Towards High Performance and Efficient Brain Computer Interface Character Speller: Convolutional Neural Network based Methods

Hongchang Shan

Towards High Performance and Efficient Brain Computer Interface Character Speller: Convolutional Neural Network based Methods

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus Prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 25 februari 2020
klokke 16:15 uur

door

Hongchang Shan
geboren te Heilongjiang, China
in 1989

Promotors:	Dr. Todor P. Stefanov	Universiteit Leiden
	Prof. Dr. Aske Plaat	Universiteit Leiden
Promotion Committee:	Prof. Dr. Tom Heskes	Radboud Universitiet
	Prof. Dr. Shaowei Cai	Chinese Academy of Sciences
	Prof. Dr. Holger Hoos	Universiteit Leiden
	Prof. Dr. Fons Verbeek	Universiteit Leiden
	Dr. Wojtek Kowalczyk	Universiteit Leiden

Towards High Performance and Efficient
Brain Computer Interface Character Speller:
Convolutional Neural Network based Methods
Hongchang Shan. -
Dissertation Universiteit Leiden. - With ref. - With summary in Dutch.

Copyright © 2020 by Hongchang Shan. All rights reserved. No part of this thesis may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without prior permission from the author.

This dissertation was typeset using LaTeX in Linux and version controlled using Git.

Contents

Contents **v**

List of Tables **ix**

List of Figures **xiii**

List of Abbreviations **xv**

1 Introduction **1**
1.1 Development Trends in P300-based Brain Computer Interface Systems 2
1.1.1 High Performance P300-based Brain Computer Interface Systems . . . 2
1.1.2 Efficient P300-based Brain Computer Interface Systems . . 4
1.2 Problem Statement . . . 6
1.2.1 Problem 1 . . . 7
1.2.2 Problem 2 . . . 8
1.3 Research Contributions . . . 9
1.4 Dissertation Outline . . . 13

2 Background **15**
2.1 Machine Learning . . . 15
2.2 Neural Network . . . 17
2.2.1 Neurons . . . 17
2.2.2 The Architecture of a Neural Network . . . 19
2.2.3 Learning Process of a Neural Network . . . 22
2.3 Convolutional Neural Network . . . 27
2.3.1 The Convolution Operation . . . 29
2.3.2 The Characteristics of Convolutional Neural Network . . . 29
2.3.3 The Architecture of Convolutional Neural Network . . . 30
2.4 P300-based Brain Computer Interface . . . 31

2.4.1 P300 Signal 31
2.4.2 P300 Speller 33
2.4.3 Performance Assessment of P300 Speller 34
2.5 Datasets 35

3 A Simple Convolutional Neural Network for P300 Signal Detection and Character Spelling 39
3.1 Related Work 41
3.2 Proposed Convolutional Neural Network 44
3.2.1 Input to the Network 44
3.2.2 Network Architecture 45
3.2.3 Training 47
3.3 Experimental Evaluation 48
3.3.1 Experimental Setup 48
3.3.2 Complexity 48
3.3.3 P300 Signal Detection Accuracy 49
3.3.4 Character Spelling Accuracy 50
3.3.5 Information Transfer Rate 52
3.4 Conclusions 55

4 Ensemble of Convolutional Neural Networks for P300 Signal Detection and Character Spelling 57
4.1 Proposed Network 59
4.1.1 Ensemble of Convolutional Neural Networks 59
4.1.2 Proposed OSLN and OTLN 59
4.1.3 Training 61
4.1.4 P300 Signal Detection and Character Spelling using EoCNN 61
4.2 Experimental Evaluation 62
4.2.1 Complexity 63
4.2.2 P300 Signal Detection Accuracy 63
4.2.3 Character Spelling Accuracy 64
4.2.4 Information Transfer Rate 67
4.3 Discussions 69
4.3.1 Analysis of Our Proposed OTLN and OSLN 69
4.3.2 Ablation Study on EoCNN 71
4.3.3 Exploration on the Importance of Extracting P300-related Features from Raw Signals 72
4.4 Conclusions 73

5 A Novel Sensor Selection Method based on Convolutional Neural Network for P300 Speller 75
5.1 Related Work . . . 77
5.2 Our Sensor Selection Method . . . 78
5.2.1 Spatial Learning based Elimination Selection . . . 78
5.2.2 Parameterized OSLN . . . 79
5.2.3 Ranking Function . . . 81
5.3 Experimental Evaluation . . . 82
5.3.1 Experimental Setup . . . 82
5.3.2 Experimental Results . . . 84
5.4 Discussions . . . 85
5.4.1 Configuration of E_s in SLES . . . 86
5.4.2 Exploring the Impact of the CNN Architecture on Sensor Selection . . . 88
5.5 Conclusions . . . 92

6 An Improved Ensemble of Convolutional Neural Networks for P300 Speller with a Small Number of Sensors 95
6.1 Study on EoCNN-based P300 Speller with Different Number of Sensors 97
6.1.1 Experimental Setup . . . 97
6.1.2 Experimental Results . . . 98
6.2 Our Solution Approach . . . 100
6.2.1 Parameterized Ensemble Processing . . . 100
6.2.2 Parameter Configuration for Parameterized Ensemble Processing . . . 101
6.3 Experimental Evaluation . . . 103
6.3.1 Experimental Setup . . . 103
6.3.2 Experimental Results . . . 105
6.4 Conclusions . . . 108

