NAZARBAYEV
UNIVERSITY

Robust Data-driven Predictive Model for Brain-Computer Interface

by

Irina Dolzhikova

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Science Engineering and Technology

Date of Completion
May, 2024

Robust Data-driven Predictive Model for Brain-Computer Interface

by
Irina Dolzhikova

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Science Engineering and Technology

School of Engineering and Digital Sciences
School of Sciences and Humanities
Nazarbayev University

May, 2024

Supervised by
Amin Zollanvari
Berdakh Abibullaev
Reza Sameni

Declaration

I, Irina Dolzhikova, declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the author's original work. The thesis has not been previously submitted to this or any other university for a degree and does not incorporate any material already submitted for a degree.

Signature:

Date:

Abstract

A Brain-Computer Interface (BCI) system enables communication and control between a user and an external device without relying on peripheral and muscular activity. Effective control of such a device hinges on accurately recognizing and decoding intricate brain activity patterns generated by the user. The goal of this PhD project is to develop a robust model for predicting human mental intentions using electroencephalography (EEG) signals. EEG, a widely used non-invasive method for monitoring brain activity, is considered due to its ethical considerations, relatively low cost, and its ability to provide a high temporal resolution of received signals. The robustness of the system is verified based on the classification accuracy with respect to the previously unknown subjects such that the performance of subject-independent (SI) BCI system could be evaluated. An essential challenge in BCI research is developing a classifier capable of interpreting users' mental states from EEG data collected from independent subjects. The focus on SI classification is justified because it can lead to BCIs that eliminate the need for individual calibration processes. Over the past few years, deep neural networks (DNNs) in general, and in particular Convolutional Neural Networks (CNNs), have shown impressive training efficiency and performance, leading to the development of state-of-the-art architectures for accurate EEG classification. In this Thesis, to further enhance the performance of the CNN in SI classification, multi-subject ensemble CNN (MS-En-CNN) models are designed. These are the ensembles of CNN classifiers where each base classifier is built using data aggregated from multiple subjects. Based on the distribution of subject-specific data for training and tuning the base learners of the ensemble, three design strategies for MS-En-CNN are introduced: Subject-Specific Training and Model Selection (SS-TM), Subject Pairs Training and Model Selection (SP-TM), and Delete-a-Subject-Jackknife (DASJ) approach. The predictive performance of the proposed techniques is evaluated across two BCI paradigms, namely motor imagery (MI) and P300, using various publicly available datasets. Empirical results show that with any of the presented strategies constructing MS-En-CNN leads to a significantly better SI classification performance with respect to the average performance of the base CNN classifiers. Moreover, MS-En-CNN notably enhances average classification accuracy compared to a single CNN trained on pooled data from training subjects. Among the three strategies, the latter approach, a jackknife-inspired deep learning technique, emerges as the most promising one. It is then benchmarked against state-of-the-art methods, highlighting its superior performance in single-trial SI classification. While these results show potential for datasets with a small number of subjects, addressing computational requirements for large-scale datasets involves extending this approach through the consideration of K-fold cross-validation (CV). In this extended approach, instead of deleting a single subject to form a jackknife sample, a group of K subjects is set aside. On one of the largest MI datasets a K-fold CV-based MS-En-CNN demonstrated a statistically significant improvement ($p < 0.001$) over the best previously reported results. In addition to MS-En-CNN, proven as a simple yet effective method to enhance the performance of existing CNN models, a new adaptive boosting strategy on the basis of CNN base classifiers (AdaBoost-CNN) with iterative oversampling is proposed. This innovative approach is contrasted with the conventional sample reweighting method, showcasing its potential. Encouraged by promising results, the AdaBoost-CNN warrants further investigation. Overall, this study highlights the effectiveness of MS-En-CNN and AdaBoost-CNN and offers valuable insights that pave the way for further advancements in SI classification within BCI applications.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is a Brain Computer Interface (BCI)?

1.1.1 Short Overview

Brain Computer Interface (BCI) has emerged as a cutting-edge technology with the aim to establish an alternative communication channel between humans and computers or control devices, bypassing the need for peripheral or muscular engagement [1]. Effectively operating the device requires the BCI system to precisely identify complex brain activity patterns generated by the user and convert them into actionable commands [2].

1.1.2 Applications

The original idea behind developing BCIs was related to medical applications and assistive technology. They were originally conceived as a means to help individuals with severe motor disabilities or neurological conditions regain communication and control abilities. The primary focus was on developing technology that could enable people who are paralyzed or unable to communicate through traditional means to interact with the world using their brain activity. Currently, BCI systems find their applications in various domains beyond medicine. This include neuromarketing and advertisement [3, 4, 5], education [6] and emotional intelligence [7, 8], games and entertainment [9, 10, 11], and security and authentication [12, 13].

BCIs could provide real-time feedback on a user's emotional state, helping individuals recognize and manage their emotions more effectively [14]. This could be particularly beneficial for individuals dealing with conditions like anxiety, depression, or stress. In the area of education, emotional states could be monitored for the BCIs to adapt learning materials and approaches based on the learner's emotional engagement. This has the potential to optimize learning experiences and outcomes [15].

1.2 Brain Signal Acquisition

Process of brain signal acquisition is a fundamental part of any BCI system. Throughout the years numerous techniques for acquiring the signals have been investigated. These techniques could be generally divided into two categories: invasive and non-invasive methods. Invasive methods required the electrodes to be surgically implanted either within the user's brain or placed on the outer surface of the brain. On the other hand, non-invasive technologies involve measuring brain activity using external sensors, without the need for any surgical procedures [1].

1.2.1 Invasive

The main benefit in utilizing the invasive methods for brain signals recording lies in their high temporal and spatial resolution, which in turn enhances the quality of the acquired signal and its signal-to-noise ratio. However, such a great advantage cannot compensate the issues these techniques imply. The primary concerns revolve around the requirement for surgical intervention and its potential impact on the patient, such as the risk of infections, bleeding, and tissue damage.

Intracortical (IC) acquisition

The most intrusive method of data acquisition is referred to as the intracortical (IC) technique, where the electrode (or arrays of electrodes) is surgically inserted under the cortex surface of the brain. Placing the electrodes at specific locations restricts the coverage area, while the process of adjustment of the electrodes after their implantation is challenging. Moreover, long-term electrode stability is questionable as the brain tissue around the placed electrodes may undergo changes, for example due to cell death or elevated tissue resistance.

Due to involved risks, there are limited number of IC-based studies, most of which consider monkeys as the subjects of the experiments [16]. A study involving human subjects while employing IC acquisition was conducted by Pandarinath *et al.* [17], who developed a high-performance communication system with point-and-click commands. The system allowed patients with amyotrophic lateral sclerosis (ALS) to control a cursor by recording the motor intention from 96-channel microelectrode array.

Electrocorticography (ECoG)

A less invasive means of capturing brain signals is electrocorticography (ECoG). It involves the implantation of the electrodes over the cortex surface, thereby retaining potential risks from surgical procedure. Being placed relatively close to the source of the signal, electrode strips allow to capture a signal with high resolution. In [18] ECoG (along with electroencephalogram-based) signals were used for motor-imagery-based BCI. It was shown that while distinguishing signals from healthy people is relatively straightforward, it is hard to classify the signals from paralyzed patients (using the same methods). Elghrabawy and Wahed [19] considered finger flexion classification problem using ECoG signal. They employed shift invariant wavelet decomposition and multi-taper frequency spectrum for feature extraction and multilayer perceptron and pace regression for classification. Wang *et al.* using Naive Bayes and Support Vector Machines (SVMs) predicted the semantic information from ECoG [20].

1.2.2 Non-invasive

Invasive techniques require surgical intervention, thereby entailing significant risks. In contrast, non-invasive acquisition techniques provide a safer and more accessible approach to interfacing with the brain. These non-invasive methods encompass a range of technologies such as electroencephalography (EEG), functional magnetic resonance imaging (fMRI), and magnetoencephalography (MEG), offering promising opportunities for advancing BCI research and applications. The fMRI and fNIRs methods record metabolic signals, while the EEG and MEG capture electrical and magnetic signals, respectively.

Magnetoencephalography (MEG)

MEG measures magnetic fields resulting from inherent electrical currents in the brain [1]. MEG signals have the potential to be affected by external magnetic fields like the Earth's, necessitating a controlled laboratory setup with shielding and dedicated equipment. In contrat to electric fields, MEG signals experience lower distortion from the skull layer.

Functional magnetic resonance imaging (fMRI)

The fMRI-based method monitors the neural activity by detecting the changes in blood flow. It relies on the principle that any engagement of a brain area necessitates an augmented inflow of blood. When neural activity increases, it initiates the changes in blood circulation and oxygen levels, resulting in modifications to the blood-oxygen-level-dependent (BOLD) response. The brain images are acquired using a specialized scanner that generates magnetic fields, which is not only expensive but also heavy and lacks portability. While the temporal resolution is low in fMRI, there is a very good spatial resolution which allows measuring signals from deep regions within the brain.

Functional near-infrared spectroscopy (fNIRS)

Similar to fMRI, fNIRS captures the metabolic type of the signal. However, in this case, the near-infrared range of light (as opposed to the magnetic method utilized in fMRI) is used to measure the dynamics of blood flow. Specifically, fNIRS operates based on the idea that oxygenated hemoglobin and deoxygenated hemoglobin have higher light absorbance compared to skull and scalp. A limitation of fNIRS is its inability

Table 1.1: Summary of the characteristics of various brain signal acquisition methods.

Method	Invasive	Risk	Signal	Resolution		SNR	Portability	Cost
				Spatial	Temporal			
IC	Yes	High	Electrical	Very high	High	High	Mediate	High
ECoG	Yes	High	Electrical	High	High	High	Mediate	High
EEG	No	Low	Electrical	Low	Mediate	Low	High	Low
fNIRS	No	Low	Metabolic	Mediate	Low	Low	High	Low
fMRI	No	Low	Metabolic	High	Low	Mediate	Low	High
EOG	No	Low	Electrical	Low	Mediate	Mediate	High	Low
MEG	No	Low	Magnetic	Mediate	High	Low	Low	High

to measure cortical activity beyond a depth of 4 cm within the brain, which is due to restrictions in light emitter power and spatial resolution [21].

Electroencephalogram (EEG)

The first human EEG signal was recorded by Berger [22] in 1924 [23]. Currently, it is the most frequently employed non-invasive method for measuring brain activities. EEG is a technique used for electrophysiological monitoring of the electrical activity inside the brain by placing electrodes on the scalp. Monitoring method records the voltage fluctuations that occur from ionic current within the neurons of the brain. EEG-based data acquisition method has strong temporal resolution, but is limited by its low spatial resolution and signal-to-noise ratio (SNR).

1.2.3 Summary

The summary of the characteristics of various brain signal acquisition methods is presented in the Table 1.1 [1, 21]. In short, due to the risks involved, invasive BCIs are generally restricted to carefully selected patients and research participants, while the non-invasive methods (EEG and fNIRS, in particular) are the most practical and are widely used for various applications.

As a part of this Thesis only EEG based datasets were considered as EEG carries no clinical risk and can be measured using portable and cost-effective devices.

1.3 Subject-Independent Classification for BCI: Why?

The majority of current research focuses on subject-dependent scenarios, where both training and test data involve the observations from the same individual. Consequently, a calibration session becomes necessary before a BCI system can be used by a new user. This calibration process is time-consuming and requires individual recalibration for each new subject and each usage. In contrast to this, it is highly desirable to develop the subject-independent system, which involves training using data from known subjects and applying it directly to new users without prior adaptation.

1.4 Problem Statement

Given the same experimental framework, EEG signals exhibit substantial variability not only among different subjects, but even within the same user across recording sessions. Thus, developing an accurate subject-dependent system is challenging, not to mention the subject-independent scenario.

This Thesis aims to explore and develop novel approaches for subject-independent BCIs. The research investigates the design and training ensemble classifiers to create robust BCI systems that can effectively handle subject-independent data.

1.5 Hypothesis

Hypothesis behind this study is that in classifying an individual's mental intention based on EEG waveforms, the discriminative features of collected measurements lie in a *latent space* that is captured by some classifiers trained using measurements collected from previous trials of the same individual and/or other subjects.

1.6 Organization of the Thesis

This Thesis is organized as follows:

- Chapter 1 was the introduction to the study, presenting the concept of BCI, its applications, different methods of brain signal acquisition, and highlighting the importance of conducting the research in the subject-independent scenario. It also formulated the hypothesis behind the current study.

- Chapter 2 provides the general background on EEG-based BCI paradigms.

- Chapter 3 documents the literature review that summarizes previous CNN architectures and ensemble methods based works in the context of EEG classification in general, and subject-independent classification in particular.

- Chapter 4 outlines the methodologies employed in this study and introduces the datasets utilized to validate the proposed designs.

- Chapter 5 defines the base classifiers that constitute the proposed ensemble.

- Chapter 6 presents the idea of multi-subject ensemble CNN and its possible strategies for implementations. This chapter shows both, the designed approaches and the experimental results that were achieved on publicly available datasets.
- Chapter 7 demonstrates the performance of the jackknife-inspired approach with the state-of-the-art base classifiers.
- Chapter 8 shows the promising K-fold cross-validation approach for large datasets.
- Chapter 9 explores an alternative ensemble approach featuring sequentially trained CNN base classifiers, specifically referred to as AdaBoost-CNN.
- Chapter 10 offers discussion and summary of the achieved results.
- Chapter 11 is dedicated to the conclusions and possible future work.

Chapter 2

EEG-based BCI

In the realm of EEG-based BCI research, various paradigms are explored to decode neural signals and establish effective communication pathways between the brain and external devices. Examples of such paradigms include event-related potentials (ERPs), event-related synchronization/desynchronization (ERS/ERD), error potentials, sensorimotor rhythms, slow cortical potentials, steady-state visual evoked potential, etc. This Thesis focuses on two commonly used BCI paradigms: P300, which is an important component of ERP and motor imagery (MI), which is closely related to ERD, particularly concerning the mental rehearsal of motor actions. These paradigms are associated with the endogenous process, which is the internally generated cognitive or mental activities that are initiated and controlled by the individual, rather than being driven by external stimuli. Notably, although both are associated with the endogenous process, P300 is an evoked type of activity, while MI is spontaneous. This distinction arises from the fact that P300 is elicited by external stimuli, whereas MI occurs independently, without any external trigger. Further elaboration on these brain activity patterns will be provided in the subsequent sections, where first the general characteristics of EEG are considered and the main principles of the work of EEG-based BCI are presented, followed by the discussion of P300 and MI paradigms.

2.1 General Principles

2.1.1 EEG characteristics

EEG activity has quite small signal intensity that is measured in microvolts. According to the signal frequency, the EEG waves are categorized into five groups: delta, theta, alpha, beta, and gamma rhythms [24]. The summary of the EEG rhythms with corresponding frequency subbands and behavioral states is demonstrated in Table 2.1. In addition to these five basic groups of brain waves there is also mu rhythm, which is the EEG rhythm with the signal amplitude and frequency of approximately 50 μV and 10 Hz, respectively. Although the mu rhythms are in the similar frequency range as alpha waves, the nature of these two waves is physiologically different. There is a strong correlation between the motor activities and mu waves, which are predominantly present in the motor cortex region of the cerebral cortex.

Table 2.1: Rhythms of EEG signals and corresponding behavioral states

EEG rythms	**Frequency subbands**	**EEG behavioral states**
Delta	0-4 Hz	Deep/Restful sleep
Theta	4-8 Hz	Visual imagery, light sleep, deep meditation
Alpha	8-12 Hz	Low level of stimulation: relaxed awake or drowsy states without focused attention or vigilance (mindless states)
Beta	12-30 Hz	High level of stimulation: awake alert states with the focus or attention on the problem solving; dream or rapid eye movement sleep phases
Gamma	>30 Hz	State of perception and consciousness
Mu	around 10 Hz	States of motor activities

2.1.2 Basic framework for EEG-based BCI

The general process involved in EEG-based BCI is illustrated in Fig.2.1. First, the EEG signals are recorded by placing the electrodes on the user's head. The acquired signals are then preprocessed with the aim of eliminating noise, artifacts, and unwanted frequencies. This improves the clarity and accuracy of the data. Next is the feature extraction stage, where specific patterns are extracted from EEG data. These patterns are associated with distinct paradigms, such as P300 and MI. For example, when extracting the features for P300-based applications, the main focus is on identifying the specific neural response occurring around 300 milliseconds after the presentation of stimulus. For this, the temporal and amplitude features of EEG are analyzed. In the context of MI-based classification problems, the feature extraction process centers on isolating and quantifying patterns associated with the mental rehearsal of motor actions without actual execution. This involves examining changes in spectral power, particularly in mu and beta frequency bands, over the sensorimotor cortex. The analysis captures the ERD

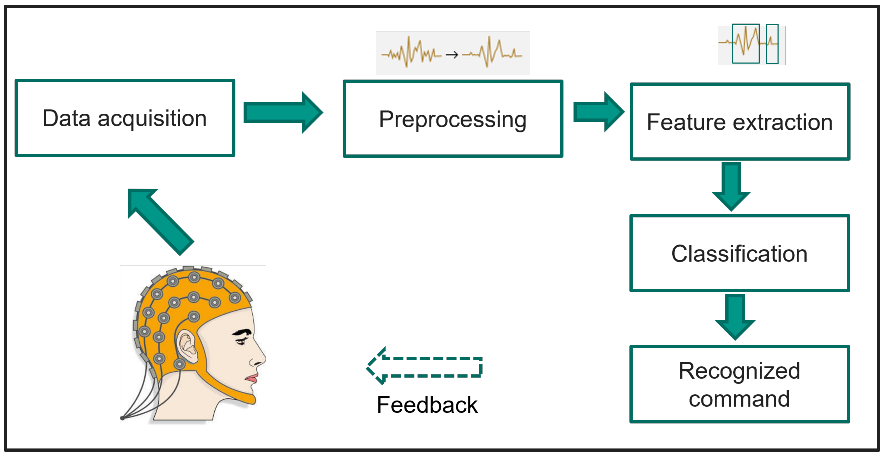

Figure 2.1: General scheme for EEG-based BCI

phenomenon, intricately linked with imagined movements. ERD is characterized by a decrease in the amplitude of specific frequency bands, such as mu and beta, in the sensorimotor cortex. For example, when an individual envisions moving their hand, there is often a reduction in the amplitude of these neural oscillations.

The next step is to classify the extracted features using machine learning algorithms. For this, the classification models are trained to interpret the brain signals and associate them with intended actions or commands. Once the intended action is classified/recognized it is translated to the command for the external device. This can be controlling a cursor on a screen, selecting letters for communication, or moving a prosthetic limb. In addition to above mentioned steps, feedback mechanisms could be integrated into BCI system to inform the user about the decoded results of his intentions. By receiving certain feedback the user can be aided to modulate brain activity, therefore improving the performance of BCI.

The focus of the current study is on designing the classification algorithms that are aimed to accurately decode the P300 and MI-based brain activity patterns. The proposed algorithms are tested on publicly available EEG data. In a binary classification scenario involving P300-based brain activity patterns, the goal is to distinguish between two neural responses: those correlated with the occurrence of the P300 event (target) and those lacking the P300 event (non-target). Regarding MI-based brain activity patterns, the objective is to differentiate between distinct types of imagined motor actions or tasks. The system is trained to identify and categorize various patterns of brain activity linked to specific intended motor imagery tasks, such as discerning between left-hand movement and right-hand movement.

There are standardized methods for positioning of electrodes on the scalp based on a percentage of head circumference and specific skull landmarks [25]. The 10-10 and 10-20 systems are commonly used electrode placement techniques. The key difference between the two lies in the number of electrodes used and the coverage they provide. Electrodes are placed at locations determined by dividing the head into regions, each representing either 10% or 20% of the total distance between specific landmarks. The 10-10 system is an expansion of the 10-20 system, incorporating additional electrode positions. The positions of most commonly used EEG electrodes are illustrated in Fig.2.2.

2.2 P300 based BCI

P300-based BCI is a type of technology that utilizes the ERP of the human brain for communication and control. ERPs are brain reactions triggered by external sensory, motor, or cognitive events, detected as small changes in voltage within EEG. These responses have been the main focus for creating non-invasive BCI systems. Specifically,

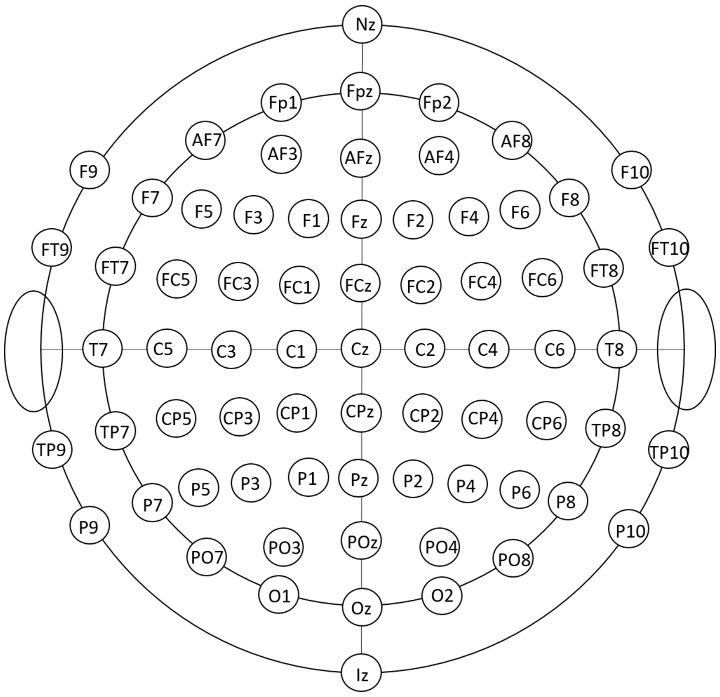

Figure 2.2: EEG electrode positions [25].

the analysis of P300-waveforms has been extensively explored in the majority of BCI research. P300 is a prominent component of ERP, which is observed as a positive voltage change in the brain occurring approximately 300 milliseconds after a person is presented with a stimulus.

One of the widely recognized and popular applications within the realm of EEG-based BCI technology is P300 speller. It was developed by Farwell and Donchin in the 1980s and therefore is also known as Farwell & Donchin style visual speller [26]. The typical visual stimuli character matrix used to evoke P300 ERPs is demonstrated in Fig.2.3. The rows and columns of the grid flash systematically. The user's task is to focus the attention on the desired character. As a result, when the selected character's row or column is flashed, the P300 response is elicited, which can be detected in the user's EEG signals. This way, the person can spell out words or convey messages by selecting characters on the visual matrix using only their brain activity.

2.3 Motor Imagery (MI) based BCI

In the MI-based BCI paradigm, individuals are asked to imagine performing a particular movement without actually engaging in physical action. This mental rehearsal of movement involves activating and simulating brain regions associated with the execution of the intended movement. MI is used as a method for users to control external devices or applications using their brain activity.

While specific details of experimental procedure (such as the number of sessions,

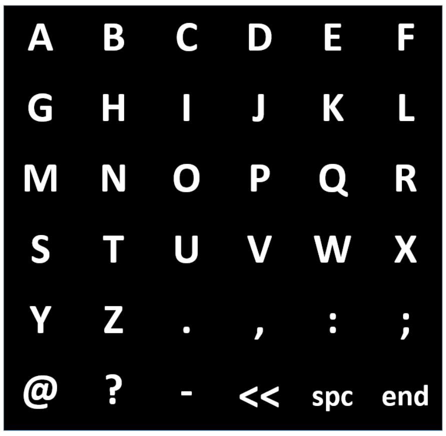

Figure 2.3: Farewell & Donchin style P300 speller.

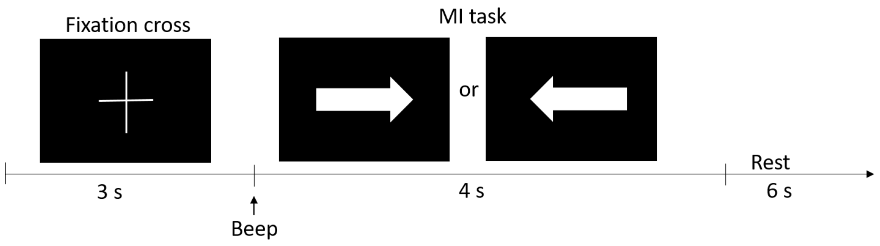

Figure 2.4: Procedure for MI-based BCI.

their duration, number, and types of mental states) may vary, the general protocol for MI-based experiments typically follows a standard format. The subject/user is placed in front of a computer. The user is tasked with mentally envisioning certain movements in response to a visually displayed cue stimulus. The idea of the experimental design is illustrated in Fig.2.4 [27, 28]. Assuming the focus of the experiment is on distinguishing between two mental states, the cue in Fig.2.4 is represented by a left or right arrow, indicating either left-hand or right-hand movement. Each trial is of a fixed duration and it is started with a displayed fixation cross at the center of the monitor. This is followed by a short beep sound, after which a visual cue is demonstrated (for approximately 4 seconds). There are several trials within one session which are repeated after resting period. The whole experiment consists of several sessions, which might be held on the same or different days.

Chapter 3

Literature Review

Various techniques exist for feature extraction, feature selection, and classification in EEG-based BCI systems. Nonetheless, current studies in the BCI domain demonstrate the trend of moving towards employing deep learning methodologies. This shift is driven by the capability of deep learning to execute a sequence of procedures for processing raw input data within one block. This chapter presents a summary of research works that have presented approaches for EEG-based BCI applications. The focus is on deep learning (in particular on Convolutional Neural Networks) and ensemble methods. The last section is dedicated to the methods that have been proposed in the context of subject-independent (SI) classification of EEG.

3.1 Deep Learning for EEG-based BCI

Deep learning (DL) proved to be successful in complex data processing [29]. A number of DL architectures that allow effective preprocessing, feature extraction and classification have been developed for applications related to the images and signals in the form of audio, video and text [30]. However, talking about the EEG-related applications, in the past there was a certain degree of incredulity among the research community that deep learning is the optimal tool for it [30]. The skepticism regarding DL tools in the area of BCI was mostly due to the availability of EEG data and its characteristics [30]. Due to challenges associated with EEG data collection [31], EEG-based datasets are considerably smaller compared to the extensive sample sizes commonly utilized for training purposes in fields such as computer vision and natural language processing [30]. This limitation arises due to the restricted number of recorded trials available for each participant. Moreover, low SNR of the data is another significant problem in the EEG data classification. As a result, it was thought that the performance of the readily available machine learning algorithms might be significantly affected [31], making them inapplicable for EEG processing [30]. Nevertheless, this has not prevented the investigation and eventually, in the last few years, remarkable progress in deep learning research for designing accurate BCIs has been witnessed with the results highlighting the significant potential of DL.

3.2 Convolutional Neural Networks (CNNs) for EEG classification

One of the most frequently used DL architectures for EEG-based applications is the convolutional neural network (CNN). This is due to its regularization framework and its ability to learn the discriminative features from the input data [32]. A number of CNN-based techniques have been recently applied for the purpose of improving the performance of BCI systems. A summary of the EEG-based BCI studies that leverage the advantages of CNNs is provided in Table 3.1, where the following details are given: the proposed architecture designs, the datasets (name of the dataset, type of the BCI paradigm, number of subjects N_s) that were used to verify them, and reported performance.

CNN proved to be an effective method for single-trial ERP detection that combines spatial filtering and classification stages. Cecotti *et al.* [33] presented several CNN-based designs for P300 classification problem. Among these designs, four of the classifiers are single architectures and three multiclassifiers. All of the designs represent variations to the same architecture with two convolutions, subsampling layer and fully connected (FC) layer. The variations are mostly in terms of the input data that is used to train the classifier and base classifiers of the multiclassifier. The authors of [34] proposed a parallel convolutional-linear neural network that combines the static and dynamic energy classification of MI-based EEG through multilayer perceptron (MLP) and CNN, respectively. Schirrmeister *et al.* [35] explored various options in crafting CNN-based structures to interpret tasks involving imagination and execution using unprocessed EEG data. Among the proposed designs, two architectures, namely DeepConvNet and ShallowConvNet, became later popular and were used in other investigations as benchmarks for comparison.

Liu *et al.* [36] proposed the use of batch normalization in CNNs (BN3) to prevent overfitting. Joshi *et al.* [37] designed Convolutional Long Short Term Memory (ConvLSTM) network. In addition, the author represented the input EEG data in the form of a 3D map and implemented CNN model for 3D data (CNN3D). Moreover, the proposed CNN models, along with the BN3 were used as the base classifiers for the ensemble with simple averaging.

Lawhern *et al.* [38] designed a CNN-based architecture called EEGNet. It is a robust architecture that is able to learn a wide range of features for accurate decoding of ERP, as well as oscillatory data (movement-related cortical potential and sensory-motor rhythm signals). The structure comprises three categories of convolutional layers: a traditional layer, a depthwise layer, and a separable layer. The results of the study demonstrated that the EEGNet system does not require augmentation of data for proper performance across different datasets. An instance of CNN architecture with data

Table 3.1: A summary of CNN-based designs for EEG-based BCI. N_s is the number of subjects that participated in the BCI experiment to collect the data for a given dataset. Different performance measures indicate the average values (with the standard deviation in some cases) across all of the subjects and across different folds, where the cross-validation has been done.

Study	**Dataset** **Type of EEG Signals**	**Architecture**	**Performance measures**
Cecotti *et al.* [33], 2010	ERP: BCI III 2 ($N_s = 2$) [46]	<u>CNN-1</u>: Conv + Conv + Subsampling + FC <u>Multiclassifier MCNN</u>: 5 CNN-1 classifiers are trained on different database *Decision making*: average for fusing the output of each classifier	<u>CNN-1</u>: 5/15-epoch RR: 70.00% / 94.50% 5/15-epoch ITR: 12.69 / 8.08 <u>MCNN-1</u>: 5/15-epoch RR: 69.00% / 95.50% 5/15-epoch ITR: 12.40 / 8.25
Cecotti *et al.* [47], 2014	ERP: RSVP1 ($N_s = 8$) RSVP2 ($N_s = 10$) RSVP3 ($N_s = 10$)	CNN is used as a preprocessing method for several classifiers: BLDA, MLP, SVM	AUC: $86.10 \pm 7.3\%$ (RSVP1 using MLP+CNN) $85.00 \pm 6.10\%$ (RSVP2 using BLDA+CNN) $81.90 \pm 5.5\%$ (RSVP3 using SVM+CNN)
Sakhavi *et al.* [34], 2015	MI: BCI IV 2a ($N_s = 9$)	<u>CNN $\Vert$ MLP</u> 3-layered MLP for static energy features + CNN for dynamic energy features (Dropout and ReLU activation functions are used) *CNN Configuration*: Parallel {Conv in time + Avg. Pool + Conv in time + Avg. Pool +MLP} + Concatenation + MLP *Decision making*: max value of each class from MLP and CNN	Acc: 70.60%

Table 3.1: Continued

Study	Dataset Type of EEG Signals	Architecture	Performance measures
Tabar *et al.* [48], 2016	MI: BCI IV 2b ($N_s = 9$) [49] BCI II 3 ($N_s = 1$) [50]	CNN-Stacked AE: 1 conv. layer + 6 $\times$ AE + output layer	Subject-Specific: Acc: $77.6 \pm 2.1\%$ (BCI IV 2b) Kappa: 0.547 ± 0.083 (BCI IV 2b) Acc: 90.00% (BCI II 3) Kappa: 0.800 (BCI II 3) Session-to-session: Acc: 75.10% (BCI IV 2b)
Schirrmeister *et al.* [35], 2017	MI: BCI IV 2a ($N_s = 9$) [51], BCI IV 2b ($N_s = 9$) [49], High Gamma Dataset (HGD) ($N_s = 14$) [35], Mixed Imagery Dataset (MID)	(1) *DeepConvNet inspired by architecture of AlexNet [52]*: 4 convolution-max-pooling blocks + dense softmax classification layer (2) *ShallowConvNet is inspired by FBCSP [40, 41, 42]*: 1 convolution-max-pooling block + dense softmax classification layer (3) *hybrid CNN* - fusion of deep and shallow CNN after the final layer; (4) *Residual CNN* is based on ResNet [53]	DeepConvNet: Acc: 70.9% (BCI IV 2a) Acc: 92.5% (HGD) Acc: 72.2% (MID) Kappa: 0.598 (BCI IV 2b) ShallowConvNet: Acc: 73.7% (BCI IV 2a) Acc: 89.3% (HGD) Acc: 67.7% (MID) Kappa: 0.899 (BCI IV 2b)
Lawhern *et al.* [38], 2018	ERP: P300 ($N_s = 18$), ERN ($N_s = 26$), MRCP ($N_s = 13$); MI: SMR ($N_s = 9$)	*EEGNet inspired by FBCSP [40, 41, 42]*: Temporal Convolution + BN + Depthwise convolution + BN + ELU + Avg. Pooling + Dropout + Separable convolution + BN + ELU + Avg. Pooling + Dropout + Flatten + Dense	AUC: 90.54% (P300)
Liu *et al.* [36], 2018	ERP (P300): BCI III 2 ($N_s = 2$) , BCI II 2b ($N_s = 1$)	*BN3*: BN + Conv + Conv + ReLU + FC + tanh + Dropout + FC + tanh + Dropout + Sigmoid	F-1 Score: 0.54

Table 3.1: Continued.

Study	**Dataset** **Type of EEG Signals**	**Architecture**	**Performance measures**
Joshi *et al.* [37], 2018	ERP (P300): BCI III 2 ($N_s = 2$)	*ConvLSTM*: Recurrent convolution + tanh + Dropout + Recurrent dropout + BN + Recurrent convolution + BN + Dropout + FC layer + Sigmoid *CNN3D* - CNN for 3D data: Conv + ReLU + BN + Dropout + Conv + ReLU + BN + Dropout + Avg. pool + BN + Dense + Sigmoid Ensembles of CNNs	F1-score: 0.54 (ConvLSTM) 0.50 (CNN3D)
Sakhavi *et al.* [44], 2018	MI: BCI IV 2a ($N_s = 9$) [51]	*Feature extraction by FBCSP* Channel-wise CNN (CW-CNN): Convolution + ReLU + Convolution + ReLU + Linear+ ReLU + Linear + Log-SoftMax Channel-wise convolution with channel mixing (C2CM): same as CW-CNN with additional convolution for channel mixing	CW-CNN: Acc: 73.07% Kappa: 0.641 C2CM: Acc: 74.46% Kappa: 0.659
Tang *et al.* [54], 2019	MI: BCI IV 1 ($N_s = 4$) [50]	Sparse AE - CNN: Single Sparse AE + 1 conv. layer+ Pooling + logistic regression layer	Acc: 92% MSE: 0.575
Zhu *et al.* [45], 2019	MI: MI data ($N_s = 25$) [45] BCI IV 2b ($N_s = 9$) [49]	Separated Channel convolutional network (SCCN)	Acc: 73.00% (MI data) ITR: 3.33 (MI data) Acc: 64.00% (BCI IV 2b) ITR: 0.83 (BCI IV 2b)

Table 3.1: Continued.

