pp. 279-289, ISSN 1613-0073
- [C3] Y. Xie and S. Oniga, “Cross-task EEG Signal Classification Method Based on Transfer Learning,” 2025 26th International Carpathian Control Conference (ICCC), Starý Smokovec, High Tatras, Slovakia, 2025, pp. 1-4, doi: 10.1109/ICCC65605.2025.11022881.