Study	**Dataset** **Type of EEG Signals**	**Architecture**	**Performance measures**
Qiao *et al.* [55], 2019	MI: BCI IV 2a ($N_s = 9$) [51]	IncepCNN-BGRU: Inception module (incorporates 4 levels of convolutional kernels) + Conv + ReLU + Conv + ReLU + Pooling + Conv + ReLU + Conv + ReLU + Pooling + FC +BGRU + Softmax	Acc: 76.62%
Dai *et al.* [56], 2019	MI: BCI IV 2b ($N_s = 9$) [49] BU data ($N_s = 5$) [56]	CNN-VAE: 1 conv. layer + ReLU+ max pool + FC	Kappa: 0.564 ± 0.065 (BCI IV 2b) Kappa: 0.603 ± 0.067 (BU data)
Zhang *et al.* [57], 2019	MI: BCI IV 2a ($N_s = 9$) [51]	CNN-LSTM: For each time window (5 time windows): {one-versus-rest *FBCSP for feature extraction* + Conv +Max Pool + Conv +Max Pool + Conv +Max Pool+ Dropout + Reshape} + LSTM	Subject-specific: Kappa: 0.8 Merged data: Acc: 84.00% Kappa: 0.81
Ingolfsson *et al.* [58], 2020	MI: BCI IV 2a ($N_s = 9$) [51]	*EEG-TCNet inspired by EEGNet [38]*: EEGNet + TCN + FC	Acc: $77.35 \pm 11.57\%$
Dai *et al.* [39], 2020	MI: BCI IV 2a ($N_s = 9$) [51], BCI IV 2b ($N_s = 9$) [49]	*HS-CNN*: Three filter banks + Convolution in time + Convolution in space + Max-pool + Flatten + Fully Connected (FC) + FC	Acc: 91.57 (BCI IV 2a) Acc: 87.6 (BCI IV 2b)
Musallam *et al.* [59], 2021	MI: BCI IV 2a ($N_s = 9$) [51], HGD ($N_s = 14$)	*TCNet-Fusion inspired by EEG-TCNet*: concatenation of the outputs from EEGnet and TCN	Acc: $83.73 \pm 9.79\%$ (BCI IV 2a) Acc: 94.41% (HG)
Bang *et al.* [60], 2022	MI: BCI IV 2a ($N_s = 9$), BCI IV 2b ($N_s = 9$), KU data ($N_s = 54$)	*3D-CNN*: 3D conv layer + 3D conv layer + FC (ReLU activation)	Acc: BCI IV 2a: $87.15\% \pm 7.31\%$ BCI IV 2b: $75.85\% \pm 12.80\%$ KU data: $70.37\% \pm 17.09\%$

augmentation method is illustrated by Dai *et al.* [39], who designed a hybrid scale (HS) CNN. ShallowNet and EEGNet are some examples of the architectures that were inspired by the Filter Bank Common Spatial Patterns (FBCSP) method [40, 41, 42], which is the extension of the Common Spatial Pattern (CSP) algorithm [43]. Other examples of CNN-based studies with the use of FBCSP/CSP include [44], [45].

More recently, Bang *et al.* [60] designed a 3D convolutional layers-based framework with spatio-spectral feature representation. Other examples of CNN-based classification methods for EEG-based BCI are [61]. Unlike most of the common CNN-based designs, Cecotti *et al.* [47] considered CNN as a preprocessing method to evaluate different classifiers (Bayesian Linear Discriminant Analysis (Bayesian LDA), MLP, Support Vector Machines (SVMs)).

Besides purely CNN-based architectures, hybrid models incorporating CNN and other deep structures are extensively used for the EEG signal interpretation and classification. Such an example is presented by Tabar *et al.* [48], who classified the MI EEG signals by investigating the combination of CNN and stacked autoencoders (AE). A combination of CNN architecture with sparse autoencoder (SAE) is exploited in [54]. Other examples of hybrid CNN models include integration of CNN and variational AE [56], CNN and Bidirectional Gated Recurrent Unit (BGRU) [55], convolutional layers and structure of long-term short-term memory (LSTM) network [62, 37, 57].

In addition to regular CNN architectures, nowadays temporal convolutional networks (TCNs) are getting a lot of attention in various applications. Recently, being inspired by the architecture of EEGNet, Ingolfsson *et al.* [58] designed the EEG-TCNet, which represents the combination of regular convolutional layers, as in EEGNet, (temporal, depthwise and separable convolutions) and TCN with its causal and dilated convolutions, and residual blocks. Such an architecture allowed to improve the performance of its predecessor (EEGNet) on average by 4.95%, while preserving its compactness (in fact, EEG-TCNet requires 1.6 times more number of trainable parameters, but 6.3 million less (twice less) multiply-accumulate (MAC) operations per inference). The performance of EEG-TCNet, was further improved by Musallam *et al.* [59], who concatenated the outputs of EEGNet and TCN.

3.3 Ensemble Methods for EEG-based BCI

The concept of ensemble learning is renowned for its ability to exhibit enhanced generalization and performance by combining a collection of individual models. In this section the research works that have been conducted in the area of EEG-based BCI on ensemble techniques are summarized. Although the focus of the current study is on the CNN-based ensemble methods, for comprehensiveness of the literature review here the designs for non-CNN based designs are presented as well. This will give an idea of the general methods that are used to ensemble different models. The summary of the works is presented in Table 3.2.

The first attempts to demonstrate the effectiveness of the ensemble over the single classifier in the context of EEG classification were presented by Rakotomamonjy *et al.* [63] and Sun *et al.* [64]. The former study used the ensemble of linear SVMs, while the latter evaluated bootstrap aggregation (bagging), adaptive boosting (AdaBoost), and random subspace techniques with the k-Nearest Neighbors (KNN), decision tree, and SVM-based base classifiers. To ensure diversity of the ensemble Ramos *et al.* [65] considered different learning paradigms of Probabilistic Neural Networks (PNN), SVM with linear and quadratic kernels, Radial Basis Function Network (RBF), LDA with different discriminant functions, and KNN with the euclidean and mahalanobis distances, cosine and correlation metrics. Rakotomamonjy and Guigue [66] performed bagging of SVM models. Later, on the same dataset, another research group [67] analyzed CNN-based bagging approach, in addition to the bagging ensemble of SVMs. However, the former method (with SVM base classifiers) demonstrated superior 5-trial and 15-trial recognition rates. Other examples of using SVMs as a part of the ensemble can be found in [68], [69]. Salvaris and Sepulveda [70] combined simple linear classifiers, namely Fisher's linear discriminants (FLDs), that are trained on a different partition of the training data. Fazli *et al.* [71, 72, 73] considered various gating functions to combine the outputs of the base classifiers (CSP with matching LDAs). This included the commonly used way of averaging the outputs of individual classifiers (mean), employing different classification methods (KNN, LDA, SVM, Linear Programming Machine (LPM)), quadratic regression with l_1 regularization, and least squares regression.

Ebrahimpour *et al.* [74] using MLPs as the base classifiers have shown that mixture of experts as an ensemble technique outperforms simple averaging and decision templates and stacked generalization. Datta *et al.* [75] performed a comparative study on different types of ensemble architectures, which combined KKN classifier trained with different parameters, different classifiers (KNN, SVM, Naive Bayes (NB)), classifiers of one kind trained on different features. The latest approach with KNN base classifier demonstrated the best performance on a motor imagery task. In [76], Subasi and Qaisar have used Multiscale Principal Component Analysis (MSPCA) for denoising and Wavelet Packet Decomposition (WPD) for feature extraction. The extracted features are further reduced in dimensionality and fed to a classification algorithm. Six different algorithms, namely KNN, C4.5 Decision Tree (C4.5 DT), REP Tree, SVM, Random Tree (RT), and RF, were considered. For each of the above-mentioned algorithms, a separate ensemble is formed on the basis of Rotation Forest (RoF) and Random Subspace Method (RSM). On average (among 5 subjects) the best classification performance was demonstrated by the RoF with KNN base classifier.

Table 3.2: Examples of research works on ensemble learning for EEG-based BCI.

Study	**Dataset**	**Base Classifiers**	**Ensemble**	**Performance measures**
Sun *et al.* [64], 2007	MI ($N_s = 3$), 3 classes	KNN, decision tree (C4.5), SVM	Bagging, AdaBoost, random subspace (RanSub)	Acc: KNN: 53.12% (Bagging); 51.90% (AdaBoost); 56.35% (RanSub) C4.5: 55.98% (Bagging); 55.49% (AdaBoost); 56.13% (RanSub) SVM: 46.47% (Bagging); 57.20% (AdaBoost); 55.20% (RanSub)
Fazli *et al.* [71], 2008	ERD/ERS data ($N_s = 45$)	Predefined filter-bank of temporal filters	Mean (other considered options: highest absolute value, majority vote, median)	Loss (mean): 25%-tile: 3.6 median: 11.2 75%-tile: 30.7
Rakotomamonjy *et al.* [66], 2008	ERP (P300): BCI III 2 ($N_s = 2$)	SVM	Bagging	5-trial RR: 73.5% 15-trial RR: 96.5% (MCCP)
Alzoubi *et al.* [78], 2008	MI: BCI III 3a ($\mathrm{N}_s = 3$)	DT	AdaBoost, bagging, stacking, RF	Target hit rate: 57.31%
Fazli *et al.* [72], 2009	MI ($N_s = 45$)	CSP+LDA classifiers	Mean of the outputs and various forms of regression (l_1 regularization)	Loss: 26.70% (l_1 regularized regression)
Fazli *et al.* [73], 2009	MI ($N_s = 83$)	Subject-dependent CSP+LDA	mean, quadratic regression with l_1 regularization, least square regression, classification algorithms (LDA, SVM, KNN, LPM)	Loss (SVM): 28.7%

Table 3.2: Continued.

Study	**Dataset**	**Base Classifiers**	**Ensemble**	**Performance measures**
Salvaris *et al.* [70], 2009	ERP: BCI II 2b ($N_s = 1$), BCI III 2 ($N_s = 2$)	FLD trained on different data segments	Sum of the each base classifiers scores	RR: 100% (BCI II 2b) 5-trial RR: 71.5% (BCI II 2) 15-trial RR: 95% (BCI II 2)
Ebrahimpour *et al.* [74], 2012	MI: BCI 2005 3a [94]	MLP	Simple averaging, decision templates, mixture of experts, stacked generalization	Acc: 77.91% (mixture of experts)
Chaurasiya *et al.* [69], 2016	ERP: Devanagari script based P300 ($N_s = 9$)	SVMs trained on sequentially partitioned training data	Weighted ensemble of SVMs	5-trial RR: 72.4% 15-trial RR: 91.9%
Ramos *et al.* [65], 2017	MI ($N_s = 3$)	PNN, SVM linear, SVM quadratic, RBF, LDA linear, LDA quadratic, LDA Mahalanobis, KNN euclidean, KNN mahalanobis, KNN cosine and KNN correlation	Majority voting, weighted majority voting	Acc: 95.50%
Kundu *et al.* [68], 2018	ERP: BCI II ($N_s = 1$), BCI III 2 ($N_s = 2$)	SVM; LDA	Weighted average of SVMs; Weighted average of LDAs;	RR: 100% (BCI II); 5-trial RR: 72% (BCI II 2) 15-trial RR: 98% (BCI II 2) (SVM-ensemble)

Table 3.2: Continued.

Study	**Dataset**	**Base Classifiers**	**Ensemble**	**Performance measures**
Joshi *et al.* [37], 2018	ERP (P300): BCI III 2 ($N_s = 2$)	Various combinations of CNN models: BN3, CNN3D, ConvLSTM	Simple averaging	F1-score: 0.51 (BN3 + CNN3D + ConvLSTM)
Barsim *et al.* [67], 2018	ERP (P300): BCI III 2 ($N_s = 2$)	SVM; Different variants of CNNs (conventional CNN, Inception, Xception, and IGC modules)	Bagging of SVMs; Bagging of CNNs	5-trial RR: 76.5% 15-trial RR: 98.5% (SVM-ensemble)
Datta *et al.* [75], 2018	MI: BCI II 3 ($N_s = 1$)	KNN with different parameters, different classifiers (KNN, SVM, NB), different features using one of the classifiers (KNN, SVM, NB)	Majority voting, mean	Acc: 82.86 % (MV)
Zhu *et al.* [95], 2021	SSVEP: ($N_s = 11$)	EEGNet (with different kernel numbers)	Averaging	Acc: $79.56 \pm 15.40\%$
Subasi *et al.* [76], 2021	MI: BCI III 4a ($N_s = 5$)	MSPCA + WPD + classification algorithm (KNN; C4.5 DT; REP Tree; SVM; RT; and RF)	RoF (average combination method) and RSM (majority voting) ensemble methods with different classification algorithms	Acc: 94.83% (RoF with KNN)

There is a number of other non-CNN based ensembles that can be found in the literature. Cavrini *et al.* [77] for instance presented a fuzzy integral ensemble method. Other examples of ensembles include the works that consider the following base classifiers: DTs [78]; stepwise LDA (SWLDA) [79]; SVMs [80, 81]; feed forward neural network (FFNN) [82]; LDA and Nearest Neighbor [83]; LDA, SVM and DT [84]; decision stump and KNN [85]. Comparative analysis of ensemble methods (bagging, boosting) are conducted by [86, 85, 87]. The latter two considered different feature extraction methods (autoregression (AR), power spectrum density (PSD), and CSP, discrete wavelet transform (DWT)). Different implementations of AdaBoost were demonstrated by [88, 89, 90, 91, 92, 93].

Talking about the CNN-based ensemble learning, Zhu *et al.* [95] trained EEGNet models and combined them by averaging strategy. Extensive reviews on the usage of ensembles in BCI can be found in [96].

3.4 Subject-Independent (SI) Classification in BCI

This section presents the overview of the approaches that have been proposed and tested so far for SI BCIs. Here, various paradigms, including those tested in the current study, namely MI and ERP, as well as others, such as visual imagery (VI) and mental imagery, are considered to give a full picture of the current state of the subject-independent classification problem in the BCI field. The summary from different studies is presented in Table 3.3, while a brief description of each work is provided below.

Before deep learning became so popular in EEG classification, Lotte *et al.* [97] compared the performance of LDA, quadratic discriminant analysis (QDA) and Gaussian mixture model (GMM) with different feature extraction methods and concluded that linear classifiers are the most suitable choice for constructing MI-based BCIs. Using LDA with multi-resolution (MR) decomposition based FBCSP feature extraction method the average accuracy of almost 71% was achieved. As a baseline, Lotte *et al.* [97] also considered knowledge from neurophysiology and evaluated SI performance on MI data without machine learning. For this, logarithmic band-power (BP) for mu and beta bands for electrodes in motor cortices region was computed, then the features were produced by averaging these BP over electrodes in left and right motor cortices separately, where the difference between the obtained features indicates the class. This a-priori knowledge-based approach demonstrated an average SI accuracy of 64.2%.

Fazli *et al.* [72] trained a set of subject-dependent (SD) CSP filters and LDA classifiers and combined the outputs to check the performance of the approach on the unseen data. The authors [72] sparsified the ensemble with quadratic regression with l_1 regularization. A similar idea has been tested on a different MI dataset in [73]. Here, in addition to the LDA classifier Fazli *et al.* [73] tested other classification methods such as KNN, SVM and LPM. An alternative to commonly used CSP features has been proposed by Reuderink *et al.* [98], who have employed the second-order baseline (SOB) covariance features. The SOB approach used a pre-trial baseline to adaptively normalize the covariance of the EEG channels. With a standard SVM classifier, the mean accuracy of 67.3% (across 51 test subjects) was achieved, which is on average higher than that with the CSP features (given the same classification algorithm) by 9%.

Lotte *et al.* [99] examined different ideas to suppress the calibration times required for the operation of BCI, one of which was a SI approach. In regard to SI classification, two designs have been considered. One is the pooled-data approach (the data for all but the test subject is pooled together and used for training) that was used to train CSP filters and an LDA classifier (from the earlier work [97]). The second design is the ensemble technique (similar to [72]) that incorporates the outputs from separately trained user-specific LDA classifiers with CSP filtering. The effect of using automatic covariance matrix shrinkage (ACMS) for both SI designs was analyzed.

Kwon *et al.* [100] established one of the largest MI-based EEG databases that involves 54 subjects, referred to as KU dataset. Based on this dataset, the authors generated the sets of features that are formed by spectral-spatial filtering at predefined frequency bands. Each set of features is fed through three convolutional layers, then a spatial fusion technique is used to integrate spectral-spatial features from different frequency ranges. Kwon *et al.* [100] analyzed the performance of the proposed method with different kernel sizes and number of feature maps. On average (across all subjects) the reported SI accuracy was $74.15 \pm 15.83\%$.

Different feature extraction methods, such as Katz Fractal Dimension (Katz FD), Sub band Energy, Log Variance and Root Mean Square (RMS), along with the LDA classification algorithm have been tested by Joadder *et al.* [101]. Best performing Katz FD method (on average) has been further analyzed in terms of different time windows, frequency bands, and number of channels. Zhang *et al.* [102] proposed a Convolutional Recurrent Attention Model (CRAM), which on a 4-class classification MI problem outperformed the state-of-the-art methods with the mean accuracy and AUC of $59.10 \pm 10.85\%$ and $81.86 \pm 8.05\%$, respectively. While describing the adaptation procedure (which is out of the scope of the current study) Hosseini *et al.* [103] reported the SI accuracy of 68.4% on the first 20 trials of each test subject using SVM with features from logarithms of variances of CSP channels.

Ghane *et al.* [104] conducted a comparative study on the performance of the popular classification algorithms, such as LDA, SVM, classification and regression tree (CART), and KNN, with power spectral density (PSD) based features in the context of MI classification task. The effect of different number of training samples has been investigated. It was concluded that with the relatively small sample size the best performing is LDA, while with the large number of training samples CART performs the best.

Lee *et al.* [105] proposed subepoch-wise feature encoder (SEFE) that was integrated as a part of popular CNN architectures, namely DeepConvNet [35], ShallowConvNet [28], and EEGNet [38]. The idea was to incorporate an additional convolutional block, which consists of two convolution layers with 64 and 32 1×1 filters separated by the rectified linear unit (ReLU) activation function layer, before the dense layer of the conventional architectures. On average, on a VI classification task this allowed to improve the performance by 5% in comparison to the regular CNN models.

Abibullaev *et al.* [108] conducted a brute-force (BF) search model selection among 9 predefined CNN architectures to define a model for subject-independent evaluation. The model selection procedure was based on leave-one-subject-out cross-validation (LOSO-CV), which means that for each non-test subject out of nine architectures, one best was selected and used in the test phase. Ayoobi and Sadeghian [109] used CSP [110] to extract features from EEG signals and fed them to a supervised autoencoder, which consists of an autoencoder network and a fully connected feed-forward binary classifier. Lu *et al.* [106] used FLD to first train the set of subject-specific classifiers. These multiple weak classifiers are then used to form a boosted classifier through the weighting based on their confidence, which is measured according to the classifier consistency. According to the authors [106], with such a boosting classification technique heavy noise could be suppressed. This allowed to achieve a performance even better than that of a subject-specific approach.

One of the most recent studies conducted by Jeon *et al.* [112] has considered feature decomposition and feature enrichment techniques along with existing deep neural network architectures to learn the subject-invariant and class-relevant features, which showed promising results on two large MI datasets with DeepConvNet [35] and EEGNet [38] architectures (approximate average accuracy of 73% with the standard deviation of about 14%).

Table 3.3: Summary of research works on subject-independent classification methods for EEG-based BCI.

Study	**Type of EEG Signals (Dataset)**	**Methods**	**Performance measures**
Lu *et al.* [106], 2008	ERP ($N_s = 10$)	FLD + Boosting	Acc: $> 95\%$
Lotte *et al.* [97], 2009	MI: BCI IV 2A ($N_s = 9$)	LDA, QDA, GMM with different feature extraction methods	Acc: 70.99% (MR FBCSP with LDA)
Fazli *et al.* [72], 2009	MI ($N_s = 45$)	Ensemble of SD CSP+LDA classifiers with l_1 regularization	Loss: 26.70% (l_1 regularized regression)
Fazli *et al.* [73], 2009	MI ($N_s = 83$)	Ensemble. Classification algorithms: LDA, SVM, KNN, LPM	Loss: 28.7%
Reuderink *et al.* [98], 2011	MI ($N_s = 109$)	SOB features + SVM	Acc: $67.3 \pm 13.4\%$ (average across 51 test subjects)
Lotte [99], 2015	MI: BCI IV 2A ($N_s = 9$); Workload ($N_s = 21$); Mental Imagery ($N_s = 20$)	Pooled-data and ensemble approaches [72] with CSP+LDA with ACMS	Acc: $\approx 72.5\%$ (pooled-data); $\approx 69.5\%$ (ensemble)
Joadder *et al.* [101], 2019	MI: BCI III 4A ($N_s = 5$)	Katz FD, Sub band Energy, Log Variance, RMS + LDA	Acc: 84.35% (Katz FD)
Kwon *et al.* [100], 2019	MI ($N_s = 54$)	CNNs with concatenation fusion technique	Acc: $74.15 \pm 15.83\%$
Zhang *et al.* [102], 2019	MI: BCI IV 2A ($N_s = 9$)	CRAM	Acc: 59.10 ± 10.85 AUC: 81.86 ± 8.05 NB: (MCCP)
Hosseini *et al.* [103], 2019	MI: BCI IV 2A ($N_s = 9$)	Log Variance of CSP features +SVM	Acc: 68.4% (tested on first 20 trials)
Ghane *et al.* [104], 2021	MI ($N_s = 20$)	LDA, SVM, KNN, and CART (PSD features)	Acc: $78.00 \pm 2.00\%$ (CART); $73.00 \pm 1.00\%$ (LDA)

Table 3.3: Continued.

Study	**Type of EEG Signals (Dataset)**	**Methods**	**Performance measures**
Zhang *et al.* [107], 2021	MI: KU [27] ($N_s = 54$)	DeepConvNet	84.19 ± 9.98
Lee *et al.* [105], 2021	VI ($N_s = 10$)	DeepConvNet [35], ShallowConvNet [28], and EEGNet [38] architectures with SEFE	Acc: $72.00 \pm 5.00\%$ (DeepConvNet [35] with SEFE)
Abibullaev *et al.* [108], 2022	ERP: BNCI14 ($N_s = 10$), BNCI15 ($N_s = 10$), ALS ($N_s = 8$), EPFL ($N_s = 8$)	CNNs with optimal hyperparameters selected through BF-based LOSO-CV model selection procedure	Acc: 75.80% (BNCI14); 61.00% (BNCI15); 69.47% (ALS); 59.43% (EPFL)
Ayoobi *et al.* [109], 2022	MI: BCI IV 2A ($N_s = 9$)	CSP features [110] + SAE	Kappa value: 0.50
Salami *et al.* [111], 2022	MI: BCI IV 2A ($N_s = 9$) MI: BCI IV 2B ($N_s = 9$) MI: KU [27] ($N_s = 54$)	Inception Temporal Convolutional Network (EEG-ITNet)	Acc: $69.44 \pm 8.98\%$ (BCI IV 2A) $78.74 \pm 9.40\%$ (BCI IV 2B) 73.52% (KU dataset, 20 channels are used)
Jeon *et al.* [112], 2023	MI: GIST [113] ($N_s = 52$), KU [27] ($N_s = 54$)	Feature decomposition and feature representation enrichment using deep learning network architectures (DeepConvNet [35] and EEGNet [38])	Acc: $73.32 \pm 13.55\%$ (with DeepConvNet [35] on KU dataset); $73.73 \pm 13.75\%$ (with EEGNet [38] on GIST [113])
Nouri *et al.* [114], 2023	MI: KU [27] ($N_s = 54$)	Convolutional Common Spatial Pattern Network (CCSPNet)	Acc: $74.28 \pm 16.12\%$
Luo *et al.* [115], 2023	MI: BCI IV 2A ($N_s = 9$) MI: BCI IV 2B ($N_s = 9$) MI: KU [27] ($N_s = 54$, but 21 subject is used)	Shallow Mirror Transformer	Acc: 52.31% (BCI IV 2A) 67.03% (BCI IV 2B) 77.18% (KU dataset, leave 3 subjects out)

Chapter 4

Materials and Methods

4.1 Ensemble Learning

An ensemble classifier is a machine learning model that incorporates multiple individual models, known as base models or base classifiers [116, 96]. It takes advantage of the collective knowledge and different perspectives of the base classifiers to enhance overall prediction accuracy and robustness [117].

The ensemble learning techniques could be broadly categorized into parallel and sequential ensembles. These categories differ in how they combine the predictions of multiple base models. While most of this Thesis focuses on the parallel ensembles, there will also be a discussion on the possible implementation of the sequential ensemble for the EEG classification.

4.1.1 Parallel ensembles

In parallel ensembles, multiple base models are trained independently in parallel. Each base model is typically trained on a given subset of the training data and the final prediction is made by aggregating the individual predictions of these base models.

Consider a binary classification scenario where given an input p-dimensional feature vector $\mathbf{x}_i$ a classifier $\psi(\mathbf{x}_i)$ assigns a binary label $y_i = 0, 1$, where $i = 1, ..., n$ with n indicating the total number of observations. In other words, the classifier is a mapping $\psi : \mathbb{R}^p \rightarrow \{0, 1\}$ where ψ is defined as $\psi(\mathbf{x}_i) = 0$ when $\mathbf{x}_i$ is in the set R_0, and $\psi(\mathbf{x}_i) = 1$ when $\mathbf{x}_i$ is in the set R_1, with R_0 and R_1 being the measurable sets that divide the sample space.

An ensemble classifier $\psi^{\mathrm{E}}(\mathbf{x}_i)$ combines a group of M base models $\psi_m(\mathbf{x}_i)$ with $m = 1, \ldots, M$ to produce a single prediction or probability distribution for classifying observation $\mathbf{x}_i$. This is represented as

$$\psi^{\mathrm{E}}(\mathbf{x}_i) = \mathcal{C}\left(\bigcup_{m=1}^{M} \{\psi_m(\mathbf{x}_i)\}\right), \tag{4.1}$$

where $\mathcal{C}(.)$ is a specific mechanism known as a combination rule or a combiner that operates in a function space.

Two commonly used approaches in ensemble learning for combining the predictions of multiple base classifiers are hard voting and soft voting. A hard voting based ensemble classifier $\psi^{\mathrm{E}}_{HV}(\mathbf{x}_i)$ relies on the class labels predicted by the base classifiers.

The reliability of each base classifier $\psi_m(\mathbf{x}_i)$ can be encoded using the weight w_i, such that the decision of ensemble classifier $\psi^{\mathrm{E}}_{HV}(\mathbf{x}_i)$ is formulated as

$$\psi^{\mathrm{E}}_{HV}(\mathbf{x}_i) = \underset{y_i \in \{0,1\}}{\arg\max} \sum_{m=1}^{M} w_m \mathrm{I}_{\{\psi_m(\mathbf{x}_i)=y_i\}} \,, \tag{4.2}$$

where $\mathrm{I}_{\{S\}}$ is 1 if statement S is true, zero otherwise [118]. In case all the base classifiers $\psi_m(\mathbf{x}_i)$ are given the same weight of $w_m = 1$, the weighting combination rule in Eq. 4.2 is reduced to the basic majority vote (MV) combining scheme, such that ensemble decision is determined based on the outputs of the majority of base classifiers [118].

In contrast to hard voting, in soft voting the ensemble classifier $\psi^{\mathrm{E}}_{SV}(\mathbf{x}_i)$ considers the probability distributions assigned by the base classifiers for each class label. These probability values are combined across the base classifiers to determine the final prediction. With $p^m_{y_i}$ denoting the estimate of the posterior probability for class $y_i = 0, \ldots, c-1$ ($c = 2$ in the context of binary classification) by classifier $\psi_m(\mathbf{x}_i)$, the ensemble's decision based on soft voting [119] is

$$\psi^{\mathrm{E}}_{SV}(\mathbf{x}_i) = \underset{y_i \in \{0,1\}}{\arg\max} \frac{1}{M} \sum_{m=1}^{M} p^m_{y_i} \,. \tag{4.3}$$

In addition to the regular probability averaging combining scheme (soft voting presented in Eq.4.3), the probabilistic classifier ensemble weighting (PCEW) voting mechanism [118] is considered. In PCEW, the probabilities p^m_{yi} are weighted by the estimated accuracies of the base classifiers ($\hat{\mathrm{acc}}_m$) raised to the power of α, which represents the degree of trust to the estimated accuracies. The ensemble classifier $\psi^{\mathrm{E}}_{PCEW}(\mathbf{x}_i)$ assigns $\mathbf{x}_i$ to the class that has the greatest sum of weighted probabilities

$$\psi^{\mathrm{E}}_{PCEW}(\mathbf{x}_i) = \underset{y_i \in \{0,1\}}{\arg\max} \sum_{m=1}^{M} \hat{\mathrm{acc}}^{\alpha}_m p^m_{y_i} \,. \tag{4.4}$$

4.1.2 Sequential ensembles

Adaptive boosting (AdaBoost) is a specific type of ensemble method, where the base models are generated sequentially. It is popular for its compatibility, high speed, and low complexity [120]. Initially introduced in 1997 by Freund and Schapire [121], AdaBoost has demonstrated notable practical success across a range of applications, including face detection [122], human detection [123], hand tracking [124], brain-computer interface [88], and image classification [125]. Over the past two decades, various versions of AdaBoost have emerged, each exploring different base estimators and enhancements to the boosting algorithms. Each variant of AdaBoost seeks to build upon its predecessor by addressing specific limitations.

The sequential process of training ensures the transfer of acquired knowledge from one model to the next. This knowledge transfer is achieved through the incorporation of sample weights. In the context of AdaBoost, these sample weights play a pivotal role in highlighting the importance of specific observations in subsequent learning iterations. Higher weights indicate that the subsequent classifier should allocate more attention to the designated observations.

To construct AdaBoost, M base classifiers are trained sequentially. Unlike the process described in subsection 4.1.1, here the output of a given base classifier $\psi_m(\mathbf{x}_i)$ has an effect on the subsequent learner $\psi_{m+1}(\mathbf{x}_i)$. Consider a p-dimensional feature vectors $\mathbf{x}_i$, where $i = 1, \ldots, n$ with n being the total number of observations. Each of these observations is assigned a weight d_i^m, where $m = 1, \ldots, M$ indicates the m-th base estimator for which the weight is applicable [126]. A class label associated with each observation $\mathbf{x}_i$ is $y_i = 0, 1, 2, ..., c-1$ with c being the total number of classes. The label is encoded in a c-dimensional vector form as:

$$\mathbf{y}_i = \begin{cases} [1, -\frac{1}{c-1}, \ldots, -\frac{1}{c-1}]^T, & \text{for class } 0 \\ [-\frac{1}{c-1}, 1, \ldots, -\frac{1}{c-1}]^T, & \text{for class } 1 \\ \vdots & \\ [-\frac{1}{c-1}, \ldots, -\frac{1}{c-1}, 1]^T, & \text{for class } c-1 \end{cases} \tag{4.5}$$

At the beginning, all of the data observations are initialized with the same weights of $d_i^1 = 1/n$ [126]. The weights for the next steps d_i^{m+1} are updated based on

$$d_i^{m+1} = d_i^m \exp\left(-\frac{c-1}{c}\mathbf{y}_i^T \cdot \log(\mathbf{p}^m(\mathbf{x}_i))\right), \tag{4.6}$$

where operation $\log(\mathbf{p}^m(\mathbf{x}_i))$ produces the vector obtained by taking the logarithm of each element in $\mathbf{p}^m(\mathbf{x}_i)$, which is the output vector from the m-th base model $\mathbf{p}^m(\mathbf{x}_i) = [p_{y_0}^m(\mathbf{x}_i), p_{y_1}^m(\mathbf{x}_i), \ldots, p_{y_{c-1}}^m(\mathbf{x}_i)]$ showing the probabilities $p_{y_i}^m(\mathbf{x}_i)$ that the input is associated with particular class y_i [126]. When the training sample is effectively trained, meaning that the model's output for this sample closely aligns with the label vector, the weight d_i^{m+1} is reduced due to the small value of the exponential function in Eq.4.6. Subsequently, the process of adjusting sample weights is followed by normalization, which is accomplished by dividing the sample weights by the total sum of weights.

Each base classifier $\psi_m(\mathbf{x}_i)$ is trained on the data with the corresponding weights d_i^{m+1} and the predictions of $\psi_m(\mathbf{x}_i)$ are calculated as

$$\psi_m(\mathbf{x}_i) = (c-1)\left(\log(p_{y_i}^m(\mathbf{x}_i) - \frac{1}{c}\sum_{k=0}^{c-1}\log(p_k^m(\mathbf{x}_i))\right). \tag{4.7}$$

After all of the base classifiers are trained, the decision of the AdaBoost classifier $\psi_{AB}^{\mathrm{E}}(\mathbf{x}_i)$ is a combination of individual outputs from the base classifiers [126]:

$$\psi_{AB}^{\mathrm{E}}(\mathbf{x}_i) = \underset{y_i \in \{0,1\}}{\arg\max} \sum_{m=1}^{M} \psi_m(\mathbf{x}_i). \tag{4.8}$$

4.2 Convolutional Neural Networks

Like any other neural network, a CNN is composed of different layers: an input layer, an output layer, and several hidden layers in between. The hidden layers of a CNN are specifically designed to incorporate convolutional layers. Convolutional layers perform convolutions on a three-dimensional input tensor, which consists of elements represented by $\mathbf{X}_{i,j,k}$, where $1 \leq i \leq c_{\mathbf{X}}$, $1 \leq j \leq h_{\mathbf{X}}$, and $1 \leq k \leq w_{\mathbf{X}}$, with $c_{\mathbf{X}}$, $h_{\mathbf{X}}$, and $w_{\mathbf{X}}$ being channels, height and width of the input, respectively. For the type of data (recordings of EEG signals) used in this Thesis, in the first convolutional layer there is one input channel ($c_{\mathbf{x}} = 1$, similar to a grayscale image); the height of the input is represented by the number of electrodes (EEG channels) that were used to record the data (see N_E is Table 4.1 for each of the dataset); and the width of the input is $w_{\mathbf{x}} = f_s \times t$, where f_s is the sampling rate in Hz and t is the duration of the signal segment is seconds. Each convolutional layer applies convolutions using a multidimensional array of parameters called a kernel. The kernel is a small four-dimensional tensor that slides over the input data during the convolution operation. At each position, the kernel performs element-wise multiplication with the corresponding portion of the input, so that a single value in the output feature map is generated. The elements of the kernel are represented by $\mathbf{K}_{l,i,j,k}$, where $1 \leq l \leq c_{\mathbf{Y}}$ (number of output channels), $1 \leq i \leq c_{\mathbf{X}}$, $1 \leq j \leq h_{\mathbf{K}} < h_{\mathbf{X}}$, $1 \leq k \leq w_{\mathbf{K}} < w_{\mathbf{X}}$ with $w_{\mathbf{K}}$ and $h_{\mathbf{K}}$ being the height and width of the kernel.

The output from the convolution of the kernel $\mathbf{K}_{l,i,j,k}$ and input $\mathbf{X}_{i,j,k}$ is a three dimensional feature map represented by:

$$\mathbf{Y}_{l,m,n} = \sum_{i,j,k} \mathbf{X}_{i,j+m-1,k+n-1} \mathbf{K}_{l,i,j,k} \, , \tag{4.9}$$

where $1 \leq l \leq c_y$, $1 \leq m \leq h_x - h_k + 1$, and $1 \leq n \leq w_x - w_k + 1$ with the summation over all valid indices and assuming no padding, subsampling, bias term. This feature map (or activation map) represents the response of the kernel at different parts of the input image.

Another important type of layer in CNN is the pooling layer, which reduces the spatial size of the feature map by summarizing the statistics of nearby outputs. The most commonly used pooling function is max-pooling, which selects the maximum value from the neighborhood.

To introduce non-linearity into the network, an activation function is applied. The ReLU is commonly used in hidden layers, while the Softmax activation function is typically used in the output layer. The output of the last convolutional or pooling layers is usually flattened and fed into a fully connected (FC) layer. However, to reduce the dimensionality and complexity of the architecture, the FC layer could be replaced with global average pooling. This approach reduces the number of trainable parameters and speeds up training.

4.3 Datasets

The methods investigated in this Thesis were verified on the publicly available datasets, that were collected using different experiments with the prescribed protocol. The type of data includes EEG signals from ERP and MI-based experiments. Below is the general description of the datasets. Table 4.1 presents a summary of information regarding the type of the BCI paradigm (ERP or MI), the number of participants involved in the experiments, the channel types used, and the task performed by the subjects to collect particular datasets.

4.3.1 ERP-based datasets

The experiments conducted to collect the ERP-based datasets described below (BNCI, ALS, EPFL) followed the Farwell and Donchin style paradigm.

BNCI dataset [127] represents a collection of ERP data from participants under both overt and covert attention conditions using a P300 speller. In the context of this study, only the data related to covert visual attention were considered. The participants, specifically ten healthy females with a mean age of 26.8 ± 5.6, were presented with a 6×6 grid of alphanumeric characters on a liquid crystal display monitor. EEG signals were recorded using a 16-electrode setup. The reference electrode was placed on the right earlobe, and the ground electrode was placed on the right mastoid. The EEG signals were recorded at a sampling rate of 256 Hz using g.USBamp from g.tec (Austria), and they were filtered with high-pass and low-pass cutoff frequencies of 0.1 and 20 Hz, respectively.

ALS dataset [128] comprises P300 evoked potentials data obtained from eight participants diagnosed with ALS, including three women and five men with a mean age of 58 ± 12. EEG signals were recorded with eight electrodes. The recording was conducted at a sampling rate of 256 Hz using g.MOBILAB equipment from g.tec (Austria). The reference electrode was positioned at the user's right earlobe, while the ground electrode was placed on the left mastoid.

During the experiment, the participants were instructed to use a grid of 6×6 characters presented through the P300 matrix speller. Their task involved spelling seven predetermined words, each comprising five characters. For each character, each column and row of the grid were flashed ten times, resulting in a total of 20 flashes per character on the grid. This sequence was repeated for each character choice, constituting a trial.

EPFL dataset [129] was obtained from a total of eight participants, consisting of four individuals with disabilities and four healthy subjects. All participants were male, with an average age of 30 ± 2.3. During the experiment, six images representing a lamp, a television, a door, a window, a telephone, and a radio were presented in a random sequence. Each image was flashed for 100 milliseconds, with a stimulus interval of 400 milliseconds between consecutive flashes. The same target image did not reappear for a duration of 700 milliseconds. The experiment was conducted in two sessions on one day and another two sessions on a different day for each participant. Each session consisted of one run per stimulus image, lasting approximately 1 minute. EEG data were collected using a set of 32 electrodes placed according to the international 10-20 system. The sampling frequency for recording the EEG signals was set at 2048 Hz.

These datasets are accessible through the MOABB tool [130]. Additional details regarding the experimental protocols employed to collect these datasets can be found at [127, 128] and [129].

4.3.2 MI-based datasets

BCI IV 2a dataset represents the data from the cue-based BCI experiment with nine participants engaging in four different types of imagery tasks involving left-hand, right-hand, both feet and tongue movements. The data were collected using 22 Ag/AgCl electrodes placed according to the 10-20 system for electrode placement. The left mastoid served as the reference electrode, while the right mastoid was used as the ground electrode. The EEG signals were sampled at a frequency of 250 Hz, and bandpass filtering was applied within the range of 0.5 Hz to 100 Hz. Further information can be found in [29].

BCI IV 2B dataset consists of EEG data from nine healthy individuals who were right-handed. During the experiment, the participants performed motor imagery tasks involving two classes: left-hand and right-hand movements. The EEG data were recorded at a sampling frequency of 250 Hz using three channels. The Fz electrode was employed as the reference lead for the EEG, and bandpass filtering was conducted within the frequency range of 0.5 Hz to 100 Hz. For more detailed information, refer to [30].

In addition to above mentioned BCI competition MI-based datasets, one of the largest publicly available EEG-based MI datasets collected by Lee et al. [27] was used. This dataset is referred to as KU dataset.

KU dataset involved 54 participants (25 females and 29 males, aged from 24 to 35) who were free from any relevant diseases. Among them, 16 individuals had prior experience with BCI experiments, while the remaining participants were new to BCI usage. The dataset comprises EEG recordings from two sessions of BCI experiments conducted with the same subjects and following identical protocols.

Each experiment session consisted of both a training phase (offline) and a testing phase (online), with 100 trials in each phase. Half of the trials were specific to right-

Table 4.1: Details about the datasets that are used in this study.

Dataset	Subjects	Channels	Task
ERP-based datasets			
BNCI [127]	- 10 healthy female - mean age: 26.8 ± 5.6	- 10–10 electrode placement system - $N_E = 16$ (Ag/AgCl electrodes): Fz, FCz, Cz, CPz, Pz, Oz, F3, F4, C3, C4, CP3, CP4, P3, P4, PO7, PO8	Alphanumeric characters on a 6×6 grid (Farwell & Donchin style)
ALS [128]	- 8 participants (3 women and 5 men) with amyotrophic lateral sclerosis (ALS) - mean age: 58 ± 12	- 10–10 electrode placement system - 8 electrodes: Fz, Cz, Pz, Oz, P3, P4, PO7, PO8	Spell 7 predefined words consisting of 5 characters using the grid of 6 x 6 characters (Farwell & Donchin paradigm)
EPFL [129]	- 8 male (4 disabled and 4 healthy subjects) - mean age: 30 ± 2.3	- 10–20 placement system - 32 electrodes: Fp1, AF3, F7, F3, FC1, FC5, T7, C3, CP1, CP5, P7, P3, Pz, PO3, O1, Oz, O2, PO4, P4, P8, CP6, CP2, C4, T8, FC6, FC2, F4, F8, AF4, Fp2, Fz, Cz	Six images: lamp, television, door, window, telephone, and radio
MI-based datasets			
BCI IV 2a	- 9 participants	- 10-20 placement system 22 Ag/AgCl electrodes	Four imagery tasks: left-hand, right-hand, both feet, and tongue movements
BCI IV 2b	9 healthy participants	-3 electrodes: C3, Cz, and C4	Left-hand and right-hand movements
KU [68]	- 54 healthy participants (25 females and 29 males) - ages: $24 - 35$	- 10-20 placement system - 62 Ag/AgCl electrodes	Imagery task of grasping left or right hand

hand imagery tasks, while the remaining half focused on left-hand imagery tasks. The EEG signals were recorded using 62 Ag/AgCl electrodes at a sampling rate of 1000 Hz. A comprehensive description of the BCI experiment can be found in [27] and the dataset itself can be downloaded from GigaDB webpage.

4.4 Preprocessing

One of the key advantages of deep learning is its ability to mitigate the need for preprocessing or handcrafted features. Instead of relying on manually designed features, deep learning models can learn relevant representations directly from the raw input data, reducing the burden of feature engineering. In order to make the most of this benefit and allow the CNNs to learn the discriminative features automatically without relying on prior knowledge or extensive preprocessing, minimal preprocessing procedures were applied to the above-mentioned datasets.

ERP datasets (BNCI, ALS, EPFL)

The pipeline for ERP preprocessing prepared by Farquhar and Hill [131] was followed. The same has been followed in [132, 108]. The continuous EEG data were divided into target and non-target trials, each lasting 600 ms, starting from the onset of the stimulus event markers. To remove noise caused by slow drifts, which can be attributed to factors like sweating or a weak sensor-to-head contact [133], any arbitrary offsets present in the data were eliminated by subtracting the overall mean from each channel. The EEG data trials underwent an artifact editing process using statistical thresholding. This procedure aimed to eliminate trials that contained severe movement artifacts. Trials whose mean absolute values exceeded three standard deviations from the median trial were excluded. All channels were analyzed across the trials to identify electrodes that exhibited excessive noise due to improper connection with the subject's scalp. Channels displaying abnormally high power were identified and removed from further analysis. The removed channels were substituted with a single reference channel known as the common averaged reference (CAR) channel, which is created by averaging the electrical activity measured across all channels.

To perform spectral filtering on the EEG data, a bandpass filter was applied within the range of 0.5-12 Hz using a Fourier filter [131]. Initially, the signal underwent Fourier transformation, and then a weighting was applied to suppress and eliminate undesired frequencies outside the desired frequency range. The weighted signal was inverse Fourier transformed to obtain the filtered signal.

BCI IV 2A and 2B datasets

To reduce electro-oculographic artifacts caused by eye movement, which are predominant in EEG signals within the frequency band of 0.1 to 4 Hz, the high-pass filter (namely fourth-order Butterworth Infinite Impulse Response (IIR) filter) was applied to EEG waveforms with a cut-off frequency set at 4 Hz. Apart from this specific high-pass filtering, as recommended by Schirrmeister *et al.* [35], in order to preserve the original raw EEG data no low-pass filtering was used.

The continuous EEG recording was divided into separate segments corresponding to left-hand and right-hand motor imagination trials. Each segment had a duration of four seconds, starting from the onset of motor imagery. Bad trials and noisy channels were excluded following the same procedure as for ERP datasets described above.

KU dataset

The EEG recordings were down-sampled to 250 Hz and an 8th order Chebyshev anti-aliasing filter was applied [100, 114, 112, 107].

4.5 Dataset Partitioning and Model Evaluation

Partitioning dataset into training, validation, and test sets is a fundamental aspect in machine learning [134]. The training set is used to train the model. It is utilized to optimize the model's parameters (weights and biases) through iterative learning algorithms. During training, the model learns patterns and features from the data, aiming to minimize the discrepancy between its predictions and the ground truth labels. The validation set is crucial for assessing the performance of the model during training and for tuning hyperparameters. The validation set helps in selecting the best hyperparameters by evaluating the model's performance on data that it hasn't been trained on. This helps to prevent overfitting, where the model performs well on the training data but fails to generalize to unseen data. The test set is used to evaluate the final performance of the trained model. It serves as an unbiased estimate of the model's performance on unseen data. The test set is separate from the training and validation sets and is only used once the model has been fully trained and tuned. By dividing the dataset into training, validation, and test sets, it is possible to train, fine-tune, and evaluate the models in a systematic and robust manner. This process helps to ensure that the model generalizes well to new, unseen data and performs optimally in real-world applications. Given the interest in the subject-independent classification problem, within this Thesis the way the data is allocated for the test set for model evaluation is common across various methods following the idea of holding out one subject. This process is elaborated in the subsequent subsection. After setting aside the test set, the remaining data is divided into training and validation sets depending on specific strategy. This is discussed in Chapter 6 while presenting the proposed methods.

4.6 Subject-Independent Classification

In subject-independent evaluation of classifiers in EEG-based BCI, a commonly used approach is Leave-One-Out Cross-Validation (LOO-CV) [105, 112, 99, 72, 108, 109, 73, 45], which is also aptly named as Leave-One-Subject-Out CV (LOSO-CV) as it accurately reflects the underlying process it entails. In LOO-CV (LOSO-CV), for the dataset consisting of N subjects, the trials associated with subject S_i (where i ranges from 1 to N) are systematically excluded (left out). Then, a classifier is trained using data from all other subjects except S_i, and the excluded trials of S_i are used to evaluate the classifier's performance. This ensures that the test subject S_i is only used during the final stage to evaluate the designed classifier, and is not involved in the training or model selection phases. Throughout this Thesis, the LOSO-CV method is employed for evaluating classifiers in an SI manner. Thus, considering data from N subjects in each dataset, we are left with the observations from $N - 1$ subjects for training and model selection. The distribution of these observations across the training and validation sets varies across the proposed methods and therefore is discussed separately in each case.

4.7 Computational resources

The simulations and analyses presented in this Thesis were performed using different computational resources. The choice of specific resources was based on their availability during period when the work was conducted. Below is the summary of the resources used so far:

- Windows workstation with Intel(R) Xeon(R) CPU E5-1650, (3.50 GHz) processor, 16 GB of RAM, and NVIDIA TITAN RTX (RAM = 24GB)
- Remote access to NVIDIA DGX-1 server [1]
- Remote access to NVIDIA DGX-2 server [1]

All of the codes and analyses were implemented in Python. Mostly Pytorch deep learning environment was utilized.

[1]provided by Institute of Smart Systems and Artificial Intelligence at Nazarbayev University

Chapter 5

Base Classifiers: Convolutional Neural Networks

Before delving into the proposed ensemble methods themselves, it is crucial to explore the structure of utilized types of base classifiers, as these form the core of any ensemble. Many research studies have already shown that CNNs are efficient in terms of training time and effective at capturing latent features from raw EEG data (see Table 3.1, Chapter 3). The majority of these studies rely on well-established architectures (*e.g.* AlexNet or ResNet), leverage domain knowledge to create the final architecture, or employ an unclear strategy for model selection [135].

In this study, two cases for designing the base models for the ensemble are considered. First, using the systematic model selection the aim is to demonstrate that even with the limited domain knowledge the accurate classification models could be designed. Second, with the state-of-the-art architectures being used for designing the base models, the proposed methods can be viewed as simple, yet effective ways to improve the classification performance of *existing* CNN architectures.

5.1 A Brute-Force CNN Model Selection

As one of the methods to define the base models for the ensemble, the principled model selection, namely brute-force search, is considered. It is a straightforward and exhaustive search algorithm that systematically explores and evaluates all possible combinations within a given problem space. This, however, might entail significant computational overhead (depending on the defined hyperparameter space). Nevertheless, given the availability of computational resources, this method can serve as a tool to devise an efficient solution even in cases of limited domain knowledge.

One of the first works on a brute-force CNN model selection for accurate EEG classification has been presented in [135]. The evaluation of the approach has been conducted on an independent MI dataset that was unseen during the architecture selection phase. In [132], a systematic model selection was used for the classification of P300 waveforms. A different approach for CNN parameter (kernel sizes and number of convolutional nodes) selection, namely coordinate descent as a suboptimal method to perform cross-validation for the network parameters, was used by Sakhavi *et al.* [44].

In this work, the exhaustive (grid) search is used to tune the hyperparameters of each base CNN classifier within the ensemble for SI classification of MI and ERP-based EEG. For the purpose of model selection, the initial step involves defining a hyperparameter space denoted as Θ. Within this space, an exhaustive search to determine the optimal combination of hyperparameters is performed. Let $\theta_h \in \Theta$, where h takes values from 1 to H (being the cardinality of Θ), represent various combinations of hyperparameters. For each θ_h, a classifier with a fixed set of hyperparameters using the training dataset S_{train} is constructed. Subsequently, the accuracy of this classifier on the corresponding validation set, denoted S_{val}, is assessed. This process results in the creation of a classifier and its associated estimated accuracy, denoted $\psi_{\theta_h, S_{\text{train}}, S_{\text{val}}}$ and $\hat{\text{acc}}_{\theta_h, S_{\text{train}}, S_{\text{val}}}$, respectively. Upon reaching the conclusion of this iterative process ($h = H$), the estimated accuracies $\hat{\text{acc}}_{\theta_h, S_{\text{train}}, S_{\text{val}}}$ are compared. The classifier exhibiting the highest estimated accuracy is selected. This chosen classifier, along with its corresponding estimated accuracy, is denoted as $\psi^*_{S_{\text{train}}, S\text{val}}$ and $\hat{\text{acc}}^*_{S_{\text{train}}, S_{\text{val}}}$.

To search for the optimal hyperparameters, it is necessary to first define a limited parameter space. For each base CNN model the following hyperparameters are used for tuning: number of convolutional layers in the classifier, number of filters in different layers, convolutional kernel sizes, and dropout rate. The possible search space of hyperparameters is described below.

The CNNs with a minimum of 2 and maximum of 6 convolutional layers are used: $L = \{2, 3, ..., 6\}$, where L is the total number of convolutional layers in the classifier. Two cases to define the number of filters in j-th convolutional layer (f_j) are considered: $2^{(L+3-j)}$ and $2^{(L+4-j)}$, where $j = L, ..., 1$ for expanding pattern and $j = 1, ..., L$, for shrinking pattern. For example, when considering two-layered CNNs, four types of CNNs are constructed: $\{f_1 = 2^3, f_2 = 2^4\}$, $\{f_1 = 2^4, f_2 = 2^5\}$, $\{f_1 = 2^4, f_2 = 2^3\}$, $\{f_1 = 2^5, f_2 = 2^4\}$. Two types of convolutional filters are employed: one with a fixed size across all layers, including 3×3, 3×8, 3×24, 3×40, 5×5, and 7×7; and the other with varying kernel sizes, such as 7×7 in the first layer, 5×5 in the second layer, and 3×3 in the subsequent layers. As could be noticed above mostly structural hyperparameters, which determine the overall structure of the network, were varied. Most of the algorithmic hyperparameters, which govern how the network is trained and how the learning algorithm operates, were set to fixed values. The only exception was the dropout rate, for which two options were considered: 10% and 50%. All CNNs were trained with Adam optimizer for 150 epochs with early stopping with a patience of 30. The learning rate was set to 10^{-4} with a weight decay of 10^{-5}. A batch size of 32 was used. The loss function was cross-entropy. Consideration of structural and algorithmic hyperparameters mentioned above results in 280 distinct CNN architectures, which are used in the model selection process.

5.2 State-of-the-art Architectures

Searching through a wide range of hyperparameters with the aim to construct a reliable (base) classifier is a viable option. This, however, entails significant computational expenses. An alternative approach involves designing a classifier based on an adaptation of an existing deep neural Network (DNN) by incorporating domain knowledge. For instance, domain expertise was originally applied in the development of EEGNet [38], a CNN architecture that has demonstrated exceptional performance in diverse tasks related to EEG classification. Nevertheless, integrating domain knowledge into the architecture of a DNN remains a challenging process. Therefore, instead of conducting an extensive search or attempting to incorporate domain-specific expertise, another option is to adopt a preexisting architecture that has already proven its efficacy in addressing the problems of interest. It is hypothesized that by employing certain ensemble strategies, the performance of the given predefined well-performing model could be further enhanced.

In the field of BCI, EEGNet stands out as a prominent CNN architecture. Another noteworthy pair of models, ShallowConvNet and DeepConvNet, were introduced by Schirrmeister *et al.* [35] before EEGNet's inception. In this Thesis, these three architectures (ShallowConvNet, DeepConvNet and EEGNet) are used as the architectures of the base models within the ensembles. The implementation of the CNN architectures was adopted from the work of Schirrmeister *et al.* [35]. The models were downloaded using Braindecode. It is Python-based open-source toolkit designed for using deep learning models to decode raw electrophysiological brain data. The codes for the models are available on the GitHub[1]. A brief description of the CNN architectures is presented below.

5.2.1 DeepConvNet

DeepConvNet is a deep CNN architecture that is comprised of four convolutional layers and one dense softmax classification layer. The first layer is a special convolutional layer performing split temporal and spatial convolutions. The number of filters follows an expanding pattern, starting with 25 filters in the first layer (for both temporal and spatial convolutions), then progressing to 50, 100, and 200 filters in the subsequent layers. Following each convolutional step, except for the temporal convolution in the first layer, batch normalization, an exponential linear unit (ELU) activation function, max-pooling, and dropout with a 0.5 dropout rate are applied. Adam optimization algorithm with decoupled weight decay for training [39] is employed.

[1]https://github.com/robintibor/braindecode/tree/master/braindecode/models

5.2.2 ShallowConvNet

ShallowNet is a more shallow architecture with respect to DeepConvNet. It has also demonstrated its ability to decode raw EEG signals without the need for hand-crafted features. ShallowNet comprises two key blocks. The first block performs temporal and spatial convolution, followed by batch normalization and a squaring non-linearity. The second block incorporates average pooling with a logarithmic activation function, culminating in classification through the Softmax function. Taking inspiration from FBCSP, ShallowConvNet was designed to decode band power features.

5.2.3 EEGNet

The aim of designing the EEGNet [38] was to investigate whether it is possible to create a single CNN architecture that can accurately classify EEG signals from various BCI paradigms while also keeping the model as compact as possible. The compactness was defined as minimizing the number of parameters in the model. EEGNet consists of three blocks. Being also inspired by the FBCSP, the first block of EEGNet employs a two-step convolution, resulting in feature maps across various band-pass frequencies. Depthwise convolution, borrowed from computer vision applications, is applied to create frequency-specific spatial filters. Each convolutional step is followed by batch normalization, with an ELU nonlinearity as the activation function. The average pooling layer reduces the signal's sampling rate, and a dropout rate of 0.5 serves as a regularizer to prevent overfitting.

In the second block, separable convolution (combining depthwise and pointwise convolutions) is employed. This not only reduces the number of trainable parameters but also allows for the decoupling of relationships within and across feature maps. The dimension reduction is achieved through an average pooling layer. The final block performs classification by directly feeding features into a Softmax function. By removing the dense layer, the authors have effectively reduced the number of free parameters in the architecture.

Chapter 6

MS-En-CNN: Multi-Subject Ensemble CNN for EEG-based BCI

Despite the advantages associated with CNNs and ensemble learning, there has been a lack of research on the effectiveness of ensemble CNN models in the broader field of machine learning and, more specifically, in applications related to BCI. Table 6.1 shows examples of studies that have employed ensemble CNNs in different applications.

The creation of a neural network ensemble can involve different approaches, such as altering the network's structure, modifying initial weights, or introducing variations in the training dataset [136]. In Table 6.1, these ensembles are referred to as multi-structure ensembles, multiple-weight initializer-based ensembles, and multiple-input representation-based ensembles. Furthermore, the multi-scheme ensemble combines base classifiers using any of these aforementioned methods to enhance ensemble diversity.

In this study, the primary focus is on assessing the viability of a *multi-subject* ensemble CNN (MS-En-CNN) for SI classification of EEG data. MS-En-CNN is an ensemble of CNN base classifiers that are trained from a subset of training subjects. In this chapter three distinct methods for training base classifiers, combining their outputs to create the MS-En-CNN classifier are introduced. The effectiveness of these ensemble models is compared against the individual CNN base classifiers that constitute the ensemble. In addition, a comparison with the conventional approach, where a single CNN is trained using combined data from various training subjects, is conducted. The significance of this study lies in the potential of MS-En-CNN to enhance the robustness and accuracy of SI neural decoding systems, particularly those that leverage efficient deep CNN architectures.

The remainder of this chapter is dedicated to discussing the strategies proposed for constructing MS-En-CNN. Before introducing these strategies, a description of the conventional approach for training the classification model is provided, serving as the foundation for the comparative analysis.

Table 6.1: Examples of CNN ensemble-based studies for different applications.

Study	Applications	Ensemble methods
[137]	Classification of the transportation mode of trip	**Multi-structure ensemble**: combination of different CNN architectures with different hyper-parameter values **Ensembling methods**: average voting, majority voting and optimal weights method
[138]	Gender prediction from face images	**Multiple weight initializer-based ensemble**: combination of three instances of the same CNN model trained with random initialization of weights **Ensembling method**: averaging
[139]	Image classification problem	**Multi-structure ensemble**: combination of 4 CNNs with different filter sizes in two convolutional layers **Multiple feature subset based ensemble**: CNN models trained on different feature sets **Multiple input representation based ensemble**: combination of different input forms with various spectral characteristics. **Multi-scheme ensemble**: combinations of three different schemes
[140]	Environmental Event Sound Recognition	**Multi-structure ensemble**: combination of two architectures **Ensembling method**: Dempster-Shafer evidence theory (uncertainty reasoning theory) [141]

6.1 Conventional approach

6.1.1 Pooling strategy

A typical approach for training a classifier is pooling the data from multiple subjects to train a single classifier. The details for such a process is described in this subsection. As for all of the other methods used throughout this Thesis, to ensure subject independence, the data pertaining to a single subject, denoted S_i, is successively held out. Data from all other subjects is then combined and divided in a stratified manner into a training set (comprising 80% of the data: $|S_{\text{train}}| = 0.8|S - S_i|$) and a validation set (20%, $|S_{\text{val}}| = 0.2|S - S_i|$) [142]. Here, $|S|$ denotes the total number of observations available in S, with $S = \cup_{i=1}^{N} S_i$, and N being the total number of subjects in a dataset. The

Algorithm 1 Pooled-data CNN

```
 1: Let $\mathcal{I} = \bigcup_{l=1}^{N} l$
 2: for $S_i \in S$
 3:     Partition $S - S_i$ into $S_{\text{train}}$ ($|S_{j,\text{train}}| = 0.8|S - S_i|$) and $S_{\text{val}} = S_j - S_{j,\text{train}}$
 4:         $\hat{acc}^* = 0$
 5:         for $\theta_h \in \Theta$
 6:             Train $\psi_{\theta_h, S_{\text{train}}, S_{\text{val}}}$
 7:             Check $\hat{acc}_{\theta_h, S_{\text{train}}, S_{\text{val}}}$
 8:             if $\hat{acc}_{\theta_h, S_{\text{train}}, S_{\text{val}}} > \hat{acc}^*$
 9:                 $\hat{acc}^* = \hat{acc}_{\theta_h, S_{\text{train}}, S_{\text{val}}}$
10:                 $\psi^*_{S_{\text{train}}, S_{\text{val}}} = \psi_{\theta_h, S_{\text{train}}, S_{\text{val}}}$
11:     Report the performance of $\psi_i$ on $S_i$
```

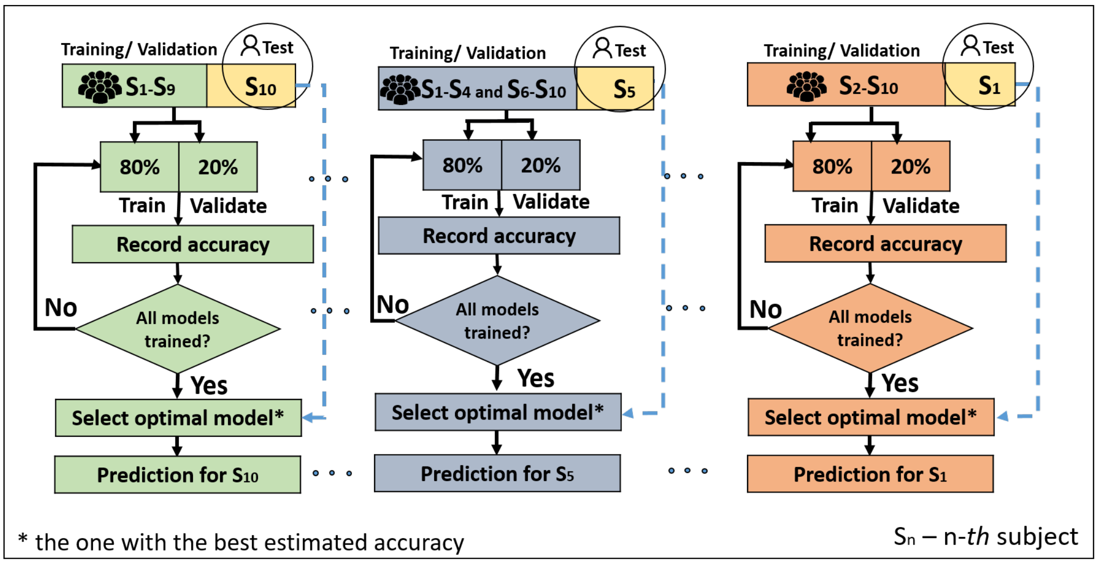

Figure 6.1: Graphical representation of training process for pooled-data CNN classifier.

validation set is utilized for tuning hyperparameters of a single CNN classifier. In the context of this study, a CNN classifier trained using the aforementioned procedure is referred to as a *pooled-data CNN*. The process for training pooled-data CNN classifier is illustrated in Fig.6.1 and elaborated in Algorithm 1. This process follows brute-force search model selection (see lines 5-10 in Algorithm 1) to define the optimal model (the one with the highest estimated accuracy) that is eventually used to decode the data of test subject.

6.1.2 Performance of Pooled-data CNN

The performance of the conventional approach was verified on the ERP-based BNCI dataset. The accuracies of the pooled-data CNN classifiers for each test subject are presented in Table 6.2. It should be highlighted that each result represents the performance of the model with optimal hyperparameters, these are the ones that demonstrated the best accuracy on the validation set among other CNNs defined by the hyperparameter space in the BFS process (Section 5.1). None of the models are evaluated on the test subject before the optimal model is obtained. The test subject is used only once for the final evaluation of the selected optimal model and is never seen in the training and model selection process. On average (across 10 subjects) the accuracy of $70.53 \pm 6.93\%$ was achieved.

Table 6.2: SI performance of pooled-data CNN classifier achieved on BNCI dataset.

Test subject	**Pooled-data CNN, %**
1	78.80
2	66.80
3	73.70
4	77.90
5	63.30
6	77.80
7	63.80
8	76.30
9	62.50
10	64.40
Avg	**70.53**
STD	**6.93**

6.2 Subject-Specific Training and Model Selection (SS-TM)

6.2.1 SS-TM strategy

One of the strategies that could be used to train the set of base classifiers for the ensemble classifier is to perform subject-specific training and model selection (SS-TM). This approach is presented in Figure 6.2 and Algorithm 2. As previously explained in Section 4.6, first LOSO-CV is employed. The S_i is isolated and an ensemble classifier using the data from all other subjects ($S - S_i$) is built. SS-TM technique is implemented within the set of remaining subjects. This involves randomly dividing the data for each S_j into two parts: an 80% training set ($S_{j,\text{train}}$, $|S_{j,\text{train}}| = 0.8|S_j|$), and a 20% validation set ($S_{j,\text{val}}$, $|S_{j,\text{val}}| = 0.2|S_j|$). Subsequently, for each S_j and each parameter $\theta_h \in \Theta$ a classifier $\psi_{\theta_h, S_{\text{j,train}}, S_{j,\text{val}}}$ is constructed, and the one with the highest accuracy, denoted $\hat{\text{acc}}_{\theta_h, Sj,\text{train}, S_{j,\text{val}}}$ is selected. Such an ensemble classifier is referred to as $\psi^*_{S_{j,\text{train}}, S_{j,\text{val}}}$. Consequently, each base classifier is trained specifically for an individual subject present in the training data. Finally, the ensemble classifier is employed to classify observations for the subject S_i that was withheld during training.

Algorithm 2 SS-TM based approach MS-En-CNN

```
 1: Let $\mathcal{I} = \bigcup_{l=1}^{N} l$
 2: for $S_i \in S$
 3:     for $S_j \in S - S_i$
 4:         Partition $S_j$ into $S_{j,\text{train}}$ and $S_{j,\text{val}} = S_j - S_{j,\text{train}}$
 5:         $\hat{\text{acc}}^* = 0$
 6:         for $\theta_h \in \Theta$
 7:             Train $\psi_{\theta_h, S_{j,\text{train}}, S_{j,\text{val}}}$
 8:             Check $\hat{\text{acc}}_{\theta_h, S_{j,\text{train}}, S_{j,\text{val}}}$
 9:             if $\hat{\text{acc}}_{\theta_h, S_{j,\text{train}}, S_{j,\text{val}}} > \hat{\text{acc}}^*$
10:                 $\hat{\text{acc}}^* = \hat{\text{acc}}_{\theta_h, S_{j,\text{train}}, S_{j,\text{val}}}$
11:                 $\psi^*_{S_{j,\text{train}}, S_{j,\text{val}}} = \psi_{\theta_h, S_{j,\text{train}}, S_{j,\text{val}}}$
12:     $\psi_i^{\text{E}} = \mathcal{C}\left(\bigcup_{r \in \mathcal{I}-\{i\}} \{\psi^*_{S_{r,\text{train}}, S_{r,\text{val}}}\}\right)$ where $\mathcal{C}$ is a combiner
13:     Report the performance of $\psi_i^{\text{E}}$ on $S_i$
```

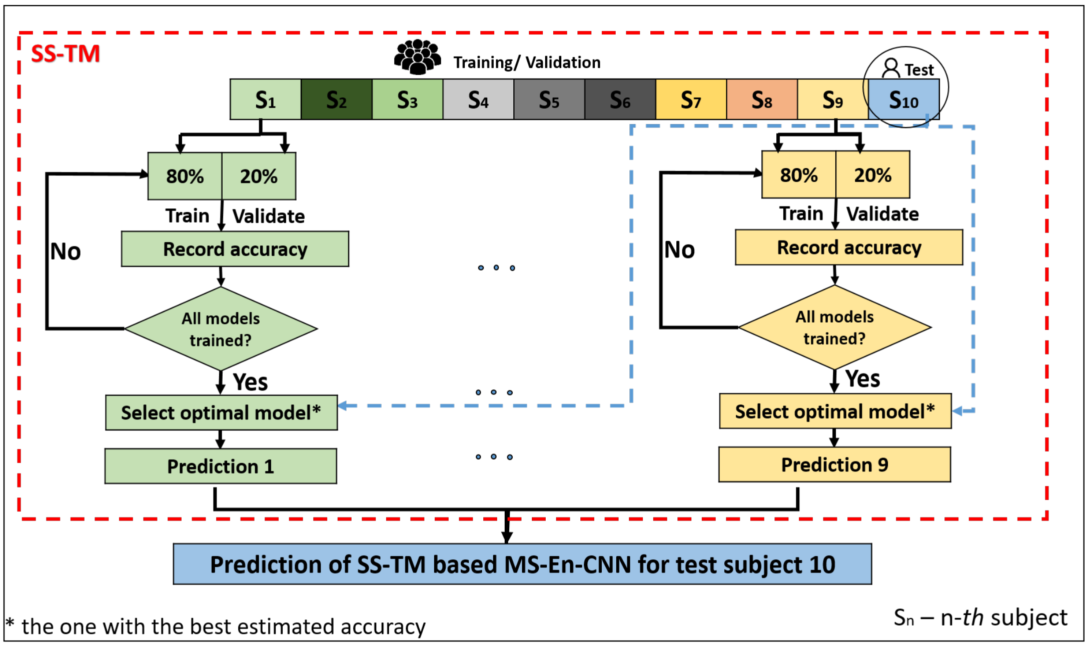

Figure 6.2: Training process for SS-TM based MS-En-CNN.

6.2.2 Performance of the SS-TM based MS-En-CNN

The method described above was evaluated on the BNCI dataset. This dataset consists of 10 subjects, thus considering the implementation procedure of the SS-TM method, the MS-En-CNN consists of 9 base classifiers. The architectures for all of the base classifiers are determined by the exhaustive search within the specified parameter space (Section 5.1). The statistics of the classification performance of these base classifiers are presented in Table 6.3, where minimum, maximum, and average SI test accuracy scores are presented. In addition, the accuracies achieved by the ensemble using majority vote (MV) combining scheme (Eq.4.2 with $w_m = 1$) and PCEW (Eq.4.4) are tabulated. In

Table 6.3: Test accuracy results achieved on BNCI dataset for MV and PCEW (with $\alpha = 1$ and $\alpha = 5$) ensembling schemes using SS-TM to establish base classifiers. Test accuracies of the base classifiers in columns columns 2-4 (minimum, maximum and average) are determined across 9 models.

Test subject	**Base classifiers, %**			**MS-En-CNN (SS-TM), %**		
	Minimum	**Maximum**	**Average**	**MV**	**PCEW** ($\alpha = 1$)	**PCEW** ($\alpha = 5$)
1	53.60	70.70	61.70	67.30	67.80	67.80
2	63.20	74.20	68.70	76.40	77.00	77.20
3	62.80	75.20	70.90	77.00	76.80	76.80
4	63.00	72.40	69.20	77.90	77.70	77.70
5	65.90	83.40	72.80	84.40	85.40	85.30
6	58.30	70.60	64.30	71.80	72.10	71.50
7	54.50	74.20	62.70	67.50	68.50	67.80
8	55.50	72.30	64.60	70.80	71.00	70.70
9	57.60	79.60	66.80	72.60	73.10	72.10
10	58.30	80.90	71.00	81.50	81.90	81.70
Avg	**59.27**	**75.35**	**67.27**	**74.72**	**75.13**	**74.86**
STD	**4.21**	**4.45**	**3.82**	**5.71**	**5.70**	**5.86**

the case of PCEW, the degree of trust to the estimated base classifiers' accuracies (α) is considered as hyperparemeter and takes values of 1 and 5. In addition to MV and PCEW, Naive Ensemble Weighting (NEW) combiner was considered. This combiner follows the general formula of hard voting (Eq.4.2) with the difference to MV in the way the weight w_m is determined. In NEW, the estimated accuracies of the base models (denoted as $\hat{\mathrm{acc}}_m$) are used to determine the weights $w_m = \frac{\hat{\mathrm{acc}}_m}{\sum_{j=1}^{M} \hat{\mathrm{acc}}_j}$, such that all weights are positive and they sum to one [143]. This *naive* way of estimating w_m is directly based on the consensus that a classifier with higher accuracy is more reliable and should weigh more in the process of ensemble classification [143]. However, this approach demonstrated classification performance similar to that of the MV combining approach. Consequently, these results are not included in Table 6.3.

For all of the columns in Table 6.3 the average performance across 10 test subjects and standard deviation is calculated. In all aspects presented in Table 6.3, the worst and best performance is associated with the test subjects 1 and 5, respectively. The lowest improvement achieved by the SS-TM based MS-En-CNN using MV with respect to the average performance of the base classifiers is 4.80%, while on average (among 10 test subjects) the SI accuracy is increased by 7.4%.

6.3 Subject Pairs Training and Model Selection (SP-TM)

6.3.1 SP-TM strategy

In the second method, the subject pairs were formed such that to construct a base classifier that is trained on $S_k \in S - S_i$, while the model selection process is carried out using other subject's data, $S_j \in S - S_k - S_i$. Based on such a procedure this method is referred to as Subject Pairs Training and Model Selection (SP-TM) approach. It is

Algorithm 3 SP-TM based approach MS-En-CNN

```
1:  Let $\mathcal{I} = \bigcup_{l=1}^{N} l$
2:  for $S_i \in S$
3:      for $S_j \in S - S_i$
4:          for $S_k \in S - S_i - S_j$
5:              $\hat{acc}^* = 0$
6:              for $\theta_h \in \Theta$
7:                  Train $\psi_{\theta_h,S_k,S_j}$
8:                  Check $\hat{acc}_{\theta_h,S_k,S_j}$
9:                  if $\hat{acc}_{\theta_h,S_k,S_j} > \hat{acc}^*$
10:                     $\hat{acc}^* = \hat{acc}_{\theta_h,S_k,S_j}$
11:                     $\psi^*_{S_k,S_j} = \psi_{\theta_h,S_k,S_j}$
12:     $\psi_i^{\mathrm{E}} = \mathcal{C}\left(\bigcup_{r\in\mathcal{I}-\{i\},a\in\mathcal{I}-\{i\}-\{r\}} \{\psi^*_{S_a,S_r}\}\right)$ where $\mathcal{C}$ is a combiner
13:     Report the performance of $\psi_i^{\mathrm{E}}$ on $S_i$
```

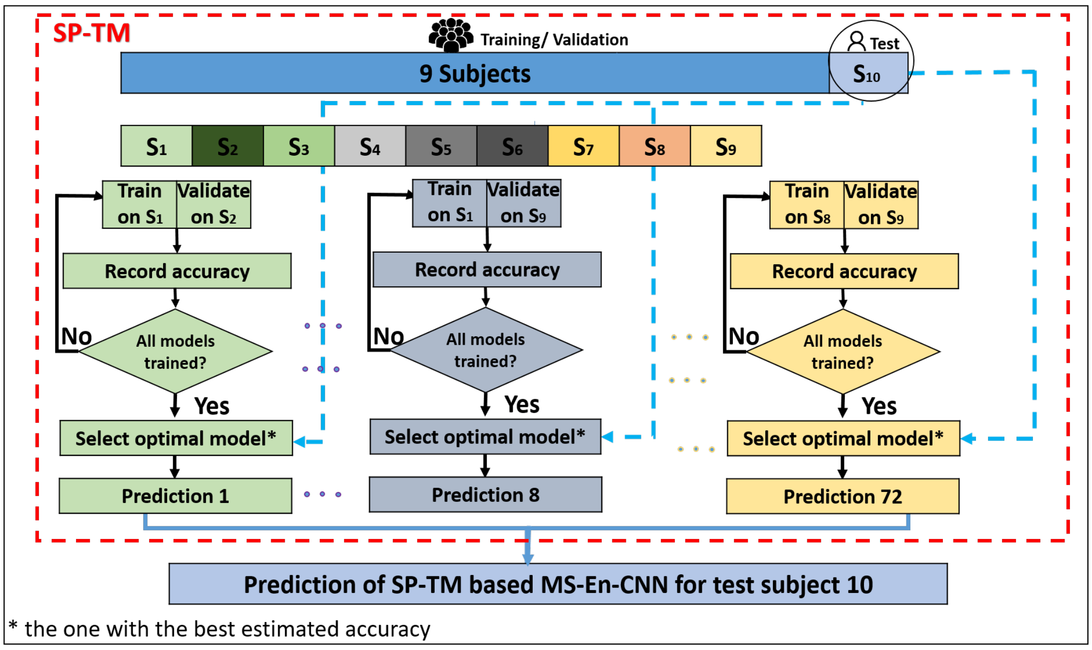

Figure 6.3: Training process for SP-TM based MS-En-CNN.

visually represented in Fig.6.3 and described in Algorithm 3. The process continues until all pairs of subjects within the training data $S - S_i$ have been used for both training and model selection. Consequently, in the context of a dataset comprising N subjects, with each subject being sequentially set aside for testing (SI classification), a total of $(N-1) \times (N-2)$ base classifiers are constructed to form a single ensemble classifier.

6.3.2 Performance of the SP-TM based MS-En-CNN

Similar to the SS-TM method, SP-TM-based MS-En-CNN approach is verified on the BNCI dataset. Considering the SP-TM strategy, in this MS-En-CNN classifier there are 72 base models. The results for the ensemble and its base classifiers are shown in Table 6.4. From this table, it can be seen that similar to the previous case, the lowest performance is associated with test subject 1. The best performance of the base classifiers in terms of minimum and maximum accuracy is demonstrated on the data of test subject 10, while the highest accuracy in terms of average among nine base classifiers and the performance of the ensemble is achieved for test subject 5. The SI accuracies of SP-TM based MS-En-CNN classifier using MV are higher than the average of its base classifiers by at least 3.90%, with the highest difference in the case of subject 9 (improvement by 10.40%).

Table 6.4: Test accuracy results achieved on BNCI dataset for MV and PCEW (with $\alpha = 1$ and $\alpha = 5$) ensembling schemes using SP-TM to establish base classifiers. Test accuracies of the base classifiers in columns columns 2-4 (minimum, maximum and average) are determined across 72 models.

Test subject	**Base classifiers, %**			**MS-En-CNN (SP-TM), %**		
	Minimum	**Maximum**	**Average**	**MV**	**PCEW** $(\alpha = 1)$	**PCEW** $(\alpha = 5)$
1	53.30	69.40	62.10	66.00	66.60	66.00
2	60.10	75.90	68.20	72.70	72.70	72.70
3	56.80	76.30	70.90	76.30	76.60	76.80
4	63.90	78.90	71.30	78.80	78.90	78.90
5	60.70	84.20	75.80	84.10	84.50	84.70
6	56.20	72.70	64.50	71.00	70.40	70.20
7	53.70	74.70	64.20	69.90	70.20	70.00
8	59.30	74.60	67.30	72.90	72.70	72.80
9	60.30	82.40	73.50	83.90	84.00	83.50
10	64.10	86.70	74.00	81.30	82.00	81.90
Avg	**58.84**	**77.58**	**69.18**	**75.69**	**75.86**	**75.75**
STD	**3.78**	**5.42**	**4.65**	**6.20**	**6.29**	**6.38**

6.4 Jackknife-inspired Deep Learning Approach

In this section a systematic approach that takes inspiration from the concept of jackknife estimation [144, 145, 146] is introduced. The fundamental idea behind the approach of jackknife resampling is to construct a novel statistical estimator, systematically omit each observation from a dataset, compute the parameter estimate using the remaining observations, and then aggregate all these individual estimates [146].

The jackknife method and its extensions have gained popularity in the field of machine learning for assessing the precision, variability, and robustness of models. Alaa and Schaar [147] have specifically considered quantification of uncertainty in deep learning models, while Kang *et al.* [148] have presented deterministic uncertainty quantification for Graph Convolutional Networks using the jackknife estimator. Further examples of the jackknife estimation's application include estimating covariance between different models and its use in quantifying uncertainty in ensemble methods, which can be found in references [149, 150, 151]. In the work by Wager *et al.* [149], it is demonstrated that the jackknife and infinitesimal jackknife methods are effective for estimating uncertainty associated with random forest predictions and for constructing valid confidence intervals. The authors suggest the integration of the estimates from both the jackknife and infinitesimal jackknife by averaging them. This approach is grounded in the idea that the biases in the individual approaches can, to some extent, offset each other, leading to a variance estimate that is closer to being unbiased. Furthermore, Ghosal and Hooker [150] estimated the variance of boosted random forests by utilizing an extension of the infinitesimal jackknife method, while in [151], Ghosal *et al.* used the infinitesimal jackknife to compute covariance between different models, offering a means to quantify uncertainty in ensembles.

Taking inspiration from the concept of jackknife estimation, which involves aggregating parameter estimates from individual subsamples created by excluding one observation at a time, an ensemble classifier is developed. This ensemble integrates outputs from classifiers trained on subsamples created by excluding all data associated with a specific individual. The base classifier's performance on unseen subjects is enhanced by conducting tuning of hyperparameters specifically for every individual base learner on the subject left out during training. First, building upon a few reasonable assumptions, a theoretical perspective on why the proposed ensemble outperforms a single model that underwent training on the entire dataset is presented. Then, the proposed strategy is described and its performance is demonstrated.

6.4.1 Motivation behind the proposed approach

Using data of M training subjects (not to confuse with N number of total subjects in a given dataset), the conventional method for creating an SI classifier involves aggregating data from all subjects (see Section 6.1), training a classifier, and applying this trained classifier to data collected from a different, independent subject. When the input $\mathbf{x}_i$ is presented into a CNN classifier, denoted $\psi(\mathbf{x}_i)$, it produces output estimates for class y_i, denoted p_{y_i}, where $i = 1, \ldots, n$ and $y_i = 0, \ldots, c-1$ with n and c being the total number of observations and total number of classes, respectively. The classifier subsequently operates as

$$\psi(\mathbf{x}_i) = \underset{y_i \in \{0,\ldots,c-1\}}{\arg\max}\; p_{y_i}\,. \tag{6.1}$$

Inspired by the methodology of jackknife estimation technique [145, 146, 152], the impact of the following process is examined in this subsection: all data related to subject m is successively excluded, where $m = 1, \ldots M$; the data of other $M-1$ subjects is pooled together; a classifier $\psi_m(\mathbf{x}_i)$ is trained employing the same procedure as for $\psi(\mathbf{x}_i)$; finally a collective decision is made. The decision-making process is based on the estimates of posterior probabilities produced by the classifier $\psi_m(\mathbf{x}_i)$, denoted $p_{y_i}^m$, for each class y_i. Given $p_{y_i}^m$ for all y_i and all m, the decision using soft voting [119] is

$$\psi^{\mathrm{E}}(\mathbf{x}_i) = \underset{y_i \in \{0,\ldots,c-1\}}{\arg\max}\; \frac{1}{M} \sum_{m=1}^{M} p_{y_i}^m\,. \tag{6.2}$$

To simplify notation, a binary classification problem with $c = 2$ is examined. Given that posterior estimates depend on training samples, which are random vectors, both p_{y_i} and $p_{y_i}^m$, $\forall i$ and $\forall m$, are also treated as random variables. A pertinent question to pose is whether the subsequent inequality is valid:

$$\begin{aligned} P\left(\frac{(-1)^{y_i+1}}{M}\Big(\sum_{m=1}^{M} p_1^m - \sum_{m=1}^{M} p_0^m\Big) > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\right) > \\ P\Big((-1)^{y_i+1}\big(p_1 - p_0\big) > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\Big)\,, \end{aligned} \tag{6.3}$$

where, with y_i taking values of 0 or 1, and π_{y_i} representing population y_i. If inequality (6.3) holds true, it suggests that $\psi^{\mathrm{E}}(\mathbf{x}_i)$ is more likely to correctly classify an observation $\mathbf{x}_i$ than $\psi(\mathbf{x}_i)$. To simplify, it is assumed that $y_i = 1$ and the definitions $z \triangleq p_1 - p_0$ and $z^m \triangleq p_1^m - p_0^m$ are introduced. Therefore, with $z^{\mathrm{E}} = \frac{1}{M}\sum_{m=1}^{M} z^m$ the objective is to demonstrate

$$P\left(z^{\mathrm{E}} > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\right) > P\Big(z > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\Big)\,. \tag{6.4}$$

The following assumptions are established:

Assumption 1: It is reasonable expectation that a classifier $\psi(\mathbf{x}_i)$ performs better than random guessing: $P\Big(z > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\Big) > 0.5$.

Assumption 2: It is assumed that z follows a normal distribution with a mean μ and variance σ^2, and similarly, z^m follows a normal distribution with a mean of $\mu - \epsilon$ and the same variance, where m ranges from 1 to M. This means that z is expected to have a positive mean ($\mu > 0$). A larger μ corresponds to a higher probability of correctly classifying $\mathbf{x}_i \in \pi_1$ by $\psi(\mathbf{x}_i)$. It is also assumed that all z^m, for all m, share identical normal distributions with a mean of $\mu - \epsilon$, where $\epsilon > 0$. This assumption is based on the fact that each $\psi_m(\mathbf{x}_i)$ is trained with one less subject than $\psi(\mathbf{x}_i)$, and it is reasonable to expect that, with well-behaved learning algorithms, $\psi_m(\mathbf{x}_i)$ is more likely to misclassify a random feature vector $\mathbf{x}_i \in \pi_1$ compared to $\psi(\mathbf{x}_i)$.

Building upon Assumption 2 and presuming a uniform correlation of at least $\rho \geq 0$ among all pairs of z^i and z^j, it can be deduced that z^{E} follows a normal distribution with parameters $\mu - \epsilon$ and $\frac{\sigma^2}{M}\Big((M-1)\rho + 1\Big)$. Consequently, the expression on the left side of (6.4) can be expressed as

$$P\left(z^{\mathrm{E}} > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\right) = 1 - \Phi\left(M \frac{\epsilon - \mu}{\sigma^2\Big((M-1)\rho + 1\Big)}\right), \tag{6.5}$$

where, $\Phi(.)$ represents the cumulative distribution function of the standard normal random variable. Under Assumption 2, the expression on the right side of (6.4) becomes

$$P\Big(z > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\Big) = 1 - \Phi\Big(\frac{-\mu}{\sigma^2}\Big). \tag{6.6}$$

As μ is greater than 0, a comparison between (6.5) and (6.6) reveals that (given Assumptions 1 and 2)

$$\epsilon < \mu(1-\rho)\big[1 - \frac{1}{M}\big] \iff P\left(z^{\mathrm{E}} > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\right) > P\Big(z > 0 \,|\, \mathbf{x}_i \in \pi_{y_i}\Big). \tag{6.7}$$

Expression (6.7) illustrates that as ρ increases, a smaller value of ϵ is required, indicating a reduced impact of subject removal, for $\psi^{\mathrm{E}}(\mathbf{x}_i)$ to remain advantageous over $\psi(\mathbf{x}_i)$. In the extreme case where $\rho = 1$, any value of ϵ results in a lower probability of correct classification for $\psi^{\mathrm{E}}(\mathbf{x}_i)$ compared to $\psi(\mathbf{x}_i)$. Conversely, when $\rho = 0$ there is plenty of room for ϵ to render $\psi^{\mathrm{E}}(\mathbf{x}_i)$ superior to $\psi(\mathbf{x}_i)$. Additionally, from (6.7), it is evident that as M increases (indicating a larger number of subjects), there is more flexibility for ϵ to enhance the benefits of $\psi^{\mathrm{E}}(\mathbf{x}_i)$.

6.4.2 “Delete-a-Subject Jackknife" (DASJ) strategy

Motivated by these observations, here the jackknife-inspired procedure for constructing an ensemble is discussed. Each classifier in this ensemble is trained on a “Delete-a-Subject Jackknife" (DASJ) sample. In this method (refer to Fig.6.4 and Algorithm

Algorithm 4 **DASJ sample based approach MS-En-CNN**

```
1:  Let $\mathcal{I} = \bigcup_{l=1}^{N} l$
2:  for $S_i \in S$
3:      for $S_j \in S - S_i$
4:          $\hat{acc}^* = 0$
5:          for $\theta_h \in \Theta$
6:              Train $\psi_{\theta_h, S-S_i-S_j, S_j}$
7:              Check $\hat{acc}_{\theta_h, S-S_i-S_j, S_j}$
8:              if $\hat{acc}_{\theta_h, S-S_i-S_j, S_j} > \hat{acc}^*$
9:                  $\hat{acc}^* = \hat{acc}_{\theta_h, S-S_i-S_j, S_j}$
10:                 $\psi^*_{S-S_i-S_j, S_j} = \psi_{\theta_h, S-S_i-S_j, S_j}$
11:     $\psi^{\mathrm{E}}_i = \mathcal{C}\left(\bigcup_{r \in \mathcal{I}-\{i\}} \{\psi^*_{S-S_i-S_r, S_r}\}\right)$ where $\mathcal{C}$ is a combiner
12:     Report the performance of $\psi^{\mathrm{E}}_i$ on $S_i$
```

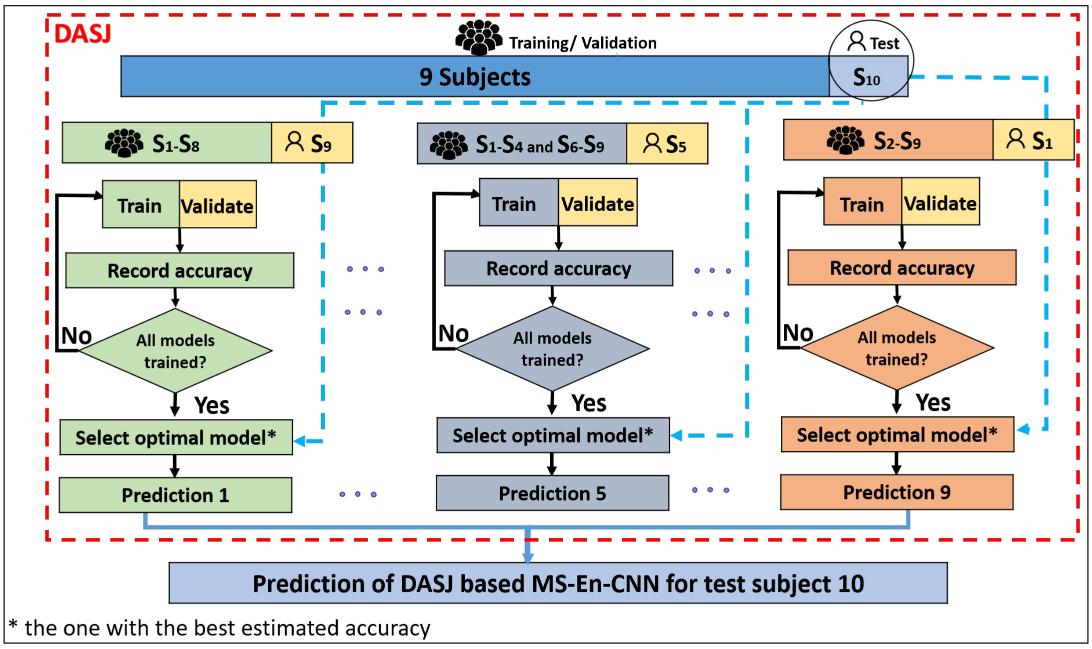

Figure 6.4: Training process for DASJ based MS-En-CNN.

4), after the data collected for one of the subjects (S_i) is set aside (the LOSO-CV for testing), a different single subject is used for model selection. The aim is to replicate the SI context within the training data $S - S_i$ and identify the optimal set of hyperparameters for training the base classifier. To clarify further, in this approach, for the dataset of N subjects, to create an ensemble classifier used for classifying S_i, $N - 1$ base classifiers are constructed. Each of these classifiers is trained using data from $M - 1 = N - 2$ subjects within $S - S_i - S_j$, where $j \neq i$ and M is the number of training subject. The hyperparameters are tuned using data of the held-out subject within the training set (S_j). These $N - 1$ base classifiers are then utilized to implement the ensemble classifier, which in turn classifies observations for the held-out test subject i.

6.4.3 Performance of the DASJ based MS-En-CNN

The classification performance of the DASJ sample based approach with the base classifiers' architecture being defined via brute-force search is presented in Table 6.5. Considering the performance of the base classifiers, the worst and best minimum/maximum accuracy scores (among other subjects) were demonstrated on the data of test subject 1 and test subject 10, respectively. The worst average performance among 9 base classifiers, as well as for the ensemble (using MV), was achieved on the data of test subject 1 (70.70% and 74.20%), while the best results are achieved for test subject 9 (84.00% and 87.30%). The accuracies obtained using MS-En-CNN classifier is greater than the average classification accuracies achieved by their base classifiers by more than 2% for most of the test subjects (except for subject 3 and 4, where the improvement is 1.10% and 1.50%, respectively). On average, this improvement amounts to 2.60%.

Table 6.5: Test accuracy results achieved on BNCI dataset for MV and PCEW (with $\alpha = 1$ and $\alpha = 5$) ensembling schemes using DASJ samples to establish base classifiers. Test accuracies of the base classifiers in columns columns 2-4 (minimum, maximum and average) are determined across 9 models.

Test subject	**Base classifiers, %**			**MS-En-CNN (DASJ), %**		
	Minimum	**Maximum**	**Average**	**MV**	**PCEW** ($\alpha = 1$)	**PCEW** ($\alpha = 5$)
1	65.10	75.10	70.70	74.20	73.90	73.90
2	72.70	79.80	77.20	81.30	80.30	80.20
3	74.50	78.90	77.10	78.20	78.60	78.80
4	74.20	81.50	78.60	80.10	79.60	79.40
5	76.90	82.40	80.40	82.80	83.90	83.80
6	70.00	74.90	72.60	75.00	74.50	74.80
7	66.40	76.80	72.50	75.50	75.60	75.70
8	67.80	74.90	72.20	74.20	74.50	74.60
9	80.20	86.70	84.00	87.30	87.60	87.60
10	74.30	81.70	78.60	81.60	82.30	82.40
Avg	**72.21**	**79.27**	**76.39**	**79.02**	**79.08**	**79.12**
STD	**4.81**	**3.91**	**4.27**	**4.36**	**4.58**	**4.52**

6.5 Key Findings

The last three sections of this chapter were dedicated to three distinct methods for training base classifiers of the MS-En-CNN classifier (SS-TM, SP-TM, and DASJ sample based approach). In this section a summary of those results, their comparison, and a short discussion are presented. Additionally, the comparison with the pooled-data CNN classifier (Section 6.1) is provided.

Upon analyzing the results presented in this Chapter, it becomes evident that the classification results exhibit a high degree of variance. Similar observations can be drawn from Chapter 3, where the methods reported in the literature are summarized with their achieved classification performance across various datasets. The variability in performance can arise from factors such as subject-specific neurophysiological characteristics, inherent variability in the neural signals being decoded, and electrode positioning variability [153]. This reflects the inherent challenges and complexities in decoding neural signals for practical applications. It is important to note that the proposed strategies demonstrate not only improved average classification accuracy but, in most cases, also reduced standard deviation. This suggests that the presented approaches not only enhance overall performance but also contribute to the reduction of variability across subjects, hence providing better generalizability.

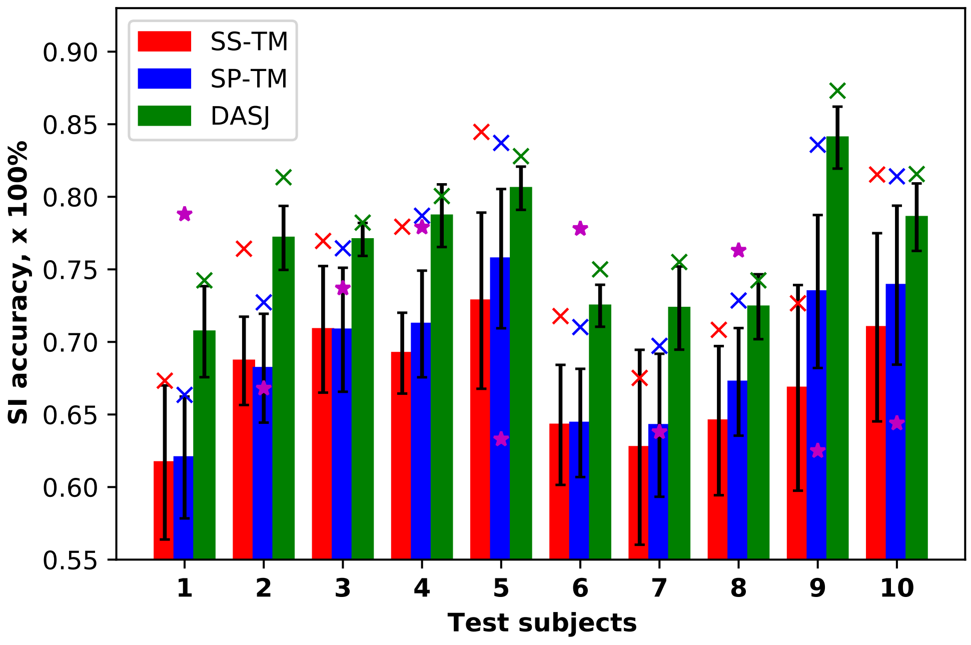

Figure 6.5: SI accuracies for three distinct ensemble methods and its base classifiers, as well as the pooled-data CNN. The bars display the mean test accuracies along with their respective standard deviations (indicated by error bars) for base classifiers trained with SS-TM, SP-TM, and DASJ approaches. Above the bars, "×" symbols denote the test accuracy of the ensemble (MS-En-CNN classifier) employing the MV combiner. Additionally, a magenta star symbol signifies the accuracy achieved for each subject using the pooled-data CNN classifier.

As was previously demonstrated, the accuracies for SI classification obtained using MS-En-CNN classifiers surpass the average results achieved by their respective base classifiers for all three strategies (SS-TM, SP-TM, DASJ) and across all test subjects. These findings are graphically illustrated in Fig.6.5, where the SI accuracies of three types of MS-En-CNN methods are presented not only with respect to its base classifiers, but also with respect to pooled-data CNN. Notably, in 25 out of 30 instances (corresponding to 10 subjects across the three strategies), the accuracy attained by MS-En-CNN classifiers exceeds the average accuracy plus their corresponding standard deviation. This difference is statistically significant, as confirmed by a one-sided (paired) Wilcoxon signed-rank test. The alternative hypothesis in this test posits that the performance achieved by the ensemble classifier is greater than the average accuracy of its base classifiers. The calculated *P*-values for each MS-En-CNN strategy (SS-TM, DASJ, and SP-TM) are all less than 0.001, demonstrating the significance with the common significance level of $\alpha = 0.05$.

If the results achieved through the three different strategies for training base classifiers are compared, it becomes evident that the DASJ-based approach generally outperforms SS-TM and SP-TM in terms of SI classification accuracy. This assertion is substantiated through statistical analysis, as shown in Table 6.6, where *P*-values resulting from one-sided Wilcoxon signed-rank tests are demonstrated.

The advantage of DASJ over the other two methods is also evident in Fig.6.5. Not only does the classification accuracy achieved by base classifiers trained using DASJ tend to be higher on average compared to the other two methods, but it also exhibits a lower standard deviation. This lower standard deviation signifies that base classifiers trained with DASJ are more consistently reliable. This enhanced robustness of base classifiers trained using DASJ can be attributed, in part, to the substantial overlap in observations used during their training. Referring to the mechanism detailed in DASJ (see Algorithm 4), it becomes apparent that there are observations from 8 subjects in common shared during the training process of each pair of base classifiers when using this method. To clarify, the term “training" specifically refers to the backpropagation process and not to model selection, with a clear distinction made between the “training set" and the “validation set". In contrast, SS-TM trains all base classifiers on distinct sets of observations, and for SP-TM, the 72 base classifiers are organized into 9 groups. Within each group, there are observations from one subject shared and used to train 8 classifiers, but there are no overlaps between the training sets employed across these groups.

The comparison between the SI classification performance of MS-En-CNN and pooled-data CNN reveals significant improvements in the former case when using any of the base classifier training methods. Specifically, when employing SS-TM, SP-TM,

Table 6.6: Results of one-sided Wilcoxon signed-rank test for comparison of the classification accuracy of DASJ sample based vs. SS-TM and SP-TM strategies (*P*-values).

	DASJ vs. SS-TM	**DASJ vs. SP-TM**
***P*-value**	.005	.007

or DASJ, MS-En-CNN achieves accuracy rates of 74.70% ± 5.70%, 75.80% ± 6.20%, and 79.00% ± 4.40% respectively, compared to the pooled-data CNN's accuracy of 70.50% ± 6.90%. In addition to SI accuracy, various other metrics to evaluate the performance of both pooled-data CNN classifier and the three MS-En-CNN learning strategies (SS-TM, SP-TM, DASJ) are presented in Table 6.7. This include Area under the Receiver Operating Characteristic Curve (AUC), recall, precision, and f_1-score. The confusion matrix for all test subjects and each method is presented in Table 6.8. As shown, the MS-En-CNN classification approach consistently demonstrates

Table 6.7: Performance measures (accuracy, AUC, recall, precision and f1-score) for pooled-data CNN and the SS-TM, SP-TM and DASJ sample based MS-En-CNN ensembles.

Parameters	**Pooled-data CNN, %**	**SS-TM, %**	**DASJ, %**	**SP-TM, %**
Test subject 1				
Accuracy	78.80	67.30	74.20	66.00
AUC	79.70	69.00	75.00	68.00
Precision	70.10	51.70	66.70	48.90
Recall	88.80	81.70	82.70	81.40
f_1-score	78.30	63.30	73.90	61.10
Test subject 2				
Accuracy	67.80	76.40	81.30	72.70
AUC	69.20	77.00	82.00	74.00
Precision	42.30	65.80	77.70	59.60
Recall	93.00	87.90	86.70	86.10
f_1-score	58.10	75.20	81.90	70.50
Test subject 3				
Accuracy	73.70	77.00	78.20	76.30
AUC	74.60	77.00	77.00	76.00
Precision	65.00	80.30	88.70	80.00
Recall	83.10	78.10	75.60	77.40
f_1-score	73.00	79.10	81.60	78.70
Test subject 4				
Accuracy	77.90	77.90	80.10	78.80
AUC	78.70	78.00	80.00	79.00
Precision	69.70	73.40	85.10	75.20
Recall	87.20	84.10	79.60	84.10
f_1-score	77.50	78.40	82.30	79.50
Test subject 5				
Accuracy	63.30	84.40	82.80	84.10
AUC	65.60	85.00	82.00	84.00
Precision	40.60	82.60	90.40	83.30
Recall	83.70	88.20	80.50	87.10
f_1-score	54.60	85.30	85.10	85.10

Table 6.7: Continued.

Parameters	**Pooled-data CNN, %**	**SS-TM, %**	**DASJ, %**	**SP-TM, %**
Test subject 6				
Accuracy	77.80	71.80	75.00	71.00
AUC	77.90	73.00	75.00	72.00
Preci- sion	76.60	58.40	70.70	56.70
Recall	81.60	85.20	81.00	85.20
f_1**-score**	79.00	69.30	75.50	68.10
Test subject 7				
Accuracy	63.80	67.50	75.50	69.90
AUC	66.20	69.00	76.00	71.00
Precision	38.90	48.60	68.70	55.40
Recall	87.90	85.60	83.50	83.90
f_1**-score**	53.90	62.00	75.40	66.70
Test subject 8				
Accuracy	76.30	70.80	74.20	72.90
AUC	77.30	72.00	74.00	74.00
Precision	66.00	82.80	75.50	65.20
Recall	87.50	79.40	76.80	81.50
f_1**-score**	75.30	70.10	76.20	72.40
Test subject 9				
Accuracy	62.50	72.60	87.30	83.90
AUC	63.80	74.00	87.00	84.00
Precision	50.10	55.80	90.90	80.00
Recall	72.70	90.40	86.50	89.30
f_1**-score**	59.30	69.00	88.60	84.40
Test subject 10				
Accuracy	64.40	81.50	81.60	81.30
AUC	66.20	82.00	81.00	81.00
Preci- sion	46.00	78.40	86.70	82.10
Recall	80.30	86.50	80.80	83.30
f_1**-score**	58.50	82.20	83.70	82.70
Average across 10 subjects $\pm$ STD				
Accuracy	70.53±6.93	74.72±5.71	79.02±4.36	75.69± 6.20
AUC	71.91±6.29	75.60±5.25	78.90 ± 4.17	76.30 ± 5.52
Recall	56.53±14.30	67.78±13.36	80.11± 9.37	68.64± 12.90
Precision	84.58±5.64	84.71±3.95	81.37± 3.64	83.93 ± 3.34
f_1**-score**	66.75±10.65	73.39±7.88	80.42± 4.90	74.92 ± 8.30

Table 6.8: Confusion matrices for pooled-data CNN and the SS-TM, SP-TM and DASJ sample based MS-En-CNN ensembles.

		Predicted		Predicted		Predicted		Predicted	
	Actual	P	N	P	N	P	N	P	N
1	P	1211	153	893	200	1153	241	845	193
	N	517	1287	835	1240	575	1199	883	1247
2	P	731	55	1137	156	1343	206	1031	166
	N	997	1385	591	1284	385	1234	697	1274
3	P	1124	229	1388	390	1533	495	1384	405
	N	604	1211	340	1050	195	945	344	1035
4	P	1204	177	1269	240	1472	376	1301	245
	N	524	1263	459	1200	256	1064	427	1195
5	P	701	136	1427	191	1562	379	1439	213
	N	1027	1304	301	1249	166	1061	289	1227
6	P	1323	298	1009	175	1222	286	980	170
	N	405	1142	719	1265	506	1154	784	1270
7	P	672	92	840	141	1187	235	958	184
	N	1056	1348	888	1299	541	1205	770	1256

Table 6.8: Continued.

		Predicted		Predicted		Predicted		Predicted	
	Actual	P	N	P	N	P	N	P	N
8	P	1141	163	1086	282	1305	393	1126	255
	N	587	1277	642	1158	423	1047	602	1185
9	P	866	325	964	102	1571	245	1383	166
	N	862	1115	764	1338	157	1195	345	1274
10	P	795	195	1355	212	1499	355	1419	284
	N	933	1245	373	1228	229	1085	309	1156
Average	P	977	182	1137	209	1385	321	1186	227
	N	751	1258	591	1231	343	1119	544	1211

Table 6.9: Results of one-sided Wilcoxon signed-rank test (*P*-values) for comparing the classification performance (in terms of accuracy, AUC, recall, precision and f_1-score) achieved by DASJ-based MS-En-CNN classifier vs. pooled-data CNN classifier.

	Accuracy	**AUC**	**Precision**	**Recall**	**f_1-score**
***P*-values**	.032	.065	.005	.958	.010

superior average performance across most of the mentioned metrics and exhibits a lower standard deviation in comparison to the CNN classifier based on pooled-data approach. Furthermore, the improvements achieved in accuracy, recall, and f_1-score are statistically significant according to one-sided (paired) Wilcoxon signed-rank tests at a common significance level of $\alpha = 0.05$. The tests indicate that the classification performance achieved by MS-En-CNN with the DASJ training strategy surpasses the average classification performance of the pooled-data CNN. The *P*-values for these statistical tests can be found in Table 6.9. The most substantial improvement is observed with the DASJ strategy, and it is statistically significant with *P*$= 0.032$. Notably, for certain test subjects, such as subjects 2, 5, and 9, DASJ-based MS-En-CNN outperforms the pooled-data CNN by margins of approximately 15%, 19%, and 25%, respectively, in terms of accuracy (refer to Table 6.7).

The preceding discussion highlights the effectiveness of the proposed DASJ-based MS-En-CNN in various scenarios. Specifically, comparisons were made to underscore its superiority over MS-En-CNNs employing alternative strategies (SS-TM, SP-TM), the average classification performance of its constituent base classifiers, and a conventional approach (pooled-data CNN). The respective P-values from the statistical one-sided

Wilcoxon signed rank test with the alternative hypothesis stating the significant improvement demonstrated by DASJ-based approach over other methods are: 0.005 (SS-TM), 0.007 (SP-TM), less than 0.001 (base classifiers), and 0.032 (pooled-data CNN). To address the challenge of multiple comparisons inherent in the above analysis, the False Discovery Rate (FDR) control method was employed. The calculated P-values were subject to FDR correction using the Benjamini-Hochberg procedure. After adjustment, the corrected significance levels remained below the predetermined threshold of 0.05, confirming the statistical significance of the observed differences in comparison of the DASJ-based approach to other methods while minimizing the risk of false positives.

6.6 Application to MI datasets

Given the promising results of the DASJ approach on ERP dataset, the same strategy has been applied to MI datasets, namely BCI IV 2A and BCI IV 2B. Table 6.10 displays test accuracies of the base classifiers (minimum, maximum, and average) and the test accuracies of the ensemble classifiers for two datasets. The results of ensembles are presented for different combining schemes. It can be noted that in addition to MV and PCEW, here the performance of NEW combination rule based MS-En-CNN is also tabulated. This is because, in contrast to the BNCI dataset cases, the results of NEW-based MS-En-CNN differ from those achieved using MV. On average MV based ensemble performs better than the NEW-based. Furthermore, a comparison between MV and PCEW reveals minimal differences in their performance. Consequently, only MV combination rule is utilized further, as it avoids the need for additional hyperparameter, such as α in PCEW.

When comparing the ensemble classifiers' accuracies to the average accuracies of the individual base classifiers on test data, a significant performance enhancement is evident in the ensemble scheme. To validate this observation statistically, a one-sided (paired) Wilcoxon signed-rank test was conducted. The resulting P-values for all pairwise comparisons between the performance of ensemble classifiers and the average performance of base classifiers are displayed in Table 6.11. Notably, all P-values are below 0.01, which is less than the commonly used significance level (0.05). This underscores the substantial improvement realized by the ensemble classifiers.

6.7 Summary

This chapter was dedicated to introducing the proposed methods. First, it defined a common approach used for comparing the designed methods. Next, it described various strategies for training the set of base classifiers employed within the MS-En-

Table 6.10: SI classification performance of DASJ-based MS-En-CNN (using MV, NEW, and PCEW combining schemes) classifier and test accuracies of its base classifiers (minimum, maximum, and average among 9 in total) using data from BCI IV 2A and 2B datasets.

	Test subject	**Base classifiers, %**			**MS-En-CNN (DASJ), %**			
		Minimum	**Maximum**	**Average**	**MV**	**NEW**	**PCEW** ($\alpha = 1$)	**PCEW** ($\alpha = 5$)
BCI IV 2A	1	63.31	70.72	66.35	70.60	67.36	71.88	72.22
	2	48.26	53.70	51.46	53.01	51.74	52.78	53.13
	3	64.93	77.31	71.08	78.82	76.04	77.89	76.16
	4	54.40	64.58	58.75	59.61	61.23	60.76	60.65
	5	52.08	56.71	54.01	56.02	52.66	56.48	54.86
	6	55.21	62.50	58.93	59.26	59.38	60.30	60.07
	7	53.36	61.81	58.09	61.57	56.13	61.34	60.42
	8	66.78	74.07	69.69	76.85	71.99	76.16	77.43
	9	58.56	66.44	62.50	61.69	67.59	61.46	62.62
	Avg	**57.43**	**65.32**	**61.21**	**64.16**	**62.68**	**64.34**	**64.17**
	STD	**6.35**	**7.75**	**6.76**	**9.12**	**8.57**	**8.82**	**8.94**
BCI IV 2B	1	59.63	65.00	62.39	64.17	64.72	65.00	64.58
	2	52.11	55.34	53.49	53.87	53.28	54.26	54.61
	3	52.31	54.91	53.69	54.12	55.19	53.84	53.70
	4	77.25	81.89	80.41	83.29	81.04	82.61	82.79
	5	57.88	64.46	62.53	63.96	63.33	65.81	64.37
	6	63.56	70.56	67.77	71.06	70.56	71.02	70.14
	7	61.25	66.85	64.37	65.74	63.75	65.83	66.02
	8	66.80	72.41	70.46	70.53	70.13	71.84	71.23
	9	65.51	70.32	68.09	69.77	69.58	70.79	70.51
	Avg	**61.81**	**66.86**	**64.80**	**66.28**	**65.73**	**66.78**	**66.44**
	STD	**7.79**	**8.42**	**8.35**	**9.04**	**8.43**	**8.94**	**8.88**

Table 6.11: Comparison of the classification accuracy of an ensemble against the average accuracy of the base classifiers for BCI IV 2A and 2B datasets based on *P*-values computed with one-sided Wilcoxon signed rank test.

Dataset	**Combiner vs. Average**		
	MV	**PCEW** ($\alpha = 1$)	**PCEW** ($\alpha = 5$)
BCI IV 2A	0.00586	0.00391	0.00195
BCI IV 2B	0.00195	0.00195	0.00195

CNN classifier. The distinction among these strategies lies in how the data is divided into training and validation subsets. The validation subsets play a crucial role in the model selection process, particularly in tuning the hyperparameters of the CNNs. The experiments presented in this chapter were conducted using a brute-force model selection approach.

Chapter 7

DASJ-CNN Ensemble with State-of-the-Art Base Models

In Chapter 6, MS-En-CNN ensembles were created using CNN base classifiers with their architectures determined by a brute-force search-based model selection process. The most promising approach appeared to be a DASJ sample based ensemble. In this chapter, the DASJ-based ensemble is investigated with the base classifiers having the architecture of the well-established CNN configurations. In this regard, the following architectures demonstrated the state-of-the-art performance in EEG-based BCI: ShallowConvNet, EEGNet, DeepConvNet. These three architectures were used to train the base classifiers of the ensemble according to the DASJ based strategy, which in general could be referred to as DASJ-CNN. When using ShallowConvNet, DeepConvNet or EEGNet architecture for the CNN base classifiers of MS-En-CNN, it's referred to as DASJ-ShallowConvNet, DASJ-DeepConvNet or DASJ-EEGNet, respectively.

7.1 CNN versus DASJ-CNN Ensemble

7.1.1 Training the Base Classifiers of MS-En-CNN

The training strategy remains the same as it was described in Section 6.4, except for the fact that instead of having the hyperparameter space to search through (defined in Section 5.1), there is a fixed architecture of CNN that is used (ShallowConvNet, DeepConvNet or EEGNet) (see Section 5.2). Each CNN base classifier underwent training for a maximum of 200 epochs without implementing early stopping. The batch size was configured at 64, and a learning rate of 0.001 was employed alongside a weight decay of 0.0001.

7.1.2 Subject-independent classification performance

The objective is to demonstrate that the DASJ-CNN ensemble classification rule systematically enhances the performance of current CNN architectures for EEG classification. A comparison is conducted between DASJ-ShallowConvNet and ShallowConvNet, DASJ-DeepConvNet and DeepConvNet, as well as between DASJ-EEGNet and EEGNet, to assess their SI classification performance. This comparison helps evaluate the effectiveness of the proposed DASJ-CNN ensemble classification rule in contrast to the conventional approach, which typically involves training a single classifier using 80% of the pooled data from training subjects and reserving the remaining 20% for hyperparameter tuning (see Section 6.1).

The experiments for both pooled-data CNN and DASJ sample based MS-En-CNN were conducted considering two cases of standardization. A standard scaler was used in both cases to standardize features by removing the mean and scaling to unit variance. The difference is that in the first case during the test stage, when the base classifiers are already trained, the test data is standardized based on the statistics of the real train data that was used in the training stage (the data from the subject that was held-out as part of the jackknife is not considered); while in the second case, the statistics of the data pooled from all subjects except the test subject is used. The results for these two cases of standardization for both the pooled-data CNNs and DASJ-CNN based ensembles are shown in Table 7.1. Here the MS-En-CNN ensemble is formed considering the majority vote (see Eq.4.2 for hard voting). The results indicate that there is barely no difference in the two cases of standardization, therefore the second case is used in the next experiments.

Another thing that was tested is the effect of using soft voting to ensemble the base classifiers. The results for three datasets are presented in Table 7.2, where the SI accuracies achieved by each of the subjects and average accuracy across all subjects with its standard deviation are shown for each classification method (pooled-data CNN versus DASJ-CNN based ensemble). The results using the multi-subject based ensemble approach (MS-En) demonstrate the accuracies achieved by combining decisions from multiple CNN models (ShallowConvNet, DeepConvNet or EEGNet) that were trained on different sets of training data via soft voting (see Eq.6.2).

Although the same classification methods were used, a considerably different performance is observed on different datasets. This might be attributed to the technical variability associated with the datasets. Across all datasets, the best performance was demonstrated by DeepConvNet (in both pooled-data CNN and DASJ-CNN based ensemble scenarios). The difference in terms of using a particular CNN architecture for pooled-data CNN and DASJ-CNN is minimal in the case of ALS dataset, while the most significant difference is observed in the case of EPFL dataset.

Returning to the primary objective of this study, which is to showcase the effectiveness of the proposed ensemble-based approach, the average SI classification results across all datasets highlight the improvement achieved by the MS-En-CNN when compared to a single model. Table 7.3 highlights the accuracy improvements achieved for each test subject by using the DASJ-ShallowConvNet, DASJ-DeepConvNet and DASJ-EEGNet based MS-En-CNN ensembles over their respective single models. A “+" and “-" sign in the table is used to indicate performance enhancement and deterioration when utilizing DASJ-CNN based ensemble instead of the single CNN architecture. The improvements of more than 2% are presented in bold. As it can be noticed there is a significant difference in the improvements across different datasets. Moreover, even within the same dataset, the achieved performance varies across subjects. The variability in performance can be attributed to subject-several factors. EEG signals can vary significantly between individuals due to differences in brain anatomy, electrode

Table 7.1: SI classification accuracy using a single model of particular architecture (ShallowConvNet, DeepConvNet or EEGNet), *i.e.* pooled-data CNN, versus using ensemble formed based on DASJ sample with the base classifiers being one of the three architectures (ShallowConvNet, DeepConvNet or EEGNet). The results are presented for two cases of standardization of test data using the statistics of: 1) real train data; 2) data pooled from all subjects except the test subject. The hard voting combining rule is used to form the MS-En-CNN.

(a) BNCI dataset

Pooled-data CNN						
	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	1	**2**	**1**	**2**	**1**	**2**
1	72.63	72.76	71.34	71.37	71.78	71.81
2	76.64	76.67	82.32	82.17	79.04	78.98
3	79.58	79.45	78.35	78.06	76.74	76.74
4	80.65	80.65	80.52	80.52	78.98	78.98
5	80.71	80.59	85.20	85.13	82.80	82.80
6	71.91	71.65	72.47	72.29	72.13	71.88
7	76.01	76.04	66.70	66.22	71.43	71.37
8	71.18	70.96	74.62	74.59	73.48	73.52
9	83.43	83.49	85.23	85.23	85.13	85.01
10	84.44	84.28	86.77	86.68	76.01	75.88
Avg	**77.72**	**77.65**	**78.35**	**78.23**	**76.75**	**76.70**
STD	**4.78**	**4.80**	**6.82**	**6.90**	**4.74**	**4.75**

DASJ sample based MS-En-CNN						
	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	1	**2**	**1**	**2**	**1**	**2**
1	70.45	70.52	73.26	73.74	73.20	72.85
2	81.98	81.88	84.97	85.54	82.45	82.42
3	78.35	77.94	78.06	77.94	77.56	77.49
4	77.59	77.46	80.93	80.56	80.05	80.27
5	84.56	84.60	85.64	85.51	85.04	85.04
6	73.80	73.20	78.76	79.48	77.81	77.78
7	70.71	70.58	71.72	71.65	74.37	74.78
8	73.07	72.89	75.38	74.78	74.21	74.27
9	87.53	87.56	88.57	88.13	87.85	87.78
10	85.86	85.92	84.28	84.41	82.51	82.83
Avg	**78.39**	**78.25**	**80.16**	**80.17**	**79.50**	**79.55**
STD	**6.34**	**6.43**	**5.67**	**5.65**	**4.93**	**4.95**

Table 7.1: Continued.

(b) ALS dataset

Pooled-data CNN						
	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	1	**2**	**1**	**2**	**1**	**2**
1	70.25	70.21	70.53	70.78	69.81	69.78
2	66.96	66.97	65.53	65.43	67.42	67.36
3	74.29	74.18	76.36	76.23	74.47	74.51
4	69.57	69.57	69.12	69.08	68.83	68.86
5	73.30	73.32	70.96	70.96	70.53	70.56
6	72.14	72.14	69.32	69.44	69.66	69.73
7	71.49	71.53	71.25	71.30	70.73	70.65
8	72.47	72.39	77.43	77.30	75.42	75.45
Avg	**71.31**	**71.29**	**71.31**	**71.31**	**70.86**	**70.86**
STD	**2.33**	**2.31**	**3.89**	**3.85**	**2.74**	**2.76**
DASJ sample based MS-En-CNN						
	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	1	**2**	**1**	**2**	**1**	**2**
1	71.13	71.16	72.01	72.16	71.25	71.32
2	68.26	68.10	68.16	68.08	67.53	67.73
3	75.00	75.18	75.88	75.82	75.86	75.65
4	68.91	68.39	69.57	69.70	69.27	69.12
5	70.45	70.70	70.69	70.65	71.84	71.78
6	72.39	72.05	72.90	72.95	70.95	70.97
7	71.69	71.97	71.29	71.64	71.68	71.82
8	77.56	77.62	75.71	75.77	76.36	76.31
Avg	**71.92**	**71.90**	**72.03**	**72.09**	**71.84**	**71.84**
STD	**3.09**	**3.21**	**2.74**	**2.73**	**3.00**	**2.92**

(c) EPFL dataset

	ShallowConvNet		**DeepConvNet**		**EEGNet**	
Pooled-data CNN						
	1	**2**	**1**	**2**	**1**	**2**
1	52.18	51.49	59.08	59.49	57.09	57.12
2	52.35	51.76	56.28	58.16	56.12	56.42
3	56.96	56.81	66.69	65.76	61.56	60.85
4	54.46	54.55	55.40	55.81	49.45	49.53
5	59.49	59.39	63.41	64.20	63.92	63.36
6	56.84	56.65	60.74	61.10	60.66	60.85
7	55.95	55.49	66.92	66.64	63.49	63.34
8	55.01	54.84	55.25	55.67	51.25	51.71
Avg	**55.40**	**55.12**	**60.47**	**60.85**	**57.94**	**57.90**
STD	**2.46**	**2.63**	**4.80**	**4.31**	**5.45**	**5.18**

placement, and inherent neural activity patterns. This inherent variability may affect the performance of the models differently for each subject. Different subjects may exhibit

Table 7.1:Continued.

DASJ sample based MS-En-CNN						
	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	1	**2**	**1**	**2**	**1**	**2**
1	57.39	56.16	57.86	58.10	59.02	59.46
2	52.49	50.61	58.91	59.48	58.99	58.93
3	60.89	61.22	61.69	60.88	62.76	63.10
4	59.61	59.05	58.67	58.69	54.15	54.10
5	69.12	68.86	67.29	66.68	67.55	67.74
6	65.93	66.40	63.64	64.12	59.29	59.09
7	65.50	63.55	58.83	58.31	62.46	61.56
8	56.92	57.14	55.19	54.31	54.83	55.21
Avg	**60.98**	**60.37**	**60.26**	**60.07**	**59.88**	**59.90**
STD	**5.54**	**5.91**	**3.79**	**3.84**	**4.38**	**4.34**

varying degrees of responsiveness to different model architectures (ShallowConvNet, DeepConvNet, EEGNet) and ensemble configurations (MS-En-CNN). This sensitivity can come from the complexity of neural dynamics captured by each model and the adaptability of the ensemble approach to different signal characteristics. The most significant subject-wise improvements were achieved on BNCI and EPFL data. When utilizing EEGNet-based base classifiers on the BNCI dataset, there was a 2.96% improvement over a single model. On the EPFL dataset, enhancements of 5.54% and 2.54% in accuracy were recorded for ShallowConvNet and EEGNet-based base

Table 7.2: Subject-independent classification accuracy using a single model of particular architecture (ShallowConvNet, DeepConvNet or EEGNet) vs. using ensemble formed based on DASJ sample with the base classifiers being one of the three architectures (ShallowConvNet, DeepConvNet or EEGNet). The soft voting combining rule is used to form the MS-En-CNN.

(a) BNCI dataset

	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	Single	MS-Ens	Single	MS-Ens	Single	MS-Ens
1	72.76	70.20	71.37	72.66	71.81	72.79
2	76.67	82.10	82.17	84.94	78.98	82.45
3	79.45	78.25	78.06	78.00	76.74	77.65
4	80.65	77.71	80.52	80.43	78.98	80.43
5	80.59	84.63	85.13	85.13	82.80	85.51
6	71.65	73.01	72.29	79.51	71.88	78.13
7	76.04	70.90	66.22	70.77	71.37	75.00
8	70.96	73.01	74.59	75.54	73.52	73.99
9	83.49	88.01	85.23	88.51	85.01	87.94
10	84.28	86.52	86.68	84.63	75.88	82.70
Avg	**77.65**	**78.43**	**78.23**	**80.01**	**76.70**	**79.66**
STD	**4.80**	**6.61**	**6.90**	**5.85**	**4.75**	**5.03**

Table 7.2: Continued.

(b) ALS dataset

	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	Single	MS-Ens	Single	MS-Ens	Single	MS-Ens
1	70.21	71.22	70.78	72.01	69.78	71.25
2	66.97	68.19	65.43	67.92	67.36	68.00
3	74.18	74.94	76.23	76.14	74.51	75.65
4	69.57	68.23	69.08	69.94	68.86	69.06
5	73.32	71.18	70.96	70.70	70.56	72.21
6	72.14	72.77	69.44	73.18	69.73	71.16
7	71.53	71.05	71.30	70.99	70.65	71.83
8	72.39	77.39	77.30	76.03	75.45	76.32
Avg	**71.29**	**71.87**	**71.31**	**72.11**	**70.86**	**71.94**
STD	**2.31**	**3.14**	**3.85**	**2.89**	**2.76**	**2.88**

(c) EPFL dataset

	ShallowConvNet		**DeepConvNet**		**EEGNet**	
	Single	MS-Ens	Single	MS-Ens	Single	MS-Ens
1	51.49	57.22	59.49	61.45	57.12	59.10
2	51.76	51.89	58.16	60.27	56.42	59.99
3	56.81	61.37	65.76	63.54	60.85	63.85
4	54.55	59.53	55.81	58.90	49.53	54.38
5	59.39	68.73	64.20	68.76	63.36	68.69
6	56.65	65.71	61.10	63.17	60.85	60.30
7	55.49	62.26	66.64	58.50	63.34	61.95
8	54.84	58.55	55.67	55.93	51.71	55.25
Avg	**55.12**	**60.66**	**60.85**	**61.32**	**57.90**	**60.44**
STD	**2.63**	**5.18**	**4.31**	**3.92**	**5.18**	**4.59**

estimators, respectively. Examining the table, it is possible to see that among the 78 test cases spanning the three datasets, performance improvement occurs in 47 cases, accounting for 60.25% of the instances. Furthermore, out of these 47 cases with performance improvements, 26 cases exhibit an improvement of at least 2%, representing 55.3% of the performance enhancements exceeding 2%.

The performance enhancement attributed to the use of the DASJ-CNN classifier over the typical single CNN architecture is visually depicted in Fig.7.1. This figure illustrates the range (represented by rectangular bars), average (indicated by red horizontal lines), and standard deviation (illustrated by vertical red lines) of SI classification accuracies across all 78 test cases. Notably, the minimum accuracy achieved for any test subject is consistently higher with DASJ-CNN in nearly all cases, except for ShallowConvNet in the BNCI data. Furthermore, when considering the overall accuracy among all participants in a dataset, DASJ-CNN consistently outperforms the single CNN architecture trained in a standard manner.

Table 7.3: Subject-wise test accuracy improvements achieved when comparing the performance of MS-En-CNN with ShallowConvNet (SN), DeepConvNet (DN), and EEGNet (EN) based base estimators with the corresponding single model (pooled-data CNN). The entries marked in **bold** indicate the difference in accuracies of more than 2%. The bottom line of the table presents the difference in average accuracies across all subjects. ALS and EPFL datasets contain data for only 8 subjects, as a result, there are entries with '-' for subject 9 and 10.

	BNCI			**ALS**			**EPFL**		
	SN	**EN**	**DN**	**SN**	**EN**	**DN**	**SN**	**EN**	**DN**
1	-2.56	+0.98	+1.29	+1.01	+1.47	+1.23	**+5.73**	+1.98	+1.96
2	**+5.43**	**+3.47**	**+2.77**	+1.22	+0.64	**+2.49**	+0.13	**+3.57**	**+2.12**
3	-1.2	+0.91	-0.06	+0.76	+1.14	-0.09	**+4.56**	**+3.00**	-2.22
4	-2.94	+1.45	-0.09	-1.34	+0.2	+0.86	**+4.98**	**+4.85**	**+3.09**
5	**+4.04**	**+2.71**	0.00	-2.14	+1.65	-0.26	**+9.34**	**+5.33**	**+4.56**
6	+1.36	**+6.25**	+7.22	+0.63	+1.43	**+3.74**	**+9.06**	-0.55	**+2.07**
7	-5.14	**+3.63**	+4.55	-0.48	+1.18	-0.31	**+6.77**	-1.39	-8.14
8	**+2.05**	+0.47	+0.95	**+5.00**	+0.87	-1.27	**+3.71**	**+3.54**	+0.26
9	**+4.52**	**+2.93**	+3.28	-	-	-	-	-	-
10	**+2.24**	**+6.82**	-2.05	-	-	-	-	-	-
Avg	+0.78	**+2.96**	+1.79	+0.58	+1.08	+0.80	**+5.54**	**+2.54**	+0.46

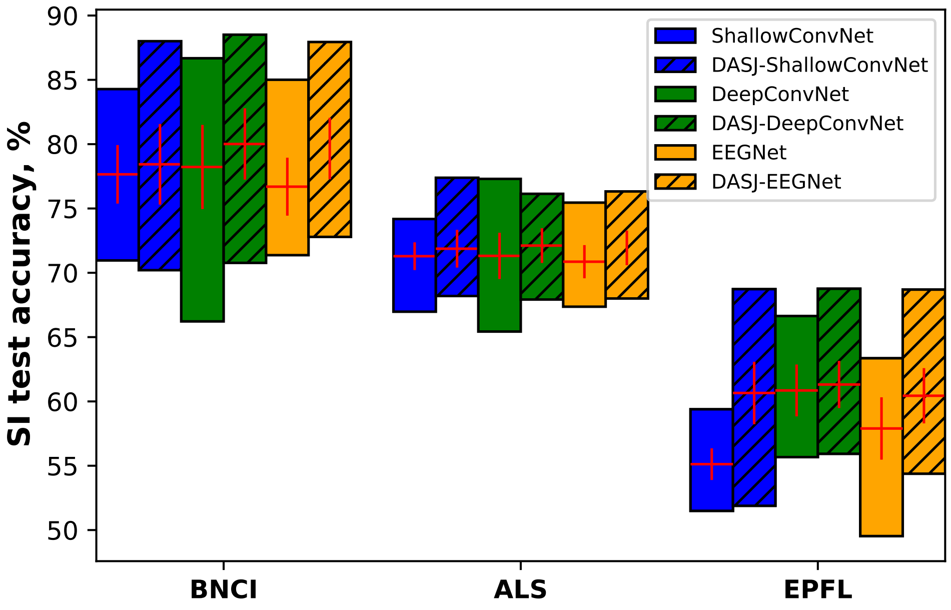

Figure 7.1: Range of accuracies attained in the assessing the performance of DASJ-CNN and individual CNN classifiers. The horizontal red lines represent the averaged SI accuracy across all subjects (BNCI dataset consists of 10 subjects, while ALS and EPFL datasets contained data for 8 subjects). Standard deviation is presented with vertical red lines.

Table 7.4: Number of trainable parameters per model across three ERP-based datasets.

	BNCI	**ALS**	**EPFL**
EEGNet	1 162	1 034	1 418
ShallowConvNet	5 352	2 952	10 152
DeepConvNet	142 552	137 552	152 552

7.2 Limitations

One limitation of the proposed strategies lies in the computational complexity associated with training the MS-En-CNN model. When dealing with a dataset containing M training subjects, DASJ approach (SS-TM and SP-TM presented in Chapter 6) demands the training of M (M and $M(M-1)$, respectively) classifiers. For a general overview of the complexity of each CNN model utilized, the number of trainable parameters per model for each dataset is presented in Table 7.4.

To address the computational challenge, a potential solution is to distribute the training process across multiple graphics processing units (GPUs) or tensor Processing Units (TPUs). For instance, the training for the above experiments was successfully executed on 4 NVIDIA® Tesla V100 GPUs. However, even with a relatively moderate M value, such as $M = 9$ (given a BNCI dataset with $N = 10$), training MS-En-CNN using the SP-TM strategy can quickly become problematic due to its computational demands. In such scenarios, even training DASJ or SS-TM may pose computational difficulties, depending on the available resources. Nevertheless, there is room for exploration of alternative strategies in the future to alleviate the computational burden. For instance, instead of employing DASJ, one could consider dividing the training subjects into K subsets, where K is significantly less than M, and train K base models, similar to a K-fold cross-validation approach.

7.3 Summary

This chapter demonstrated the potential of the DASJ sample based MS-En-CNN to improve the performance of the existing CNN architectures. For this, ShallowConvNet, DeepConvNet and EEGNet were used as archetypical. The results were verified across three ERP-based datasets. Considering the computational limitation of the DASJ-based approach (as well as previously discussed strategies of SS-TM and SP-TM), an alternative option to consider a K-fold cross-validation is suggested.

Chapter 8

MS-En-CNN using *K*-fold Cross-validation

Considering the limitations of the DASJ-based approach (as well as previously discussed SS-TM and SP-TM), the *K*-fold cross-validation is investigated as an alternative method to train the base classifiers for the MS-En-CNN. To check the potential of such an approach, one of the largest datasets [27] collected from 54 subjects, known as KU dataset, is used.

8.1 Training Strategy

One of the recent works on SI classification on KU dataset was presented by Zhang *et al.* [107], who utilized DeepConvNet architecture. This architecture is used to train the base classifiers of MS-En-CNN.

Unlike the case with brute-force search-based hyperparameter tuning (Section 5.1), here the architecture for the base classifier is fixed (i.e. DeepConvNet) and the hyperparameter tuning for the base models is focused solely on the number of training epochs (as in Chapter 7). Specifically, each base model is trained for a maximum of 200 epochs (without early stopping), and the epoch that yields the lowest validation loss is selected. Each base model was trained using the Adam optimization algorithm with decoupled weight decay, as described in [154]. The learning rate was set to 0.01, and a weight decay of 0.0005 was applied. A batch size of 16 was utilized during training. This was done to ensure a fair comparison with [107].

A general visual representation of K-fold CV-based MS-En-CNN is presented in Fig. 8.2. To design a *K*-fold CV-based ensemble, the $M = 53$ (1 out of 54 subjects is set aside for SI evaluation) of training subjects are randomly split into K folds. Each of the CNN base classifiers (DeepConvNets) is then trained on the data from a subset of subjects, consisting of $\lceil 53 - 53/K \rceil$ subjects (equivalent to subjects in $K - 1$ folds), and hyperparameter tuning is performed on the remaining $\lfloor 53/K \rfloor$ subjects (found in the remaining fold). Here, $\lceil . \rceil$ and $\lfloor . \rfloor$ represent the ceiling and floor functions, respectively. Taking into account the K-fold cross-validation process, K base DeepConvNet classifiers are trained and tuned and subsequently used to build the ensemble classifier employing a majority vote combination rule. For a given dataset, the number of trainable parameters required to train a DeepConvNet base classifier is 194 102. A flowchart explaining this methodology specifically for 54 subject-based dataset can be found in Fig.8.1.

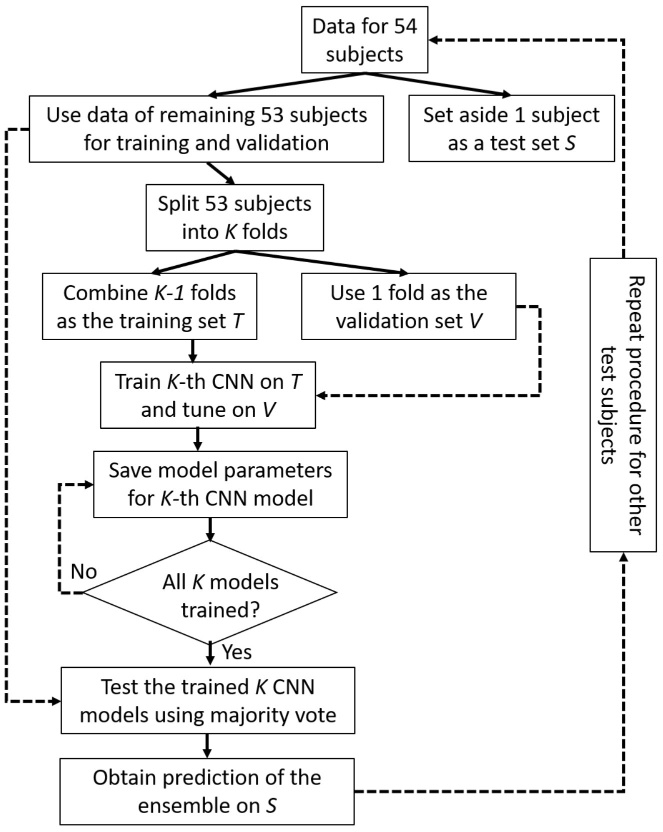
Data for 54 subjects
Use data of remaining 53 subjects for training and validation
Set aside 1 subject as a test set S
Split 53 subjects into K folds
Combine K-1 folds as the training set T
Use 1 fold as the validation set V
Train K-th CNN on T and tune on V
Save model parameters for K-th CNN model
No
All K models trained?
Yes
Test the trained K CNN models using majority vote
Obtain prediction of the ensemble on S
Repeat procedure for other test subjects

Figure 8.1: A flowchart of the training and evaluation process in a K-fold CV-based MS-En-CNN for 54 subject-based dataset.

Although generally, K could be considered as a hyperparameter to optimize, it is set to a fixed value of 13 because of computational constraints. This choice was determined based on the maximum odd number feasible given our available computational resources.

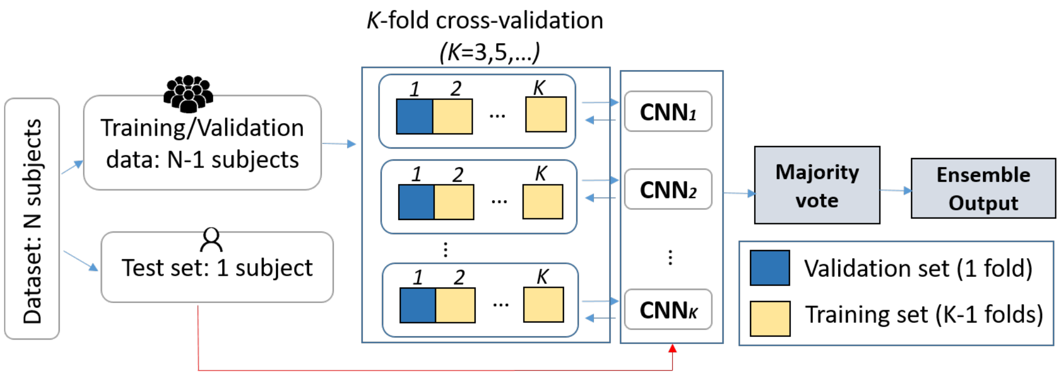

Figure 8.2: Architecture of K-fold CV based MS-En-CNN, where CNN stands for DeepConvNet base classifier.

8.2 SI Classification Performance

Table 8.1: SI performance of MS-En-CNN for each of the test subjects for different scenarios of test data split, where $s_i, i = 1, 2$ and "on"/"off" implies the session and the online/offline phases, respectively.

	s_1 **(off)**	s_1 **(on)**	s_2 **(off)**	s_2 **(on)**	s_1 **(off+on)**	s_2 **(off+on)**	s_1**+** s_2
1	90.00	83.00	81.00	89.00	86.50	85.00	85.75
2	84.00	63.00	83.00	92.00	73.50	87.50	80.50
3	94.00	91.00	97.00	97.00	92.50	97.00	94.75
4	83.00	95.00	91.00	92.00	89.00	91.50	90.25
5	98.00	83.00	92.00	93.00	90.50	92.50	91.50
6	99.00	97.00	98.00	100.00	98.00	99.00	98.50
7	73.00	87.00	83.00	87.00	80.00	85.00	82.50
8	76.00	76.00	84.00	91.00	76.00	87.50	81.75
9	86.00	79.00	76.00	67.00	82.50	71.50	77.00
10	73.00	59.00	72.00	57.00	66.00	64.50	65.25
11	83.00	69.00	75.00	65.00	76.00	70.00	73.00
12	86.00	84.00	78.00	80.00	85.00	79.00	82.00
13	79.00	78.00	75.00	67.00	78.50	71.00	74.75
14	81.00	78.00	77.00	85.00	79.50	81.00	80.25
15	94.00	92.00	93.00	91.00	93.00	92.00	92.50
16	78.00	84.00	97.00	75.00	81.00	86.00	83.50
17	75.00	83.00	69.00	80.00	79.00	74.50	76.75
18	94.00	76.00	81.00	91.00	85.00	86.00	85.50
19	81.00	90.00	76.00	90.00	85.50	83.00	84.25
20	86.00	93.00	88.00	86.00	89.50	87.00	88.25
21	95.00	97.00	99.00	98.00	96.00	98.50	97.25
22	78.00	96.00	87.00	84.00	87.00	85.50	86.25
23	86.00	58.00	89.00	67.00	72.00	78.00	75.00
24	68.00	62.00	61.00	71.00	65.00	66.00	65.50
25	87.00	59.00	100.00	98.00	73.00	99.00	86.00
26	82.00	88.00	97.00	83.00	85.00	90.00	87.50
27	93.00	85.00	97.00	93.00	89.00	95.00	92.00
28	99.00	100.00	97.00	100.00	99.50	98.50	99.00
29	78.00	85.00	82.00	85.00	81.50	83.50	82.50
30	85.00	94.00	90.00	73.00	89.50	81.50	85.50
31	92.00	95.00	91.00	83.00	93.50	87.00	90.25
32	98.00	80.00	92.00	91.00	89.00	91.50	90.25
33	98.00	100.00	98.00	99.00	99.00	98.50	98.75
34	65.00	67.00	68.00	66.00	66.00	67.00	66.50
35	100.00	100.00	95.00	100.00	100.00	97.50	98.75
36	99.00	99.00	99.00	100.00	99.00	99.50	99.25
37	96.00	93.00	97.00	96.00	94.50	96.50	95.50
38	76.00	73.00	79.00	78.00	74.50	78.50	76.50
39	92.00	94.00	87.00	98.00	93.00	92.50	92.75
40	84.00	76.00	89.00	82.00	80.00	85.50	82.75

Table 8.1: Continued.

	s_1 **(off)**	s_1 **(on)**	s_2 **(off)**	s_2 **(on)**	s_1 **(off+on)**	s_2 **(off+on)**	s_1**+** s_2
41	76.00	64.00	84.00	84.00	70.00	84.00	77.00
42	84.00	82.00	81.00	80.00	83.00	80.50	81.75
43	90.00	93.00	93.00	91.00	91.50	92.00	91.75
44	80.00	88.00	86.00	93.00	84.00	89.50	86.75
45	86.00	93.00	95.00	96.00	89.50	95.50	92.50
46	67.00	74.00	81.00	96.00	70.50	88.50	79.5
47	96.00	62.00	92.00	96.00	79.00	94.00	86.50
48	72.00	93.00	70.00	90.00	82.50	80.00	81.25
49	79.00	74.00	89.00	95.00	76.50	92.00	84.25
50	47.00	67.00	51.00	55.00	57.00	53.00	55.00
51	77.00	76.00	71.00	85.00	76.50	78.00	77.25
52	93.00	94.00	83.00	92.00	93.50	87.50	90.50
53	89.00	68.00	85.00	75.00	78.50	80.00	79.25
54	71.00	73.00	84.00	72.00	72.00	78.00	75.00
Mean	**84.28**	**82.26**	**85.28**	**85.56**	**83.27**	**85.42**	**84.34**
STD	**10.63**	**12.33**	**10.57**	**11.63**	**9.92**	**10.16**	**9.42**
Median	**84.50**	**83.50**	**86.50**	**89.50**	**83.50**	**86.50**	**84.88**
Range	**47.00**	**58.00**	**51.00**	**55.00**	**57.00**	**53.00**	**55.00**
Max	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**99.50**	**99.25**
Min	**53.00**	**42.00**	**49.00**	**45.00**	**43.00**	**46.50**	**44.25**

It's important to highlight that the entire dataset, comprising data from both phases and sessions (a total of 400 trials for each subject), excluding the subject reserved for testing, is utilized for training. However, in evaluating the ensemble classifier's performance, various scenarios for splitting the target subject's data have been explored. This was done due to the fact that in the past different methods have been employed and assessed using KU dataset. However, it's important to note that different research groups used different test data from sessions and/or phases of KU dataset in their evaluations. In order to ensure a fair comparison with the results previously reported in literature and to establish a benchmark for later comparisons, the performance of the MS-En-CNN classifier is evaluated in the following ways: a) separately for each phase (offline and online) and session (s_1 and s_2); b) for data combined from the online and offline phases while maintaining separation between sessions; c) for pooled data encompassing all phases and sessions. Table 8.1 displays the SI performance achieved by each of the test subjects in these scenarios. The mean with standard deviation and the median, as well as the minimum, maximum and range across 54 subjects are also tabulated.

8.3 Comparison with the State-of-the-Art Approaches

Recently, various algorithms have been tested on the KU dataset. Kwon *et al.* [100] presented a technique that involves combining individually trained CNN models through concatenation fusion. They demonstrated an enhanced performance compared to methods such as common spatial pattern (CSP) [97], multiresolution filter bank CSP (MR FBCSP) [97], and a fused model developed by Ray et al. [155]. For SI classification assessment using LOSO-CV, they utilized the online phase data from Session 2 and achieved an average accuracy of 74.15% with a standard deviation of 15.83%. In another recent study, a hybrid architecture called CCSPNet was proposed, which incorporates a wavelet kernel CNN, a temporal CNN, CSP, and a dense neural network. This combined architecture achieved a SI accuracy of 74.11% with a standard deviation of 15.42% when tested on the online phase data from Session 2 [114]. Jeon and colleagues [112] introduced a deep neural network aimed at learning subject-invariant and class-relevant representations. They employed DeepConvNet and EEGNet as feature extractors, achieving average accuracy scores of 73.32% with a standard deviation of 13.55% and 72.16% with a standard deviation of 13.51%, respectively, in a zero-training scenario. It is noteworthy that recent works include examples of applying transformers for SI classification. Autthasan *et al.* [156] presented MIN2Net, a method that combines multi-task learning with deep metric learning in a single framework to extract a compact and discriminative representation from EEG data for classification, yielding an accuracy of 72.03±14.04%. Deny *et al.* [157] introduced a hierarchical transformer architecture with a low-level transformer and a high-level transformer, where the former extracts features from short intervals, while the latter prioritizes relevant ones using transformer self-attention. The highest reported in the literature SI accuracy of 84.19% with a standard deviation of 10.08% was achieved by utilizing a DeepConvNet architecture. This study used pooled data from both phases of Session 2 for assessment [4] [107].

The state-of-the-art results are presented in Table 8.2. To facilitate comparison, the summary of the results for both the online phase and the combined data from the online and offline phases of Session 2 are included. MS-En-CNN demonstrates superior performance compared to previously reported results, considering both average and median accuracies. Given the substantial number of subjects and trials, the observed enhancements in accuracy compared to the runner-up performance reported in [107] are statistically significant, as confirmed by a one-sided paired Wilcoxon signed rank test with a $P = 2 \times 10^{-4}$.

To better visualize the achieved improvements, a scatter plot comparing SI classification accuracy achieved by MS-En-CNN with that of [107] for each subject

[4]as detailed in the source code available at https://github.com/zhangks98/eeg-adapt

Table 8.2: SI classification accuracy (in %) of various methods found in literature on the KU dataset. Across all methods the test data used is the data from Session 2, while its phases online (on) and offline (off) are indicated in table. A "–" sign in the table means that a measure was not reported by the authors. MS-En-CNN represents the *K*-fold CV based ensemble.

Method	**Phase**	**Min**	**Max**	**Range**	**Mean ± STD**	**Median**
MR FBCSP [100]	on	48.00	97.00	49.00	68.59±15.28	63.00
Pooled CSP [100]	on	45.00	100.00	55.00	65.65±16.11	58.00
Fused model [100]	on	41.00	98.00	57.00	67.37±16.01	62.50
CNN based fusion technique [100]	on	41.00	100.00	59.00	74.15±15.83	75.00
CCSPNet [114]	on	50.00	100.00	50.00	74.11±15.42	73.50
Jeon *et al.* [112] with EEGNet	off+on	–	–	–	72.16±13.51	–
Jeon *et al.*[112] with DeepConvNet	off+on	–	–	–	73.32±13.55	–
MIN2NET [156]	on	–	–	–	72.03±14.04	–
Hierarchical Transformer [157]	off+on	–	–	–	81.30	–
DeepConvNet [107]	off+on	52.00	99.50	47.50	84.19±10.08	84.5
MS-En-CNN	on	55.00	100.00	45.00	85.56±11.63	89.5
MS-En-CNN	off+on	53.00	99.50	46.50	85.42±10.16	86.5

when held out for testing is presented in Fig.8.3. Points located above the identity line in Fig.8.3 indicate that MS-En-CNN outperforms the previously reported method. In fact, MS-En-CNN achieved a higher SI classification accuracy for 35 out of 54 subjects in the KU dataset.

8.4 K as a Hyperparameter in MS-En-CNN

As previously mentioned, the choice of $K = 13$ was determined by the maximum odd number that was feasible given the available computational resources. However, it's crucial to recognize that K serves as a hyperparameter and naturally impacts the performance of the MS-En-CNN classifier. Therefore, in this section, the influence of varying K on the performance of MS-En-CNN is investigated.

Different values of K, specifically $K \in 3, 5, 7, 9, 11, 13$, are considered and the entire process outlined in Section 8.1 for each K (with the results for $K = 13$ already presented) is repeated. Table 8.3 presents the SI accuracy of MS-En-CNN (summary across 54 subjects) for varying values of K using pooled data from both the online and offline phases of Session 2. The accuracies achieved for each test subject could be found in Table A1 in Appendix section. An analysis to determine whether these results demonstrate a statistically significant improvement compared to the performance of

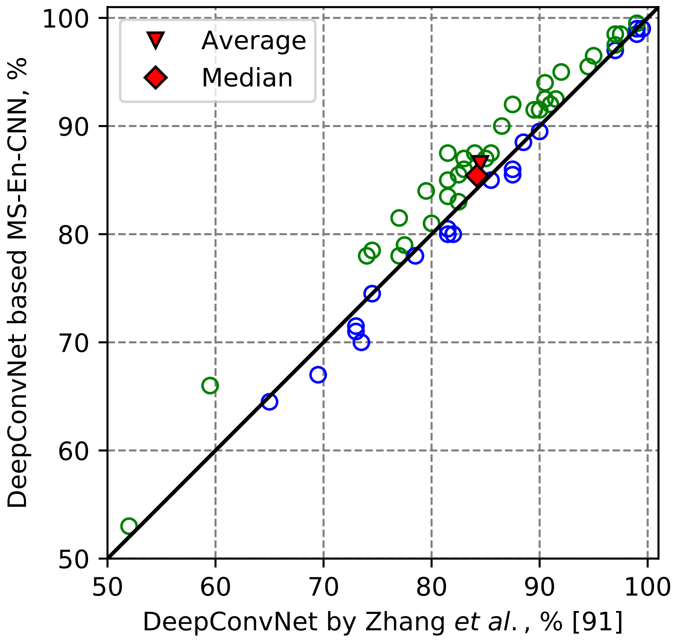

Figure 8.3: The scatter plot comparing the SI classification accuracy between MS-En-CNN and the results presented in [107] for each subject when it is reserved for testing. The vertical axis represents the accuracies achieved by the proposed MS-En-CNN algorithm with DeepConvNet base classifier, while the horizontal axis represents the accuracies reported by Zhang *et al.* in [107]. Points positioned above the diagonal line (in green) signify that MS-En-CNN performs better.

a single DeepConvNet trained in [107] was also conducted. For this purpose, the P-values from a one-sided paired Wilcoxon signed rank test are calculated and presented in Table 8.4.

Based on these results, the following observations and recommendations could be made:

- For values of K greater than or equal to 7, MS-En-CNN exhibits a statistically significant performance improvement over the single DeepConvNet trained in [107].

Table 8.3: SI performance (in %) of MS-En-CNN at different values of K using the data from Session 2 (both phases: online and offline) as a test data.

K-fold CV	Min	Max	Range	Mean ± STD	Median
$K = 3$	54.00	100.00	46.00	84.41 ±10.27	85.25
$K = 5$	53.00	99.50	46.50	84.91 ±10.12	85.50
$K = 7$	53.50	99.50	46.00	85.38 ±9.93	87.00
$K = 9$	53.50	99.50	46.00	85.30 ±9.87	87.50
$K = 11$	52.00	99.50	47.50	85.31 ±10.14	87.00
$K = 13$	53.00	99.50	46.50	85.42 ±10.16	86.50

Table 8.4: P-values computed through a one-sided Wilcoxon signed rank test to compare the classification accuracy achieved by an MS-En-CNN ensemble with base classifiers based on DeepConvNet (referred to as MS-En-DeepConvNet), trained using K-fold cross-validation, against that of a single DeepConvNet model.

K-fold CV	$K = 3$	$K = 5$	$K = 7$	$K = 9$	$K = 11$	$K = 13$
***P*-values**	0.243	0.023	9×10^{-4}	10^{-4}	0.001	2×10^{-4}

- While it might be tempting to consider larger values of K, such as 13, which could potentially yield better performance than relatively smaller values like 3, it's worth noting that moderate values like $K = 7$ or $K = 9$ demonstrate virtually comparable performance to that of $K = 13$ while being computationally less intensive (as fewer base classifiers need to be trained).

8.5 Summary

The impressive performance achieved by MS-En-DeepConvNet approach results from the utilization of a theoretically sound ensemble learning scheme in conjunction with the use of convolutional neural networks, which are known for their effectiveness in decoding intricate patterns. By harnessing the strengths of both learning approaches, it is possible to significantly outperform other methods previously attempted on a dataset of this size and complexity.

In constructing the ensemble, a fixed base CNN architecture and a single algorithmic hyperparameter, the epoch, were employed. The K base models were created with the decisions being combined using a majority voting approach. Additionally, the impact of varying K on the performance of MS-En-CNN was explored, considering values up to $K = 13$ within the limits of available computational resources. It's worth noting that in the future, depending on computing capacity, one could investigate even larger values of K (with a maximum of $N - 1$ where N represents the number of subjects in the dataset) to potentially enhance classification performance. However, as demonstrated in Section 8.4, a moderate value of K (7 or 9) already leads to a significant improvement in performance.

Chapter 9

Adaptive Boosting (AdaBoost) for EEG-based BCI

In the previous chapters, the base estimators were generated in parallel. Such a method leverages the independence among the models and typically averages out the errors. In this chapter, a different ensemble technique known as Adaptive Boosting (AdaBoost), where the base models are generated in a sequential manner, is considered.

Although AdaBoost has conventionally been associated with the basic weak classifiers, recent research has explored the effectiveness of utilizing CNNs as capable base learners within the AdaBoost framework [158]. By harnessing the robust feature extraction capabilities of CNNs and the ensemble learning prowess of AdaBoost, an innovative AdaBoost-CNN framework is investigated. This framework dynamically enhances the significance of challenging instances in the subsequent training process through iterative oversampling of the original dataset, replicating the misclassified observations.

9.1 AdaBoost-CNN with Iterative Oversampling

Typically, in AdaBoost, each base estimator ψ_m is trained while taking into account the weight vector $\mathbf{d}_m = [d_1^m, \dots, d_n^m]^T$ (see Eq.4.6 for d_i^m). These weights are assigned to individual training observations to highlight the significance of certain observations over others. This weight assignment directly impacts how the loss is computed since the model's loss function is weighted on a per-observation basis. To clarify, the weights dictate the influence of each observation within a batch when calculating the overall loss. The procedure for this boosting technique is detailed in Algorithm 5 and is commonly referred to as *boosting by (re)weighting* (in the context of this Thesis it is named as *AdaBoost.w*.)

Algorithm 5 : *AdaBoost.w*

```
1:  Set $M$, which is the number of base estimators (CNNs)
2:  for $m \in [1, M]$:
3:      if $m == 1$:
4:          Initialize the weights for each observation as $d_i^{m=1} = \frac{1}{n}$, where
            $i = \{1, 2, ..., n\}$, and $n$ is the total number of training observations:
            $\mathbf{d}_{m=1} = [\frac{1}{n}, \dots, \frac{1}{n}]$
5:          Train the CNN-based base classifier, $\psi_{m=1}$, on the training data $T_0$ using
            the initial weight vector $\mathbf{d}_{m=1}$
6:      else :
7:          Train $\psi_m$, the $m$-th CNN, on the training data $T_0$ using the weight
            vector $\mathbf{d}_m = [d_1^m, \dots, d_n^m]$
8:      Record $\mathbf{p}^m$, which represents the set of class probability estimates of
        the $m$-th CNN
9:      Use $\mathbf{p}^m$ to update the data weight vector $\mathbf{d}_{m+1}$ based on Eq. 4.6
10:     Normalize the new weight vector $\mathbf{d}_{m+1}$
```

Algorithm 6 : *AdaBoost.o*

```
1:  Set $M$, which is the number of base estimators (CNNs)
2:  for $m \in [0, M]$:
3:      if $m == 1$:
4:          Initialize the weights for each observation as $d_i^{m=1} = \frac{1}{n}$, where
            $i = \{1, 2, ..., n\}$, and $n$ is the total number of training observations:
            $\mathbf{d}_{m=1} = [\frac{1}{n}, \dots, \frac{1}{n}]$
5:          Train the CNN-based base classifier, $\psi_{m=1}$, on the training data $T_0$
6:      else :
7:          Train $\psi_m$ on the replicated training set $T_m^{**}$
8:      Record $\mathbf{p}^m$
9:      Use $\mathbf{p}^m$ to update the weight vector $\mathbf{d}_{m+1}$
10:     Use $ceil()$ function of $\mathbf{d}_m$ to determine the number of
        repetitions for each data observation
11:     Create the replicas of the observations to form a new training set $T_m^{**}$
```

In this Thesis, the *boosting by iterative oversampling* method, denoted as *AdaBoost.o*, is proposed. In this method, the weight information assigned to the observations, represented as $\mathbf{d}_m$, is utilized to increase the number of misclassified observations, essentially oversampling the training dataset. This results in a new training dataset, denoted T_m^{**}, which is subsequently used as input for the next learning iteration. The higher the weight assigned to an observation, the more attention the next base model must allocate to that particular observation, which leads to the observation being replicated more times in the new dataset T_m^{**}. The detailed procedure for implementing AdaBoost.o is shown in Algorithm 6.

While both methods (the commonly used AdaBoost.w and proposed AdaBoost.o) attribute significant importance to weights of observations, they differ in how this importance is distributed during training. To illustrate the distinctions in distributing the importance of observations across these two boosting strategies, an artificial dataset consisting of 10 observations is presented in Fig.9.1. For demonstration purposes, the weights of observations (d_i^m) are treated as integer values, although this is not the case in the actual implementation of AdaBoost.w (see line 10 for normalization in Algorithms 5), and are represented in the form of matrix D_m.

In the case of boosting by reweighting (AdaBoost.w method – Algorithm 5, Fig.9.1.a), each time the loss is computed, the complete importance of an observation is considered in the loss function for the current mini-batch. For all training instances in the mini-batch, where each training observation can appear only once, we calculate the derivative of the weighted cost function and use it to update the model weights. When the error is multiplied by the weight of the observation (as seen in the wider rectangles corresponding to higher weights in Fig.9.1.a), the error is dramatically magnified for that mini-batch. This rapid increase in error leads to quicker changes in the weights. Unlike AdaBoost.w (where observations are weighted), AdaBoost.o allows the size of the training set (T_m^{**}) after oversampling to vary based on the replication factor, indicating how many times a particular observation will be repeated.

When comparing two methods, oversampling the original training set based on weights in AdaBoost.o ensures that the effect of the weight of an observation is distributed evenly across all mini-batches, thus making the impact of observation importance global. AdaBoost.o maintains the emphasis on the importance of misclassified observations while addressing the issues of localized effects due to the high weight of an observation in a specific mini-batch (as in AdaBoost.w). By allowing certain training observations to appear in different mini-batches depending on the number of times they were replicated, the network is able to learn more smoothly.

It's worth noting that in Algorithm 6, the weight vector is not normalized, as doing so would result in entries in the normalized vector being less than 1, causing the replication factor to lose its meaning. Instead, the $ceil()$ function is calculated to determine the integer value used as the number of replications for each observation.

9.2 SI Classification Performance of AdaBoost-CNN

This section presents the results that were achieved using the AdaBoost-CNN method. The performance was evaluated on MI-based dataset. To construct the ensemble the standard two and three-layered CNN architectures were used. Both 1-dimensional (1D) and 2-dimensional (2D) CNNs were considered. For both of the boosting methods (AdaBoost.w and AdaBoost.o) the same training procedure was followed. In total 50 base estimators (CNNs) were used, and each one was trained for a maximum of 50 epochs. The batch size was set to 32. The learning rate of 0.001 with the decay

Table 9.1: Average (across 9 subjects) SI classification accuracy results achieved on MI-based dataset by the AdaBoost-CNN ensembles with various CNN configurations being used as the base estimators (BE) of the ensemble using reweighting (AdaBoost.w) and iterative oversampling based boosting (AdaBoost.o) techniques, AB.w and AB.o in the table, respectively. Here 1D convolution was considered.

(a) Two-layered CNN architectures as base estimators for AdaBoost-CNN.

BE models	**Hyperparameters of BE**			**AB.w, %**	**AB.o, %**
	conv1	**conv2**	**kernel**		
AB-CNN1D-16-32/3	16	32	3	68.83	**69.29**
AB-CNN1D-16-32/5	16	32	5	**68.94**	68.09
AB-CNN1D-16-32/7	16	32	7	**70.49**	68.94
AB-CNN1D-32-64/3	32	64	3	68.94	**68.98**
AB-CNN1D-32-64/5	32	64	5	68.33	**70.02**
AB-CNN1D-32-64/7	32	64	7	69.44	**70.68**
AB-CNN1D-64-128/3	64	128	3	69.29	**70.10**
AB-CNN1D-64-128/5	64	128	5	69.64	**69.71**
AB-CNN1D-64-128/7	64	128	7	68.79	**71.64**
Average				69.19	**69.72**

(b) Three-layered CNN architectures as base estimators for AdaBoost-CNN.

BE models	**Hyperparameters of BE**				**AB.w, %**	**AB.o, %**
	conv1	**conv2**	**conv3**	**kernel**		
AB-CNN1D-16-32-64/3	16	32	64	3	**68.48**	67.75
AB-CNN1D-16-32-64/5	16	32	64	5	68.02	**69.52**
AB-CNN1D-16-32-64/7	16	32	64	7	66.90	**69.60**
AB-CNN1D-32-64-128/3	32	64	128	3	68.75	**69.29**
AB-CNN1D-32-64-128/5	32	64	128	5	68.36	**69.95**
AB-CNN1D-32-64-128/7	32	64	128	7	68.09	**69.71**
AB-CNN1D-64-32-16/3	64	32	16	3	67.25	**71.03**
AB-CNN1D-64-32-16/5	64	32	16	5	66.13	**68.40**
AB-CNN1D-64-32-16/7	64	32	16	7	69.68	**70.87**
AB-CNN1D-128-64-32/3	128	64	32	3	69.29	**70.18**
AB-CNN1D-128-64-32/5	128	64	32	5	67.79	**68.91**
AB-CNN1D-128-64-32/7	128	64	32	7	71.84	**72.96**
Average					68.38	**69.85**

Table 9.2: Average (across 9 subjects) SI classification accuracy results achieved on MI-based dataset by the AdaBoost-CNN ensembles with various CNN configurations being used as the base estimators (BE) of the ensemble using reweighting (AdaBoost.w) and iterative oversampling based boosting (AdaBoost.o) techniques, AB.w and AB.o in the table, respectively. Here 2D convolution was considered.

(a) Two-layered CNN architectures as base estimators for AdaBoost-CNN.

BE models	**Hyperparameters of BE**			**AB.w,**	**AB.o,**
	conv1	**conv2**	**kernel**	**%**	**%**
AB-CNN2D-16-32/3	16	32	3×3	51.58	**54.17**
AB-CNN2D-16-32/5	16	32	5×5	**65.05**	63.31
AB-CNN2D-16-32/7	16	32	7×7	**63.27**	62.73
AB-CNN2D-32-64/3	32	64	3×3	51.23	**58.14**
AB-CNN2D-32-64/5	32	64	5×5	**65.97**	63.97
AB-CNN2D-32-64/7	32	64	7×7	65.12	65.12
AB-CNN2D-64-128/3	64	128	3×3	**62.31**	61.92
AB-CNN2D-64-128/7	64	128	7×7	**67.44**	66.05
Average				61.50	**61.93**

(b) Two-layered CNN architectures with rectangular shape of the kernel as base estimators for AdaBoost-CNN.

BE models	**Hyperparameters of BE**			**AB.w,**	**AB.o,**
	conv1	**conv2**	**kernel**	**%**	**%**
AB-CNN2D-16-32/3×8	16	32	3×8	64.81	**67.13**
AB-CNN2D-16-32/3×24	16	32	3×24	65.39	**66.24**
AB-CNN2D-16-32/3×40	16	32	3×40	66.78	**67.44**
AB-CNN2D-32-64/3×8	32	64	3×8	**65.55**	64.93
AB-CNN2D-32-64/3×24	32	64	3×24	67.09	**68.25**
AB-CNN2D-32-64/3×40	32	64	3×40	**66.24**	65.63
AB-CNN2D-64-128/3×8	64	128	3×8	64.62	**64.70**
AB-CNN2D-64-128/3×24	64	128	3×24	65.82	**67.55**
AB-CNN2D-64-128/3×40	64	128	3×40	**64.74**	64.24
Average				65.67	**66.23**

(c) Three-layered CNN architectures as base estimators for AdaBoost-CNN.

BE models	**Hyperparameters of BE**				**AB.w,**	**AB.o,**
	conv1	**conv2**	**conv3**	**kernel**	**%**	**%**
AB-CNN2D-16-32-64/3	16	32	64	3×3	67.40	**67.90**
AB-CNN2D-16-32-64/5	16	32	64	5×5	68.06	**69.48**
AB-CNN2D-32-64-128/3	32	64	128	3×3	**68.21**	68.02
AB-CNN2D-32-64-128/5	32	64	128	5×5	**68.44**	68.40
Average					68.03	**68.45**

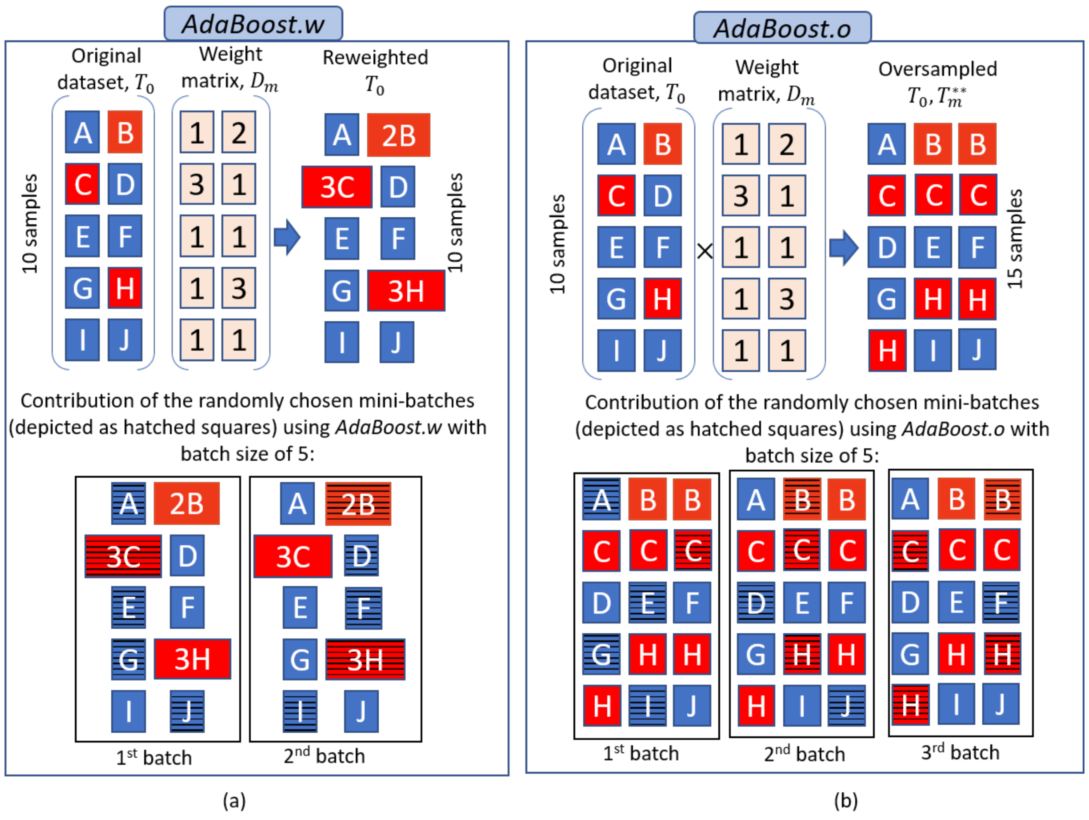
AdaBoost.w
Original dataset, T_0
Weight matrix, D_m
Reweighted T_0
10 samples
Contribution of the randomly chosen mini-batches (depicted as hatched squares) using AdaBoost.w with batch size of 5:
1st batch
2nd batch
(a)
AdaBoost.o
Oversampled T_0, T_m^{**}
15 samples
Contribution of the randomly chosen mini-batches (depicted as hatched squares) using AdaBoost.o with batch size of 5:
3rd batch
(b)

Figure 9.1: Distribution of the importance of an observation using different boosting strategies: (a) AdaBoost.w, and (b) AdaBoost.o. Blue color indicates the observations from the original dataset that were correctly classified in the m-th iteration, while the reddish color illustrates the misclassified ones, where the brighter the color, the more this observation contributed to the error (observations C and H). The lower part of each subfigure illustrates the contribution of the randomly chosen mini-batches (depicted as hatched squares) in the optimization process with a batch size of 5 (this number was selected just for illustration purposes). It should be noted that the entries in the weight matrix D_m were intentionally used as integers to provide a clearer depiction of the process, while d_i^m should be normalized as shown in line 10 of Algorithm 5.

factor of 10^{-6} was used. Tables 9.1 and 9.2 demonstrate the details about the CNN configurations: the number of convolutional filters used in each CNN layer (conv1, conv2, conv3) and the kernel size. In the case of 2D CNNs, in addition to the square shape of the kernel sizes (3×3, 5×5, 7×7) rectangular shapes (3×8, 3×24, 3×40) were considered. In the same tables, the average classification performance (across 9 subjects) is presented for each case of AdaBoost-CNN. The accuracy in bold highlights the one that on average performs better when comparing AdaBoost.w and AdaBoost.o. On average the proposed AdaBoost.o boosting technique exhibits better performance.

In evaluating the general performance of AdaBoost-CNN, it was observed that the use of 2-layered 2D CNNs with standard square-shaped kernels yielded the lowest results. However, introducing rectangular-shaped kernels led to a notable improvement, enhancing accuracy by approximately 4%. The performance was further optimized with a 3-layered architecture for 2D CNNs within the AdaBoost-CNN framework. On a given dataset, across diverse CNN configurations, the most impressive results were obtained with 1D CNNs. Among the tested 1D CNNs, AdaBoost.o achieved an average accuracy of 69.79%, surpassing AdaBoost.w by 1.06%. The classification accuracies for each test subject can be found in the Appendix Tables B2 and B3. On the given dataset with the defined architectures of the base estimators, the achieved improvement of AdaBoost.o with respect to AdaBoost.w is not substantial. Nevertheless, the overall performance of AdaBoost-CNN is remarkably impressive. Moreover, when considering the iterative oversampling method on image classification problems in most cases the improvement of more than 4% was achieved with AdaBoost.o over AdaBoost.w, while in some cases the difference in the performance was more than 12%. To showcase the potential of the proposed method, a summary of results for image classification across four widely used datasets is provided in Appendix (see Tables B4 and B5). Given the potential of the proposed boosting strategy, it is recommended for further exploration and analysis.

Chapter 10

Discussion

The emergence of deep learning technologies, powered by recent advancements in parallel computing hardware and algorithmic progress, has presented opportunities to achieve significant breakthroughs across various domains [52]. This situation is primarily attributed to the deep neural networks' (DNNs) ability to capture complex data representations through multiple hidden layers [159]. Among different types of DNNs, Convolutional Neural Networks (CNNs) have excelled in terms of training efficiency and performance, leading to several state-of-the-art architectures for accurately classifying EEG data [35, 30, 21]. Despite this, the challenge of inter-subject variability in brain signals remains a significant obstacle in developing zero-calibration subject-independent (SI) BCI systems. This PhD Thesis aims to create robust models for anticipating the user's mental intention using EEG signals. In this context, the "robustness" implies the development of SI models applicable to any user without requiring individualized calibration. Moreover, such models aim to accommodate various BCI paradigms, ensuring adaptability across a range of applications and users. Particularly in this project, several motor imagery (MI) and P300-based datasets were considered. In both cases, the project involved binary classification tasks: differentiating between left and right-hand movements in one scenario and distinguishing target (P300 signal) from non-target (non-P300 signal) signals in the other.

To push the boundaries of DNNs in the context of SI EEG classification, here the advantages of CNNs are integrated within the potential of the ensemble learning methods. However, while this integration holds promise, common ensemble learning techniques like stacking [160, 161], boosting [121, 162], bagging [163, 64], and random subspace [164] do not offer a clear framework for training and fine-tuning to enhance the performance of the existing DNN classification models, especially for use in SI applications. Furthermore, enveloping a classifier within an ensemble learning approach does not necessarily guarantee a substantial improvement in performance, if any [165, 166, 167]. This prompts the question of whether a systematic approach is viable for enhancing the performance in SI EEG classification or if "ad hoc" techniques remain the sole viable options.

In this study, a "multi-subject" ensemble CNN (MS-En-CNN) is proposed and its effectiveness is investigated. This ensemble consists of multiple CNN base classifiers, each trained on a subset of subjects' data. Three strategies for constructing the MS-En-CNN, namely Subject-Specific Training and Model Selection (SS-TM), Subject Pairs

Training and Model Selection (SP-TM), and "Delete-a-Subject-Jackknife" (DASJ), are introduced. The numerical experiments focus on evaluating the performance of these strategies in SI classification tasks. On the P300-based dataset (BNCI), the results demonstrate that all three MS-En-CNN strategies yield significantly improved SI classification performance compared to the average performance of their respective base CNN classifiers. Depending on the test subject (and the strategy), these improvements in classification accuracy range from approximately 1% to about 11%. Furthermore, the findings reveal that MS-En-CNN substantially enhances the average classification performance when compared to the conventional approach of training CNNs using pooled data from multiple subjects. The improvements of 4.19%, 5.16%, and 8.49% in average classification accuracy (across 10 subjects) are achieved when using SS-TM, SP-TM, and DASJ, respectively. Moreover, the respective reductions in the standard deviation are 1.22%, 0.73%, and 2.56%. Among the three proposed base classifier training strategies, DASJ sample based approach appears to be the most promising. This strategy was further validated on two MI-based datasets, where the performance of the MS-En-CNN was evaluated against the average accuracies of the base classifiers. Employing the DASJ based ensemble yielded 1.48% and 2.95% improvement in the case of BCI IV 2A and 2B datasets, respectively. The significance of the improvements was confirmed through a one-sided paired Wilcoxon signed-rank test with a significance level of 0.05.

The above-mentioned results were achieved while employing a brute-force search based model selection to define the architecture for each of the base classifiers of the MS-En-CNN. As it was demonstrated in [135] and verified herein in the context of SI classification using the ensembles, training and fine-tuning numerous architectures within the restrictive hyperparameter space could be a viable option to construct an accurate classifier. Considering the fact that incorporating domain knowledge into a DNN's structure is a complex task, these results on systematic CNN model selection for designing the base classifiers can function as a helpful guide for individuals in the field of deep learning who have limited expertise in the given domain. It can aid them in designing robust ensemble models for EEG classification. Performing such a model selection for base classifiers, of course, comes at the cost of computational complexity. Depending on the availability of the resources, one may explore even larger hyperparameter space. Additionally, in this work mostly the structural hyperparameters were taken into account (number of convolutional layers, number of convolutional filters in each layer, kernel size), while the algorithmic parameters (number of epochs, batch size, optimizer, learning rate, etc.) were left fixed for all of the CNN architectures. In the future, both structural and algorithmic hyperparameters could be considered during the model selection process. Moreover, instead of using the exhaustive search (as presented herein), some suboptimal search strategies could be investigated in the future.

Exploring multiple architectures for classifier development is a feasible option, but it is challenging and resource-intensive. An alternative approach for selecting the architecture for base classifiers is to tailor an existing DNN's structure. For example, EEGNet [38], a CNN architecture that was designed based on domain knowledge and has consistently demonstrated state-of-the-art performance in various EEG classification tasks. This option has been considered in this Thesis. The base classifiers for MS-En-CNN were crafted using three prominent CNN architectures within the BCI domain: ShallowNet, DeepConvNet, and EEGNet. The base classifiers were trained following the DASJ-based strategy. The empirical results derived from the experiments, conducted on three publicly accessible P300-based datasets, indicate that in most instances, the DASJ-based ensemble CNN classifier enhances the accuracy of an existing CNN architecture. The degree of improvement reaches as high as 9% in certain cases.

A limitation of the DASJ-CNN framework arises when dealing with datasets containing a large number of subjects. It becomes computationally demanding to set aside one subject at a time, train, and fine-tune a base classifier for each jackknife sample. However, depending on the available computational resources, an alternative option is to parallelize the training process. This approach is relatively straightforward since the training of each base classifier for each jackknife sample can be carried out independently of the others. In cases where computational resources are insufficient, an alternative is to divide subjects into K folds and train one base classifier using the subjects within each fold. In this scenario, K can be considered as a hyperparameter to be optimized. Such an experiment was conducted on one of the largest datasets collected from 54 subjects. This dataset, which emerged relatively recently, has garnered significant attention and has already been used to test a variety of methods, including SI approaches. Notably, the K-fold CV-based MS-En-CNN surpassed the performance of the current state-of-the-art methods reported in the literature. This research not only represents a significant contribution by achieving the highest overall classification accuracies ever reported for such an extensive MI dataset, but it also opens up the possibility of applying the MS-En-CNN classification technique to attain state-of-the-art SI classification performance in other EEG-based paradigms in the future.

As a recommendation based on the presented work and results, the choice of ensemble learning strategy may depend on the size of the dataset. For smaller datasets, the DASJ approach, which has shown promising results in enhancing classification performance when dealing with limited subject samples, is suggested. Conversely, given larger datasets, a K-fold CV-based MS-En-CNN approach is recommended. This strategy allows for the efficient utilization of data while providing robust performance assessments.

MS-En-CNN discussed above is an example of the ensemble where the base classifiers can be trained in parallel and later aggregated via certain combining schemes. A different method to construct an ensemble is to train the base classifiers in sequence, allowing the base models to learn from the errors made by their predecessors. A well-known example of this type of ensemble is AdaBoost. Given the primary focus of this Thesis on the advantages of CNNs and their application in creating ensembles to enhance performance, as well as the emerging capability to construct AdaBoost not only from weak classifiers but also from CNNs [158], the evaluation of AdaBoost-CNN within the context of EEG classification is conducted. In addition to the conventional implementation of the boosting strategy via reweighting, an alternative strategy of iterative oversampling is proposed. On average, the iterative oversampling-based AdaBoost-CNN tends to perform better. Based on the results and promising potential of the AdaBoost-CNN, it is recommended for future investigation.

Chapter 11

Conclusions and Future work

In recent years, deep neural networks have opened up remarkable opportunities for crafting highly accurate classifiers to decode EEG signals. Among these architectures, the Convolutional Neural Network (CNN) has gained popularity due to its exceptional predictive performance and efficient training. Its efficiency in extracting crucial features from raw EEG data has solidified itself as a prominent choice for BCI applications. In this study, by leveraging the advantages of both deep CNNs and ensemble learning, a "multi-subject" ensemble CNN (MS-En-CNN) is proposed and its effectiveness is investigated. This ensemble consists of multiple CNN base classifiers, each trained on a subset of subjects' data. Several distinct strategies for constructing the MS-En-CNN, depending on how subject-specific data is utilized for training and fine-tuning the base classifiers, are introduced. Depending on the strategy, the training and validation sets were formed based on the portions of subject-specific data; pairs of the subjects; or K-fold cross-validation. The validation sets were used for the model selection process to tune the hyperparameters of the base CNNs. The proposed strategies were validated on multiple MI and P300-based datasets. On average, all of the tested strategies led to a better SI classification accuracy when compared to the average accuracy of the base CNNs used within the ensembles and with respect to a single model trained on a pool of data collected from multiple subjects. While training the base classifier according to the particular strategy, two options for defining the architecture of the base estimators were investigated. One is based on the brute force search within the limited hyperparameters space and the second with the fixed structure of well-known architectures (ShallowNet, DeepConvNet, and EEGNet). The former, although being more computationally demanding, is useful for practitioners with limited domain knowledge. The latter highlights the fact the proposed methodology is a simple yet effective way to improve the performance of existing CNN architectures.

Among different strategies, a K-fold cross-validation based MS-En-CNN showed the most promising results. In this strategy, K could be treated as a hyperparameter that might be varied depending on the size of the dataset and the availability of computational resources. The case with K equal to the number of available training subjects corresponds to the jackknife-inspired strategy, where one of the training subjects is left out to form a jackknife sample. While this approach demonstrated efficiency on small datasets, scaling up to larger datasets may pose computational challenges when setting aside a single subject. Hence, a moderately sized value of

K is suggested to be used, striking a balance between achieving better results and avoiding excessive computational complexity. The experiments conducted on one of the largest MI dataset demonstrated that K-fold based implementation of MS-En-CNN consistently enhances classification accuracy with respect to a single model. Notably, it surpassed the performance of the current state-of-the-art methods tested on the same dataset. In addition to proposing MS-En-CNN, the potential of the AdaBoost-CNN was investigated and a new boosting strategy was introduced. Given the promising results, it calls for further investigations.

While the current research has concentrated on binary classification within the SI classification of EEG, the exploration of multi-task classification could be considered in the future. Shifting towards the simultaneous classification of multiple tasks not only broadens the scope of the study, but also allows to move towards the realm of online BCI. This transition reflects the complexity of real-world cognitive scenarios, where individuals often engage in various mental activities concurrently. Talking about MI-based BCI, in addition to left and right-hand movement, imagination of other activities might be considered (for example tongue or feet movement). Binary P300 classification problem can be extended toward the character recognition problem, such that the online BCI speller can be implemented. Furthermore, extending the application of CNN ensembles to other EEG-based paradigms opens doors to diverse neurological contexts. Additionally, a critical aspect of future work might involve delving into the optimization of model selection processes. Exploring and refining search strategies to identify optimal model configurations can significantly enhance the efficiency and efficacy of the proposed methods. This is related to both MS-En-CNN and AdaBoost-CNN presented herein.

Bibliography

[1] Sarah N Abdulkader, Ayman Atia, and Mostafa-Sami M Mostafa. Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, 16(2):213–230, 2015.

[2] Anirudh Vallabhaneni, Tao Wang, and Bin He. Brain—computer interface. In *Neural engineering*, pages 85–121. Springer, 2005.

[3] Fazla Rabbi Mashrur, Khandoker Mahmudur Rahman, Mohammad Tohidul Islam Miya, Ravi Vaidyanathan, Syed Ferhat Anwar, Farhana Sarker, and Khondaker A Mamun. Bci-based consumers' choice prediction from eeg signals: An intelligent neuromarketing framework. *Frontiers in human neuroscience*, 16:861270, 2022.

[4] Fazla Rabbi Mashrur, Khandoker Mahmudur Rahman, Mohammad Tohidul Islam Miya, Ravi Vaidyanathan, Syed Ferhat Anwar, Farhana Sarker, and Khondaker A Mamun. An intelligent neuromarketing system for predicting consumers' future choice from electroencephalography signals. *Physiology & Behavior*, 253:113847, 2022.

[5] Farhan Ishtiaque, Fazla Rabbi Mashrur, Mohammad Touhidul Islam Miya, Khandoker Mahmudur Rahman, Ravi Vaidyanathan, Syed Ferhat Anwar, Farhana Sarker, and Khondaker A Mamun. Bci-based consumers' preference prediction using single channel commercial eeg device. In *2022 25th International Conference on Computer and Information Technology (ICCIT)*, pages 43–48. IEEE, 2022.

[6] Mohammed Serrhini and Abdelamjid Dargham. Toward incorporating bio-signals in online education case of assessing student attention with bci. In *Europe and MENA cooperation advances in information and communication technologies*, pages 135–146. Springer, 2017.

[7] Taciana Saad Rached and Angelo Perkusich. Emotion recognition based on brain-computer interface systems. *Brain-computer interface systems-Recent progress and future prospects*, pages 253–270, 2013.

[8] Tao Xu, Yun Zhou, Zi Wang, and Yixin Peng. Learning emotions eeg-based recognition and brain activity: A survey study on bci for intelligent tutoring system. *Procedia computer science*, 130:376–382, 2018.

[9] David Marshall, Damien Coyle, Shane Wilson, and Michael Callaghan. Games, gameplay, and bci: the state of the art. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(2):82–99, 2013.

[10] Brent J Lance, Jon Touryan, Yu-Kai Wang, Shao-Wei Lu, Chun-Hsiang Chuang, Peter Khooshabeh, Paul Sajda, Amar Marathe, Tzyy-Ping Jung, Chin-Teng Lin, et al. Towards serious games for improved bci. In *Handbook of Digital Games and Entertainment Technologies*, pages 1–28. Springer Singapore, 2015.

[11] Cleiton Pons Ferreira, Carina Soledad González-González, and Diana Francisca Adamatti. Business simulation games analysis supported by human-computer interfaces: A systematic review. *Sensors*, 21(14):4810, 2021.

[12] Tabassum Hossain, Amit Konar, et al. Brain-computer interface based user authentication system for personal device security. In *2020 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, pages 1–6. IEEE, 2020.

[13] Moonwon Yu, Netiwit Kaongoen, and Sungho Jo. P300-bci-based authentication system. In *2016 4th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–4. IEEE, 2016.

[14] Avirath Sundaresan, Brian Penchina, Sean Cheong, Victoria Grace, Antoni Valero-Cabré, and Adrien Martel. Evaluating deep learning eeg-based mental stress classification in adolescents with autism for breathing entrainment bci. *Brain Informatics*, 8(1):1–12, 2021.

[15] Christopher Wegemer. Brain-computer interfaces and education: the state of technology and imperatives for the future. *International Journal of Learning Technology*, 14(2):141–161, 2019.

[16] Mijail D Serruya, Nicholas G Hatsopoulos, Liam Paninski, Matthew R Fellows, and John P Donoghue. Instant neural control of a movement signal. *Nature*, 416(6877):141–142, 2002.

[17] Chethan Pandarinath, Paul Nuyujukian, Christine H Blabe, Brittany L Sorice, Jad Saab, Francis R Willett, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. High performance communication by people with paralysis using an intracortical brain-computer interface. *Elife*, 6:e18554, 2017.

[18] N Jeremy Hill, Thomas Navin Lal, M Schroder, Thilo Hinterberger, Barbara Wilhelm, Femke Nijboer, Ursula Mochty, Guido Widman, Christian Elger, Bernhard Scholkopf, et al. Classifying eeg and ecog signals without subject training for fast bci implementation: comparison of nonparalyzed and completely

paralyzed subjects. *IEEE transactions on neural systems and rehabilitation engineering*, 14(2):183–186, 2006.

[19] Ayman Elghrabawy and Manal Abdel Wahed. Prediction of five-class finger flexion using ecog signals. In *2012 Cairo International Biomedical Engineering Conference (CIBEC)*, pages 1–5. IEEE, 2012.

[20] Wei Wang, Alan D Degenhart, Gustavo P Sudre, Dean A Pomerleau, and Elizabeth C Tyler-Kabara. Decoding semantic information from human electrocorticographic (ecog) signals. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6294–6298. IEEE, 2011.

[21] Xiang Zhang, Lina Yao, Xianzhi Wang, Jessica Monaghan, David Mcalpine, and Yu Zhang. A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv preprint arXiv:1905.04149*, 66, 2019.

[22] Hans Berger. Über das elektroenkephalogramm des menschen. *Archiv für psychiatrie und nervenkrankheiten*, 87(1):527–570, 1929.

[23] Bamidele Osalusi, Amole Abraham, and David Aborisade. Eeg classification in brain computer interface (bci): A pragmatic appraisal. *American Journal of Biomedical Engineering*, 8(1):1–11, 2018.

[24] Shabo Annerud Awrohum. Psychedelic oscillations: A systematic review of the electrophysiological correlates of classic psychedelics. 2021.

[25] G.H. Klem, H.O. Lüders, H.H. Jasper, and C. Elger. The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology. *Recommendations for the Practice of Clinical Neurophysiology, Elsevier Publishing Company*, 1999.

[26] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.

[27] Min-Ho Lee, O-Yeon Kwon, Yong-Jeong Kim, Hong-Kyung Kim, Young-Eun Lee, John Williamson, Siamac Fazli, and Seong-Whan Lee. Eeg dataset and openbmi toolbox for three bci paradigms: An investigation into bci illiteracy. *GigaScience*, 8(5):giz002, 2019.

[28] Gert Pfurtscheller and Christa Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[30] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16(5):051001, 2019.

[31] Mahsa Pourhosein Kalashami, Mir Mohsen Pedram, and Hossein Sadr. Eeg feature extraction and data augmentation in emotion recognition. *Computational Intelligence and Neuroscience*, 2022, 2022.

[32] Swati Aggarwal and Nupur Chugh. Signal processing techniques for motor imagery brain computer interface: A review. *Array*, 1:100003, 2019.

[33] Hubert Cecotti and Axel Graser. Convolutional neural networks for p300 detection with application to brain-computer interfaces. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):433–445, 2010.

[34] Siavash Sakhavi, Cuntai Guan, and Shuicheng Yan. Parallel convolutional-linear neural network for motor imagery classification. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 2736–2740. IEEE, 2015.

[35] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017.

[36] Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, and Yuanqing Li. Deep learning based on batch normalization for p300 signal detection. *Neurocomputing*, 275:288–297, 2018.

[37] Raviraj Joshi, Purvi Goel, Mriganka Sur, and Hema A Murthy. Single trial p300 classification using convolutional lstm and deep learning ensembles method. In *Intelligent Human Computer Interaction: 10th International Conference, IHCI 2018, Allahabad, India, December 7–9, 2018, Proceedings 10*, pages 3–15. Springer, 2018.

[38] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018.

[39] Guanghai Dai, Jun Zhou, Jiahui Huang, and Ning Wang. Hs-cnn: a cnn with hybrid convolution scale for eeg motor imagery classification. *Journal of neural engineering*, 17(1):016025, 2020.

[40] Kai Keng Ang, Zheng Yang Chin, Haihong Zhang, and Cuntai Guan. Filter bank common spatial pattern (fbcsp) in brain-computer interface. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 2390–2397. IEEE, 2008.

[41] Zheng Yang Chin, Kai Keng Ang, Chuanchu Wang, Cuntai Guan, and Haihong Zhang. Multi-class filter bank common spatial pattern for four-class motor imagery bci. In *2009 annual international conference of the IEEE engineering in medicine and biology society*, pages 571–574. IEEE, 2009.

[42] Kai Keng Ang, Zheng Yang Chin, Chuanchu Wang, Cuntai Guan, and Haihong Zhang. Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b. *Frontiers in neuroscience*, 6:39, 2012.

[43] Herbert Ramoser, Johannes Muller-Gerking, and Gert Pfurtscheller. Optimal spatial filtering of single trial eeg during imagined hand movement. *IEEE transactions on rehabilitation engineering*, 8(4):441–446, 2000.

[44] Siavash Sakhavi, Cuntai Guan, and Shuicheng Yan. Learning temporal information for brain-computer interface using convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5619–5629, 2018.

[45] Xuyang Zhu, Peiyang Li, Cunbo Li, Dezhong Yao, Rui Zhang, and Peng Xu. Separated channel convolutional neural network to realize the training free motor imagery bci systems. *Biomedical Signal Processing and Control*, 49:396–403, 2019.

[46] Benjamin Blankertz, K-R Muller, Dean J Krusienski, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Gert Pfurtscheller, Jd R Millan, Michael Schroder, and Niels Birbaumer. The bci competition iii: Validating alternative approaches to actual bci problems. *IEEE transactions on neural systems and rehabilitation engineering*, 14(2):153–159, 2006.

[47] Hubert Cecotti, Miguel P Eckstein, and Barry Giesbrecht. Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering. *IEEE transactions on neural networks and learning systems*, 25(11):2030–2042, 2014.

[48] Yousef Rezaei Tabar and Ugur Halici. A novel deep learning approach for classification of eeg motor imagery signals. *Journal of neural engineering*, 14(1):016003, 2016.

[49] Robert Leeb, Clemens Brunner, Gernot Müller-Putz, and Gert Pfurtscheller. BCI Competition IV data set 2b session-to-session transfer of a motor imagery bci under presence of eye artifacts. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, 2008.

[50] Müller Klaus-Robert, Blankertz Benjamin, Vidaurre Carmen, Nolte Guido, and Curio Gabriel. BCI Competition IV data set 1. *Technische Universität Berlin, Machine Learning Laboratory; Fraunhofer FIRST, Intelligent Data Analysis Group; and Campus Benjamin Franklin of the Charité - University Medicine Berlin, Department of Neurology, Neurophysics Group*, 2008.

[51] Clemens Brunner, Robert Leeb, Gernot Müller-Putz, Alois Schlögl, and Gert Pfurtscheller. BCI Competition IV data set 2a continuous multi-class motor imagery. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, 2008.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[54] Xianlun Tang, Jiwei Yang, and Hui Wan. A hybrid sae and cnn classifier for motor imagery eeg classification. In *Artificial Intelligence and Algorithms in Intelligent Systems: Proceedings of 7th Computer Science On-line Conference 2018, Volume 2 7*, pages 265–278. Springer, 2019.

[55] Weizheng Qiao and Xiaojun Bi. Deep spatial-temporal neural network for classification of eeg-based motor imagery. In *Proceedings of the 2019 international conference on artificial intelligence and computer science*, pages 265–272, 2019.

[56] Mengxi Dai, Dezhi Zheng, Rui Na, Shuai Wang, and Shuailei Zhang. Eeg classification of motor imagery using a novel deep learning framework. *Sensors*, 19(3):551, 2019.

[57] Ruilong Zhang, Qun Zong, Liqian Dou, and Xinyi Zhao. A novel hybrid deep learning scheme for four-class motor imagery classification. *Journal of neural engineering*, 16(6):066004, 2019.

[58] Thorir Mar Ingolfsson, Michael Hersche, Xiaying Wang, Nobuaki Kobayashi, Lukas Cavigelli, and Luca Benini. Eeg-tcnet: An accurate temporal convolutional network for embedded motor-imagery brain–machine interfaces. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2958–2965. IEEE, 2020.

[59] Yazeed K Musallam, Nasser I AlFassam, Ghulam Muhammad, Syed Umar Amin, Mansour Alsulaiman, Wadood Abdul, Hamdi Altaheri, Mohamed A Bencherif, and Mohammed Algabri. Electroencephalography-based motor imagery classification using temporal convolutional network fusion. *Biomedical Signal Processing and Control*, 69:102826, 2021.

[60] Ji-Seon Bang, Min-Ho Lee, Siamac Fazli, Cuntai Guan, and Seong-Whan Lee. Spatio-spectral feature representation for motor imagery classification using convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):3038–3049, 2021.

[61] John Thomas, Tomasz Maszczyk, Nishant Sinha, Tilmann Kluge, and Justin Dauwels. Deep learning-based classification for brain-computer interfaces. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 234–239. IEEE, 2017.

[62] Ramesh Maddula, Joshua Stivers, Mahta Mousavi, Sriram Ravindran, and Virginia de Sa. Deep recurrent convolutional neural networks for classifying p300 bci signals. *GBCIC*, 201:18–22, 2017.

[63] Alain Rakotomamonjy, Vincent Guigue, Gregory Mallet, and Victor Alvarado. Ensemble of svms for improving brain computer interface p300 speller performances. In *Artificial Neural Networks: Biological Inspirations–ICANN 2005: 15th International Conference, Warsaw, Poland, September 11-15, 2005. Proceedings, Part I 15*, pages 45–50. Springer, 2005.

[64] Shiliang Sun, Changshui Zhang, and Dan Zhang. An experimental evaluation of ensemble methods for eeg signal classification. *Pattern Recognition Letters*, 28(15):2157–2163, 2007.

[65] Alimed Celecia Ramos, René González Hernández, Marley Vellasco, and Pedro Vellasco. Ensemble of classifiers applied to motor imagery task classification for bci applications. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2995–3002. IEEE, 2017.

[66] Alain Rakotomamonjy and Vincent Guigue. Bci competition iii: dataset ii-ensemble of svms for bci p300 speller. *IEEE transactions on biomedical engineering*, 55(3):1147–1154, 2008.

[67] Karim Said Barsim, Wangbo Zheng, and Bin Yang. Ensemble learning to eeg-based brain computer interfaces with applications on p300-spellers. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (Smc)*, pages 631–638. IEEE, 2018.

[68] Sourav Kundu and Samit Ari. P300 detection with brain–computer interface application using pca and ensemble of weighted svms. *IETE Journal of Research*, 64(3):406–414, 2018.

[69] Rahul Kumar Chaurasiya, Narendra D Londhe, and Subhojit Ghosh. Binary de-based channel selection and weighted ensemble of svm classification for novel brain–computer interface using devanagari script-based p300 speller paradigm. *International Journal of Human–Computer Interaction*, 32(11):861–877, 2016.

[70] Mathew Salvaris and Francisco Sepulveda. Wavelets and ensemble of flds for p300 classification. In *2009 4th International IEEE/EMBS Conference on Neural Engineering*, pages 339–342. IEEE, 2009.

[71] S Fazli, C Grozea, M Danóczy, B Blankertz, KR Müller, and F Popescu. *Ensembles of temporal filters enhance classification performance for ERD-based BCI systems*. na, 2008.

[72] Siamac Fazli, Florin Popescu, Márton Danóczy, Benjamin Blankertz, Klaus-Robert Müller, and Cristian Grozea. Subject-independent mental state classification in single trials. *Neural networks*, 22(9):1305–1312, 2009.

[73] Siamac Fazli, Cristian Grozea, Márton Danóczy, Benjamin Blankertz, Florin Popescu, and Klaus-Robert Müller. Subject independent eeg-based bci decoding. *Advances in Neural Information Processing Systems*, 22, 2009.

[74] Reza Ebrahimpour, Kioumars Babakhani, and Morteza Mohammad-Noori. Eeg-based motor imagery classification using wavelet coefficients and ensemble classifiers. In *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*, pages 458–463. IEEE, 2012.

[75] Ankita Datta and Rajdeep Chatterjee. Comparative study of different ensemble compositions in eeg signal classification problem. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2*, pages 145–154. Springer, 2018.

[76] Abdulhamit Subasi, Saeed Mian Qaisar, et al. The ensemble machine learning-based classification of motor imagery tasks in brain-computer interface. *Journal of Healthcare Engineering*, 2021, 2021.

[77] Francesco Cavrini, Luigi Bianchi, Lucia Rita Quitadamo, and Giovanni Saggio. A fuzzy integral ensemble method in visual p300 brain-computer interface. *Computational intelligence and neuroscience*, 2016:49–49, 2016.

[78] Omar AlZoubi, Irena Koprinska, and Rafael A Calvo. Classification of brain-computer interface data. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 123–131, 2008.

[79] Garett D Johnson and Dean J Krusienski. Ensemble swlda classifiers for the p300 speller. In *Human-Computer Interaction. Novel Interaction Methods and Techniques: 13th International Conference, HCI International 2009, San Diego, CA, USA, July 19-24, 2009, Proceedings, Part II 13*, pages 551–557. Springer, 2009.

[80] Mehrdad Fatourechi, RK Ward, and GE Birch. A self-paced brain–computer interface system with a low false positive rate. *Journal of neural engineering*, 5(1):9, 2007.

[81] Sidath Ravindra Liyanage, Cuntai Guan, Haihong Zhang, Kai Keng Ang, JianXin Xu, and Tong Heng Lee. Dynamically weighted ensemble classification for non-stationary eeg processing. *Journal of neural engineering*, 10(3):036007, 2013.

[82] Saugat Bhattacharyya, Amit Konar, DN Tibarewala, Anwesha Khasnobish, and R Janarthanan. Performance analysis of ensemble methods for multi-class classification of motor imagery eeg signal. In *Proceedings of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC)*, pages 712–716. IEEE, 2014.

[83] Bianhong Liu and Hongwei Hao. Application of ensemble classifier in eeg-based motor imagery tasks. In *MIPPR 2007: Medical Imaging, Parallel Processing of Images, and Optimization Techniques*, volume 6789, pages 268–273. SPIE, 2007.

[84] Masoume Rahimi, Asghar Zarei, Ehsan Nazerfard, and Mohammad Hassan Moradi. Ensemble methods combination for motor imagery tasks in brain computer interface. In *2016 23rd Iranian Conference on Biomedical Engineering and 2016 1st International Iranian Conference on Biomedical Engineering (ICBME)*, pages 336–340. IEEE, 2016.

[85] Mostafa Mohammadpour, MohammadKazem Ghorbanian, and Saeed Mozaffari. Comparison of eeg signal features and ensemble learning methods for motor imagery classification. In *2016 Eighth International Conference on Information and Knowledge Technology (IKT)*, pages 288–292. IEEE, 2016.

[86] Rajdeep Chatterjee, Ankita Datta, and Debarshi Kumar Sanyal. Ensemble learning approach to motor imagery eeg signal classification. In *Machine Learning in Bio-Signal Analysis and Diagnostic Imaging*, pages 183–208. Elsevier, 2019.

[87] Md Sadiq Iqbal, Md Nasim Akhtar, AHM Shahariar Parvez, Subrato Bharati, and Prajoy Podder. Ensemble learning-based eeg feature vector analysis for brain computer interface. In *Evolutionary Computing and Mobile Sustainable Networks: Proceedings of ICECMSN 2020*, pages 957–969. Springer, 2021.

[88] Reza Boostani and Mohammad Hassan Moradi. A new approach in the bci research based on fractal dimension as feature and adaboost as classifier. *Journal of Neural Engineering*, 1(4):212, 2004.

[89] Lin Gao, Wei Cheng, Jinhua Zhang, and Jue Wang. Eeg classification for motor imagery and resting state in bci applications using multi-class adaboost extreme learning machine. *Review of scientific instruments*, 87(8), 2016.

[90] Jue Wang, Lin Gao, Haoshi Zhang, and Jin Xu. Adaboost with svm-based classifier for the classification of brain motor imagery tasks. In *Universal Access in Human-Computer Interaction. Users Diversity: 6th International Conference, UAHCI 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part II 6*, pages 629–634. Springer, 2011.

[91] Yangyang Miao, Feiyu Yin, Cili Zuo, Xingyu Wang, and Jing Jin. Improved rcsp and adaboost-based classification for motor-imagery bci. In *2019 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, pages 1–5. IEEE, 2019.

[92] Mingzhao Li and Jing Pan. An effective classification approach for eeg-based bci system. In *2011 Sixth International Conference on Image and Graphics*, pages 897–901. IEEE, 2011.

[93] Pratyusha Das, Arup Kumar Sadhu, Amit Konar, Basabdatta Sen Bhattacharya, and Atulya K Nagar. Adaptive parameterized adaboost algorithm with application in eeg motor imagery classification. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[94] Alois Schlögl, Felix Lee, Horst Bischof, and Gert Pfurtscheller. Characterization of four-class motor imagery eeg data for the bci-competition 2005. *Journal of neural engineering*, 2(4):L14, 2005.

[95] Yuanlu Zhu, Ying Li, Jinling Lu, and Pengcheng Li. Eegnet with ensemble learning to improve the cross-session classification of ssvep based bci from ear-eeg. *IEEE Access*, 9:15295–15303, 2021.

[96] Aureli Soria-Frisch. A critical review on the usage of ensembles for bci. *Towards Practical Brain-Computer Interfaces: Bridging the Gap from Research to Real-World Applications*, pages 41–65, 2013.

[97] Fabien Lotte, Cuntai Guan, and Kai Keng Ang. Comparison of designs towards a subject-independent brain-computer interface based on motor imagery. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4543–4546. IEEE, 2009.

[98] Boris Reuderink, Jason Farquhar, Mannes Poel, and Anton Nijholt. A subject-independent brain-computer interface based on smoothed, second-order baselining. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4600–4604. IEEE, 2011.

[99] Fabien Lotte. Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain–computer interfaces. *Proceedings of the IEEE*, 103(6):871–890, 2015.

[100] O-Yeon Kwon, Min-Ho Lee, Cuntai Guan, and Seong-Whan Lee. Subject-independent brain–computer interfaces based on deep convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 31(10):3839–3852, 2019.

[101] Md AM Joadder, Siuly Siuly, E Kabir, Hua Wang, and Yanchun Zhang. A new design of mental state classification for subject independent bci systems. *IRBM*, 40(5):297–305, 2019.

[102] Dalin Zhang, Lina Yao, Kaixuan Chen, and Jessica Monaghan. A convolutional recurrent attention model for subject-independent eeg signal analysis. *IEEE Signal Processing Letters*, 26(5):715–719, 2019.

[103] Seyyed Moosa Hosseini, Maysam Bavafa, and Vahid Shalchyan. An auto-adaptive approach towards subject-independent motor imagery bci. In *2019 26th National and 4th International Iranian Conference on Biomedical Engineering (ICBME)*, pages 167–171. IEEE, 2019.

[104] Parisa Ghane, Narges Zarnaghinaghsh, and Ulisses Braga-Neto. Comparison of classification algorithms towards subject-specific and subject-independent bci. In *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–6. IEEE, 2021.

[105] Dae-Hyeok Lee, Dong-Kyun Han, Sung-Jin Kim, Ji-Hoon Jeong, and Seong-Whan Lee. Subject-independent brain-computer interface for decoding high-level visual imagery tasks. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3396–3401. IEEE, 2021.

[106] Shijian Lu, Cuntai Guan, and Haihong Zhang. Subject-independent brain computer interface through boosting. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[107] Kaishuo Zhang, Neethu Robinson, Seong-Whan Lee, and Cuntai Guan. Adaptive transfer learning for eeg motor imagery classification with deep convolutional neural network. *Neural Networks*, 136:1–10, 2021.

[108] Berdakh Abibullaev, Kassymzhomart Kunanbayev, and Amin Zollanvari. Subject-independent classification of p300 event-related potentials using a small number of training subjects. *IEEE Transactions on Human-Machine Systems*, 52(5):843–854, 2022.

[109] Navid Ayoobi and Elnaz Banan Sadeghian. A subject-independent brain-computer interface framework based on supervised autoencoder. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 218–221. IEEE, 2022.

[110] Zoltan J Koles, Michael S Lazar, and Steven Z Zhou. Spatial patterns underlying population differences in the background eeg. *Brain topography*, 2(4):275–284, 1990.

[111] Abbas Salami, Javier Andreu-Perez, and Helge Gillmeister. Eeg-itnet: An explainable inception temporal convolutional network for motor imagery classification. *IEEE Access*, 10:36672–36685, 2022.

[112] Eunjin Jeon, Wonjun Ko, Jee Seok Yoon, and Heung-Il Suk. Mutual information-driven subject-invariant and class-relevant deep representation learning in bci. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[113] Hohyun Cho, Minkyu Ahn, Sangtae Ahn, Moonyoung Kwon, and Sung Chan Jun. Eeg datasets for motor imagery brain–computer interface. *GigaScience*, 6(7):gix034, 2017.

[114] Mahbod Nouri, Faraz Moradi, Hafez Ghaemi, and Ali Motie Nasrabadi. Towards real-world bci: Ccspnet, a compact subject-independent motor imagery framework. *Digital Signal Processing*, 133:103816, 2023.

[115] Jing Luo, Yaojie Wang, Shuxiang Xia, Na Lu, Xiaoyong Ren, Zhenghao Shi, and Xinhong Hei. A shallow mirror transformer for subject-independent motor imagery bci. *Computers in Biology and Medicine*, 164:107254, 2023.

[116] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[117] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[118] James Large, Jason Lines, and Anthony Bagnall. A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates. *Data mining and knowledge discovery*, 33(6):1674–1709, 2019.

[119] Stamatis Karlos, Georgios Kostopoulos, and Sotiris Kotsiantis. A soft-voting ensemble based co-training scheme using static selection for binary classification problems. *Algorithms*, 13(1):26, 2020.

[120] Hui Liu. *Unmanned driving systems for smart trains*. Elsevier, 2020.

[121] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[122] Zulhadi Zakaria and Shahrel A Suandi. Face detection using combination of neural network and adaboost. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 335–338. IEEE, 2011.

[123] Kai O Arras, Oscar Martinez Mozos, and Wolfram Burgard. Using boosted features for the detection of people in 2d range data. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 3402–3407. IEEE, 2007.

[124] Jing-Ming Guo, Yun-Fu Liu, Che-Hao Chang, and Hoang-Son Nguyen. Improved hand tracking system. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(5):693–701, 2011.

[125] Arman Zharmagambetov, Magzhan Gabidolla, and Miguel A Carreira-Perpinán. Improved multiclass adaboost for image classification: The role of tree optimization. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 424–428. IEEE, 2021.

[126] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[127] Fabio Aloise, Pietro Aricò, Francesca Schettini, Angela Riccio, Serenella Salinari, Donatella Mattia, Fabio Babiloni, and Febo Cincotti. A covert attention p300-based brain–computer interface: Geospell. *Ergonomics*, 55(5):538–551, 2012.

[128] Angela Riccio, Luca Simione, Francesca Schettini, Alessia Pizzimenti, Maurizio Inghilleri, Marta Olivetti Belardinelli, Donatella Mattia, and Febo Cincotti. Attention and p300-based bci performance in people with amyotrophic lateral sclerosis. *Frontiers in human neuroscience*, 7:732, 2013.

[129] Ulrich Hoffmann, Jean-Marc Vesin, Touradj Ebrahimi, and Karin Diserens. An efficient p300-based brain–computer interface for disabled subjects. *Journal of Neuroscience methods*, 167(1):115–125, 2008.

[130] Vinay Jayaram and Alexandre Barachant. Moabb: trustworthy algorithm benchmarking for bcis. *Journal of neural engineering*, 15(6):066011, 2018.

[131] Jason Farquhar and N Jeremy Hill. Interactions between pre-processing and classification methods for event-related-potential classification: Best-practice guidelines for brain-computer interfacing. *Neuroinformatics*, 11:175–192, 2013.

[132] Berdakh Abibullaev and Amin Zollanvari. A systematic deep learning model selection for p300-based brain–computer interfaces. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(5):2744–2756, 2021.

[133] Durk Talsma and Marty G Woldorff. Methods for the estimation and removal of artifacts and overlap in erp waveforms. *Event-related potentials: A methods handbook*, pages 115–148, 2005.

[134] Farhad Maleki, Nikesh Muthukrishnan, Katie Ovens, Caroline Reinhold, and Reza Forghani. Machine learning algorithm validation: from essentials to advanced applications and implications for regulatory certification and deployment. *Neuroimaging Clinics*, 30(4):433–445, 2020.

[135] Berdakh Abibullaev, Irina Dolzhikova, and Amin Zollanvari. A brute-force cnn model selection for accurate classification of sensorimotor rhythms in bcis. *IEEE Access*, 8:101014–101023, 2020.

[136] Amanda JC Sharkey. Combining artificial neural nets: ensemble and modular multi-net systems (perspectives in neural computing), 1999.

[137] Ali Yazdizadeh, Zachary Patterson, and Bilal Farooq. Ensemble convolutional neural networks for mode inference in smartphone travel survey. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[138] Grigory Antipov, Sid-Ahmed Berrani, and Jean-Luc Dugelay. Minimalistic CNN-based ensemble model for gender prediction from face images. *Pattern recognition letters*, 70:59–65, 2016.

[139] Yueru Chen, Yijing Yang, Wei Wang, and C-C Jay Kuo. Ensembles of feedforward-designed convolutional neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3796–3800. IEEE, 2019.

[140] Shaobo Li, Yong Yao, Jie Hu, Guokai Liu, Xuemei Yao, and Jianjun Hu. An ensemble stacked convolutional neural network model for environmental event sound recognition. *Applied Sciences*, 8(7):1152, 2018.

[141] Arthur P Dempster. Upper and lower probabilities induced by a multivalued mapping. In *Classic works of the Dempster-Shafer theory of belief functions*, pages 57–72. Springer, 2008.

[142] Hao Zhu, Dylan Forenzo, and Bin He. On the deep learning models for eeg-based brain-computer interface using motor imagery. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:2283–2291, 2022.

[143] Peter Sollich and Anders Krogh. Learning with ensembles: How overfitting can be useful. In *Advances in neural information processing systems*, pages 190–196, 1996.

[144] Maurice H Quenouille. Approximate tests of correlation in time-series 3. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 45, pages 483–484. Cambridge University Press, 1949.

[145] M H Quenouille. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956.

[146] John Tukey. Bias and confidence in not quite large samples. *Ann. Math. Statist.*, 29:614, 1958.

[147] Ahmed Alaa and Mihaela Van Der Schaar. Discriminative jackknife: Quantifying uncertainty in deep learning via higher-order influence functions. In *International Conference on Machine Learning*, pages 165–174. PMLR, 2020.

[148] Jian Kang, Qinghai Zhou, and Hanghang Tong. Jurygcn: quantifying jackknife uncertainty on graph convolutional networks. In *Proceedings of the 28th ACM*

SIGKDD Conference on Knowledge Discovery and Data Mining, pages 742–752, 2022.

[149] Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research*, 15(1):1625–1651, 2014.

[150] Indrayudh Ghosal and Giles Hooker. Boosting random forests to reduce bias; one-step boosted forest and its variance estimate. *Journal of Computational and Graphical Statistics*, 30(2):493–502, 2020.

[151] Indrayudh Ghosal, Yunzhe Zhou, and Giles Hooker. The infinitesimal jackknife and combinations of models. *arXiv preprint arXiv:2209.00147*, 2022.

[152] Phillip S Kott. The delete-a-group jackknife. *Journal of Official Statistics*, 17(4):521, 2001.

[153] Minkyu Ahn and Sung Chan Jun. Performance variation in motor imagery brain–computer interface: a brief review. *Journal of neuroscience methods*, 243:103–110, 2015.

[154] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[155] Andreas M Ray, Ranganatha Sitaram, Mohit Rana, Emanuele Pasqualotto, Korhan Buyukturkoglu, Cuntai Guan, Kai-Keng Ang, Cristián Tejos, Francisco Zamorano, Francisco Aboitiz, et al. A subject-independent pattern-based brain-computer interface. *Frontiers in behavioral neuroscience*, 9:269, 2015.

[156] Phairot Autthasan, Rattanaphon Chaisaen, Thapanun Sudhawiyangkul, Phurin Rangpong, Suktipol Kiatthaveephong, Nat Dilokthanakul, Gun Bhakdisongkhram, Huy Phan, Cuntai Guan, and Theerawit Wilaiprasitporn. Min2net: End-to-end multi-task learning for subject-independent motor imagery eeg classification. *IEEE Transactions on Biomedical Engineering*, 69(6):2105–2118, 2021.

[157] Permana Deny and Kae Won Choi. Hierarchical transformer for brain computer interface. In *2023 11th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–5. IEEE, 2023.

[158] Aboozar Taherkhani, Georgina Cosma, and T Martin McGinnity. Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404:351–366, 2020.

[159] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[160] Subhajit Chatterjee and Yung-Cheol Byun. Eeg-based emotion classification using stacking ensemble approach. *Sensors*, 22(21):8550, 2022.

[161] Asmaa Maher, Saeed Mian Qaisar, N Salankar, Feng Jiang, Ryszard Tadeusiewicz, Paweł Pławiak, Ahmed A Abd El-Latif, and Mohamed Hammad. Hybrid eeg-fnirs brain-computer interface based on the non-linear features extraction and stacking ensemble learning. *biocybernetics and biomedical engineering*, 43(2):463–475, 2023.

[162] Juan J Rodríguez and Jesus Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29(8):1049–1059, 2008.

[163] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[164] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[165] Steven Simske. *Meta-analytics: consensus approaches and system patterns for data analysis*. Morgan Kaufmann, 2019.

[166] Jan N van Rijn, Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. The online performance estimation framework: heterogeneous ensemble learning for data streams. *Machine Learning*, 107:149–176, 2018.

[167] Fu-Kwun Wang, Chang-Yi Huang, and Tadele Mamo. Ensemble model based on stacked long short-term memory model for cycle life prediction of lithium–ion batteries. *Applied Sciences*, 10(10):3549, 2020.

Appendices

Appendices

This section presents the supplementary material for the Thesis.

Appendix A

Table A1: SI performance of *K*-fold cross-validation based MS-En-CNN for each of the test subjects at different *K* values. For comparison purposes the performance of the DeepConvNet reported by Zhang *et al.* [107] is presented as well.

	[107]	$K = 3$	$K = 5$	$K = 7$	$K = 9$	$K = 11$	$K = 13$
1	81.50	82.50	83.00	85.00	83.00	85.50	85.00
2	81.50	86.50	83.50	87.50	85.50	85.00	87.50
3	97.00	98.00	97.50	97.00	97.50	97.50	97.00
4	89.50	89.50	89.50	90.00	91.00	90.50	91.50
5	90.50	88.00	89.00	92.50	92.00	92.00	92.50
6	99.00	99.00	98.50	98.50	98.50	99.00	99.00
7	85.50	82.50	85.50	87.00	88.00	86.50	85.00
8	85.50	87.00	87.00	86.50	88.00	87.00	87.50
9	73.00	70.50	73.00	72.50	73.00	73.00	71.50
10	65.00	66.50	64.50	66.00	67.50	64.00	64.50
11	73.50	70.00	75.50	74.00	72.00	71.00	70.00
12	77.50	79.50	79.50	78.00	79.00	78.00	79.00
13	73.00	74.50	71.50	73.00	73.00	72.00	71.00
14	80.00	78.50	81.00	78.00	80.50	78.00	81.00
15	87.50	88.00	91.50	92.00	90.00	91.50	92.00
16	87.50	86.50	86.00	87.50	86.50	87.00	86.00
17	74.50	72.50	71.50	73.50	73.50	76.00	74.50
18	83.00	81.50	86.50	85.00	87.50	86.00	86.00
19	82.50	85.00	84.50	81.00	83.00	82.50	83.00
20	83.00	86.00	87.00	87.00	87.50	87.50	87.00
21	97.00	99.50	99.00	99.00	98.50	99.00	98.50
22	82.50	86.50	84.00	86.50	86.00	87.00	85.50
23	77.00	78.50	76.50	76.50	78.50	80.00	78.00
24	59.50	59.00	66.00	65.00	64.00	63.00	66.00

Table A1 presents the SI classification performance achieved on 54 subject-based dataset using the *K*-fold cross-validation. The accuracy results for $K = \{3, 5, 7, 9, 11, 13\}$ are shown. Additionally, the performance of DeepConvNet [107] for each test subject is tabulated.

Table A1: Continued – Results for test subjects 42-54. The mean, STD, median, miniumum, maximum and range of the accuracies across 54 subjects are presented.

	[107]	$K = 3$	$K = 5$	$K = 7$	$K = 9$	$K = 11$	$K = 13$
25	99.50	98.50	99.50	99.50	99.50	99.00	99.00
26	86.50	86.00	84.50	88.50	87.50	87.50	90.00
27	92.00	85.00	89.50	90.50	92.00	92.50	95.00
28	97.50	98.00	98.50	98.00	97.50	98.00	98.50
29	81.50	85.00	85.00	85.00	83.50	82.50	83.50
30	77.00	83.50	80.00	83.00	82.00	83.00	81.50
31	85.00	85.50	85.50	89.00	88.50	87.50	87.00
32	90.00	92.00	92.00	91.50	91.50	92.00	91.50
33	99.00	98.50	99.00	99.00	98.50	99.00	98.50
34	69.50	65.00	64.00	67.00	67.00	66.50	67.00
35	97.00	96.50	98.00	97.50	98.50	97.50	97.50
36	99.00	100.00	99.50	99.50	99.50	99.50	99.50
37	95.00	96.50	96.50	96.50	96.00	96.00	96.50
38	74.50	72.50	78.50	75.50	78.00	80.50	78.50
39	91.50	91.50	93.00	90.50	92.50	91.50	92.50
40	87.50	84.50	81.50	84.00	82.50	84.00	85.50
41	79.50	83.50	85.50	85.00	83.00	84.50	84.00
42	81.50	79.00	81.00	79.00	80.00	80.50	80.50
43	91.00	91.50	90.00	92.00	90.50	90.50	92.00
44	90.00	89.00	89.00	89.50	89.50	90.00	89.50
45	94.50	93.50	95.00	94.00	94.00	94.50	95.50
46	88.50	90.00	87.00	89.00	88.00	87.00	88.50
47	90.50	93.50	91.50	94.50	94.00	93.00	94.00
48	81.50	79.50	79.00	80.50	78.50	81.00	80.00
49	91.00	91.00	93.50	92.50	91.50	91.00	92.00
50	52.00	54.00	53.00	53.50	53.50	52.00	53.00
51	78.50	75.00	76.00	80.00	80.00	78.00	78.00
52	84.00	85.00	91.50	87.00	87.50	91.50	87.50
53	82.00	81.50	80.00	80.00	81.50	81.50	80.00
54	74.00	78.00	77.50	81.00	76.50	77.00	78.00
Mean	**84.19**	**84.41**	**84.91**	**85.38**	**85.30**	**85.31**	**85.42**
STD	**10.08**	**10.27**	**10.12**	**9.93**	**9.87**	**10.15**	**10.16**
Median	**84.50**	**85.25**	**85.50**	**87.00**	**87.50**	**87.00**	**86.50**
Min	**52.00**	**54.00**	**53.00**	**53.50**	**53.50**	**52.00**	**53.00**
Max	**99.50**	**100.00**	**99.50**	**99.50**	**99.50**	**99.50**	**99.50**
Range	**47.50**	**46.00**	**46.50**	**46.00**	**46.00**	**47.50**	**46.50**

Appendix B

Table B2: Classification accuracy results achieved on MI-based dataset by the AdaBoost-CNN ensembles for each test subject with various CNN configurations being used as the base estimators (BE) of the ensemble using reweighting (AdaBoost.w - AB.w) and iterative oversampling based boosting (AdaBoost.o - AB.o) techniques. The average across various models is presented for each subject and each method. Results are presented for test subjects 1-5. 1D convolution is considered.

Test subject	**1**		**2**		**3**		**4**		**5**	
BE model	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**
AB-CNN1D-16-32/3	79.86	81.94	55.9	53.82	91.67	95.49	63.19	55.21	51.39	52.78
AB-CNN1D-16-32/5	73.61	81.6	52.43	50.35	92.36	90.63	67.71	72.22	51.74	51.04
AB-CNN1D-16-32/7	82.29	80.21	61.81	51.74	80.56	87.85	67.36	65.28	51.39	52.43
AB-CNN1D-32-64/3	85.76	86.46	55.56	53.47	95.14	88.19	62.85	70.83	51.74	54.86
AB-CNN1D-32-64/5	82.64	78.47	55.9	58.33	85.76	92.01	71.18	65.97	51.39	52.43
AB-CNN1D-32-64/7	81.6	85.42	56.94	55.9	87.15	88.89	61.11	76.04	52.43	50.00
AB-CNN1D-64-128/3	86.11	89.24	53.47	54.51	87.15	89.93	64.93	58.68	55.21	55.21
AB-CNN1D-64-128/5	83.68	79.86	54.17	56.94	86.11	87.15	62.5	58.33	52.08	55.21
AB-CNN1D-64-128/7	80.56	83.33	56.6	57.29	86.11	90.97	59.38	73.61	53.13	55.9
	81.79	**82.95**	**55.86**	**54.71**	**88**	**90.12**	**64.47**	**66.24**	**52.28**	**53.32**
AB-CNN1D-16-32-64/3	86.46	85.42	55.21	53.13	81.25	75.69	61.81	62.85	55.56	53.82
AB-CNN1D-16-32-64/5	79.17	86.11	51.39	54.17	80.9	79.17	61.11	57.29	55.9	55.21
AB-CNN1D-16-32-64/7	80.21	82.99	51.74	52.78	73.61	78.13	61.81	62.85	53.47	54.86
AB-CNN1D-32-64-128/3	84.38	87.15	57.99	55.21	78.47	76.04	59.38	66.32	53.82	62.15
AB-CNN1D-32-64-128/5	81.6	86.11	53.82	55.56	78.13	78.47	58.68	59.03	61.81	57.99
AB-CNN1D-32-64-128/7	84.03	83.68	50.35	52.08	78.47	74.31	61.81	63.89	58.68	54.86
AB-CNN1D-64-32-16/3	82.99	86.11	55.9	55.21	87.15	91.32	57.99	66.32	54.17	54.17
AB-CNN1D-64-32-16/5	77.08	85.76	55.56	52.43	79.51	78.13	57.99	56.94	53.47	57.99
AB-CNN1D-64-32-16/7	78.13	81.6	51.39	53.47	82.64	84.38	60.76	61.46	54.86	56.6
AB-CNN1D-128-64-32/3	86.46	86.46	57.29	53.82	78.47	81.6	60.07	58.68	60.76	57.99
AB-CNN1D-128-64-32/5	80.21	85.07	52.78	53.82	77.78	76.39	59.38	52.78	52.78	55.9
AB-CNN1D-128-64-32/7	78.13	80.9	49.65	53.13	80.56	85.42	73.61	68.75	65.28	64.93
	81.57	**84.78**	**53.59**	**53.73**	**79.75**	**79.92**	**61.2**	**61.43**	**56.71**	**57.21**

Table B2: Continued - Results for test subjects 6-9.

Test subject	**6**		**7**		**8**		**9**	
AdaBoost-CNN model	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**
AB-CNN1D-16-32/3	68.75	67.71	56.94	57.64	86.11	94.44	65.63	64.58
AB-CNN1D-16-32/5	70.14	68.06	53.82	51.39	93.75	86.46	64.93	61.11
AB-CNN1D-16-32/7	67.71	69.79	63.89	59.03	94.44	84.03	64.93	70.14
AB-CNN1D-32-64/3	68.4	62.15	52.43	57.64	85.42	85.07	63.19	62.15
AB-CNN1D-32-64/5	66.32	71.53	54.51	61.11	84.72	82.29	62.5	68.06
AB-CNN1D-32-64/7	67.01	61.81	68.4	63.89	87.5	93.4	62.85	60.76
AB-CNN1D-64-128/3	66.67	70.49	59.03	55.9	87.5	90.28	63.54	66.67
AB-CNN1D-64-128/5	68.06	70.14	66.32	67.71	85.76	89.93	68.06	62.15
AB-CNN1D-64-128/7	60.42	66.67	68.06	67.71	87.85	77.43	67.01	71.88
	67.05	**67.59**	**60.38**	**60.22**	**88.12**	**87.04**	**64.74**	**65.28**
AB-CNN1D-16-32-64/3	67.36	68.4	63.89	63.89	82.64	86.11	62.15	60.42
AB-CNN1D-16-32-64/5	62.5	71.18	66.67	67.01	87.15	86.11	67.36	69.44
AB-CNN1D-16-32-64/7	65.63	68.75	62.5	68.4	84.72	88.54	68.4	69.1
AB-CNN1D-32-64-128/3	68.4	69.79	67.01	59.72	84.72	84.03	64.58	63.19
AB-CNN1D-32-64-128/5	66.32	69.79	64.58	64.58	83.33	87.85	67.01	70.14
AB-CNN1D-32-64-128/7	66.32	69.1	60.42	70.49	88.19	87.15	64.58	71.88
AB-CNN1D-64-32-16/3	67.01	67.01	56.25	67.01	80.21	88.89	63.54	63.19
AB-CNN1D-64-32-16/5	67.71	68.75	59.72	56.25	83.33	90.97	60.76	68.4
AB-CNN1D-64-32-16/7	73.96	70.83	72.22	68.4	83.33	90.28	69.79	70.83
AB-CNN1D-128-64-32/3	67.71	68.4	63.89	69.79	79.51	86.81	69.44	68.06
AB-CNN1D-128-64-32/5	69.1	71.53	68.75	66.32	79.51	87.85	69.79	70.49
AB-CNN1D-128-64-32/7	76.74	73.96	70.83	69.1	84.72	88.54	67.01	71.88
	68.23	**69.79**	**64.73**	**65.91**	**83.45**	**87.76**	**66.2**	**68.08**

Table B3: Classification accuracy results achieved on MI-based dataset by the AdaBoost-CNN ensembles for each test subject with various CNN configurations being used as the base estimators (BE) of the ensemble using reweighting (AdaBoost.w - AB.w) and iterative oversampling based boosting (AdaBoost.o - AB.o) techniques. The average across various models is presented for each subject and each method. Results are presented for test subjects 1-5. 2D convolution is considered.

Test subject	**1**		**2**		**3**		**4**		**5**	
BE model	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**
AB-CNN2D-16-32/3	46.88	50.35	51.39	50.69	49.31	63.54	51.74	59.38	51.74	50.69
AB-CNN2D-16-32/5	72.57	60.76	49.65	48.96	88.89	82.64	58.68	60.42	51.04	48.96
AB-CNN2D-16-32/7	70.49	68.4	50.69	48.96	82.99	77.43	63.54	50.69	51.39	51.39
AB-CNN2D-32-64/3	45.83	75	50	49.65	50	71.88	51.04	56.6	50	48.26
AB-CNN2D-32-64/5	78.13	73.96	52.78	49.65	92.36	80.21	56.6	60.42	53.13	47.57
AB-CNN2D-32-64/7	76.39	72.22	49.31	51.04	82.99	71.18	58.33	68.06	50	49.65
AB-CNN2D-64-128/3	68.4	76.04	50.69	50.35	83.68	74.31	53.13	54.86	52.08	51.39
AB-CNN2D-64-128/7	77.78	76.04	49.65	48.26	85.07	87.85	63.54	55.21	52.08	51.39
	67.06	**69.1**	**50.52**	**49.7**	**76.91**	**76.13**	**57.07**	**58.2**	**51.43**	**49.91**
AB-CNN2D-16-32/3x8	77.08	73.61	47.57	47.92	84.72	94.1	63.19	68.06	49.65	50.35
AB-CNN2D-16-32/3x24	67.01	74.65	54.17	54.51	92.01	93.4	64.93	50	50.35	50.69
AB-CNN2D-16-32/3x40	79.17	77.78	53.47	53.47	92.71	94.44	63.19	59.03	50.35	50.69
AB-CNN2D-32-64/3x8	72.57	74.31	52.08	50.69	86.81	82.99	64.24	63.19	50.35	50
AB-CNN2D-32-64/3x24	78.82	80.9	54.17	52.78	91.32	95.83	60.76	53.82	51.04	50.35
AB-CNN2D-32-64/3x40	76.74	81.6	53.13	52.43	89.93	84.03	63.19	57.99	50.69	49.65
AB-CNN2D-64-128/3x8	69.1	75.69	51.39	50.35	85.07	89.24	60.07	54.17	48.26	50.69
AB-CNN2D-64-128/3x24	77.43	78.82	49.65	51.04	89.93	92.01	63.89	62.85	50.69	50.69
AB-CNN2D-64-128/3x40	75.35	84.38	51.39	58.68	88.19	60.42	60.42	59.72	50.35	51.74
	74.81	**77.97**	**51.89**	**52.43**	**88.97**	**87.38**	**62.65**	**58.76**	**50.19**	**50.54**
AB-CNN2D-16-32-64/3	77.43	75.35	51.74	47.57	78.82	86.81	66.32	73.26	51.39	51.04
AB-CNN2D-16-32-64/5	78.13	80.9	53.13	53.13	82.99	84.38	64.24	66.67	54.51	54.17
AB-CNN2D-32-64-128/3	78.82	80.9	49.65	53.47	83.68	84.03	65.97	70.83	51.04	54.17
AB-CNN2D-32-64-128/5	76.74	81.94	49.65	49.31	85.76	78.82	66.32	64.24	51.39	53.47
	77.78	**79.77**	**51.04**	**50.87**	**82.81**	**83.51**	**65.71**	**68.75**	**52.08**	**53.21**

Table B3: Continued - Results for test subjects 6-9.

Test subject	**6**		**7**		**8**		**9**	
AdaBoost-CNN model	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**	**AB.w**	**AB.o**
AB-CNN2D-16-32/3	49.31	54.17	50.35	49.31	62.15	58.68	51.39	50.69
AB-CNN2D-16-32/5	61.81	61.46	54.86	58.33	93.75	87.85	54.17	60.42
AB-CNN2D-16-32/7	61.11	59.38	57.29	56.94	73.96	91.67	57.99	59.72
AB-CNN2D-32-64/3	52.43	61.46	53.47	55.9	55.9	50.35	52.43	54.17
AB-CNN2D-32-64/5	59.03	59.03	54.17	52.78	88.89	90.28	58.68	61.81
AB-CNN2D-32-64/7	64.93	68.75	61.11	50.69	82.29	91.32	60.76	63.19
AB-CNN2D-64-128/3	59.03	56.25	54.51	56.6	88.19	79.51	51.04	57.99
AB-CNN2D-64-128/7	65.63	64.58	60.42	55.56	92.01	89.93	60.76	65.63
	59.16	**60.63**	**55.77**	**54.51**	**79.64**	**79.95**	**55.9**	**59.2**
AB-CNN2D-16-32/3x8	66.67	64.58	50.35	53.13	84.03	91.32	60.07	61.11
AB-CNN2D-16-32/3x24	61.81	65.97	53.13	51.74	81.94	89.58	63.19	65.63
AB-CNN2D-16-32/3x40	63.89	65.63	48.26	52.78	84.72	90.28	65.28	62.85
AB-CNN2D-32-64/3x8	64.24	63.19	54.17	49.65	83.33	92.01	62.15	58.33
AB-CNN2D-32-64/3x24	64.58	67.01	49.65	53.82	86.11	94.1	67.36	65.63
AB-CNN2D-32-64/3x40	63.89	60.76	48.26	53.13	84.03	87.5	66.32	63.54
AB-CNN2D-64-128/3x8	63.54	57.64	52.08	54.86	87.85	92.71	64.24	56.94
AB-CNN2D-64-128/3x24	62.15	66.32	46.53	59.38	83.68	90.28	68.4	56.6
AB-CNN2D-64-128/3x40	62.15	65.97	48.26	54.51	82.29	86.11	64.24	56.6
	63.66	**64.12**	**50.08**	**53.67**	**84.22**	**90.43**	**64.58**	**60.8**
AB-CNN2D-16-32-64/3	68.06	67.71	57.99	58.33	90.63	89.58	64.24	61.46
AB-CNN2D-16-32-64/5	65.28	68.4	59.72	65.63	88.89	88.54	65.63	63.54
AB-CNN2D-32-64-128/3	69.79	69.44	59.38	60.76	91.67	78.47	63.89	60.07
AB-CNN2D-32-64-128/5	68.75	69.1	58.68	66.32	89.93	88.89	68.75	63.54
	67.97	**68.66**	**58.94**	**62.76**	**90.28**	**86.37**	**65.63**	**62.15**

Table B4: Image classification problem: average classification accuracies and standard deviation calculated across various AdaBoost-CNNs (12 architectures for each dataset).

Dataset	**AdaBoost.w**, (%)	**AdaBoost.o**, (%)
CIFAR-10	70.37 ± 4.59	78.7 ± 4.95
EMNIST by-class	83.31 ± 2.12	85.36 ± 0.83
EMNIST by-merge	86.97 ± 2.72	89.07 ± 0.82
Fashion MNIST	89.62 ± 1.17	91.36 ± 3.01

Table B5: Image classification problem: the percentage differences in classification accuracies achieved by comparing the iterative oversampling based boosting (AdaBoost.o) method to reweighting (AdaBoost.w) methods. The '+' sign indicates that AdaBoost.o based boosting outperforms the other method.

AdaBoost-CNN model	**CIFAR-10**	**EMNIST by-class**	**EMNIST by-merge**	**Fashion MNIST**
AB-CNN-16-32/3	+6.10	+6.04	+7.54	+2.11
AB-CNN-16-32/5	+7.45	+3.15	+3.12	+1.98
AB-CNN-16-32/7	+11.14	+2.42	+1.59	-7.73
AB-CNN-32-64/3	+6.19	+1.99	+4.06	+2.98
AB-CNN-32-64/5	+6.99	+3.22	+1.65	+1.55
AB-CNN-32-64/7	+11.67	+0.77	+2.29	+4.14
AB-CNN-64-128/3	+6.23	+1.37	+1.53	+2.88
AB-CNN-64-128/5	+5.93	+1.15	-0.15	+2.60
AB-CNN-64-128/7	+12.63	+0.64	+0.54	+3.33
AB-CNN-128-256/3	+6.54	+2.34	+1.92	+2.32
AB-CNN-128-256/5	+12.67	-0.05	+1.81	+2.74
AB-CNN-128-256/7	+6.45	+1.59	-0.73	+1.97