7 Summary and Conclusions 109

Bibliography 113

List of Publications 121

Samenvatting 123

Acknowledgments 127

Curriculum Vitae 129

List of Tables

2.1 Number of P300s/non-P300s for each dataset. 37

3.1 CCNN architecture. 43
3.2 BN3 architecture. 43
3.3 CNN-R architecture. 44
3.4 OCLNN architecture. 46
3.5 Complexity comparison of different CNNs. 49
3.6 P300 signal detection accuracy of different CNNs on Dataset II, III-A and III-B. 50
3.7 Spelling accuracy achieved by different methods on Dataset II. . . . 51
3.8 Spelling accuracy achieved by different methods on Dataset III-A. . 51
3.9 Spelling accuracy achieved by different methods on Dataset III-B. . 51
3.10 Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset II. 53
3.11 Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset III-A. 53
3.12 Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset III-B. 53
3.13 The ITR of the P300 speller based on different methods on Dataset II. 54
3.14 The ITR of the P300 speller based on different methods on Dataset III-A. 54
3.15 The ITR of the P300 speller based on different methods on Dataset III-B. 54

4.1 OSLN architecture. 60
4.2 OTLN architecture. 60
4.3 Complexity of different CNNs. 63
4.4 P300 signal detection accuracy of different CNNs on Dataset II, III-A, and III-B. 64

4.5 Spelling accuracy achieved by different methods on Dataset II. . . . 65
4.6 Spelling accuracy achieved by different methods on Dataset III-A. . 65
4.7 Spelling accuracy achieved by different methods on Dataset III-B. . 66
4.8 The ITR of the P300 speller based on different methods on Dataset II. 67
4.9 The ITR of the P300 speller based on different methods on Dataset III-A. 67
4.10 The ITR of the P300 speller based on different methods on Dataset III-B. 68
4.11 Spelling accuracy achieved by OTLN, OSLN and EoCNN on Dataset III-A. 70
4.12 Spelling accuracy achieved by EoCNN after removing a separate CNN. 71

5.1 The symbols used in Algorithm 1. 79
5.2 $OSLN_{(S)}$ architecture. 81
5.3 Methods compared with SLES. 83
5.4 Minimal number of sensors selected by different methods for Dataset II. The P300 speller is implemented using the CNN-based classifier OCLNN. 85
5.5 Minimal number of sensors selected by different methods for Dataset III-A. The P300 speller is implemented using the CNN-based classifier OCLNN. 86
5.6 Minimal number of sensors selected by different methods for Dataset III-B. The P300 speller is implemented using the CNN-based classifier OCLNN. 87
5.7 Minimal number of sensors selected by different methods for Dataset II. The P300 speller is implemented using the CNN-based classifier EoCNN. 88
5.8 Minimal number of sensors selected by different methods for Dataset III-A. The P300 speller is implemented using the CNN-based classifier EoCNN. 89
5.9 Minimal number of sensors selected by different methods for Dataset III-B. The P300 speller is implemented using the CNN-based classifier EoCNN. 90
5.10 Minimal number of sensors selected by different methods for Dataset II, The P300 speller is implemented using the SVM-based classifier ESVM [RG08]. 91
5.11 Minimal number of sensors selected by different methods for Dataset III-A, The P300 speller is implemented using the SVM-based classifier ESVM [RG08]. 92

5.12 Minimal number of sensors selected by different methods for Dataset III-B, The P300 speller is implemented using the SVM-based classifier ESVM [RG08]. 93
5.13 Minimal number of sensors selected by SLES with different E_s configurations. 94
5.14 Minimal number of sensors selected by analysing different CNNs. . 94

6.1 Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset II. 105
6.2 Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset III-A. 106
6.3 Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset III-B. 106
6.4 Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art max-ITR on Dataset II, III-A, and III-B. 107

List of Figures

1.1 Workflow of a typical BCI. 1
1.2 An example of a traditional P300-based BCI system. 5
1.3 An example of an efficient P300-based BCI system. 5

2.1 The workflow of machine learning. 17
2.2 The model of a neuron. 18
2.3 Architectural graph to model a neuron. 20
2.4 An example of a single-layer neural network. 21
2.5 An example of a multi-layer neural network with one hidden layer and one output layer. 21
2.6 An example of a cost function C with two parameters v_1 and v_2. . . 23
2.7 The analogy of using gradient descent to minimize a cost function. . 24
2.8 An example of the architecture of a CNN used for the handwritten digit recognition. 31
2.9 P300 signal. 32
2.10 P300 speller character matrix. 33
2.11 An example of a set of signal samples, where F_s is the signal sampling frequency 36

3.1 Abstraction of the raw signals in the spatial convolution layer in current CNNs. x denotes a signal sample in the input tensor. f denotes a datum in a feature map. Every column in the input tensor contains a set of C signal samples. These samples come from C sensor at a certain sampling time point. The spatial convolution operation converts each column of spatial data (receptive field) from the input tensor into an abstract datum in a feature map. 42
3.2 Input tensor for our proposed OCLNN. 45
3.3 Illustration of OCLNN for P300 signal detection. 46

4.1 Workflow of our EoCNN 59

4.2 Spelling accuracy achieved by OTLN, OTLN-3l and OTLN-6l on Dataset III-A. 70
4.3 Spelling accuracy achieved by networks in set RAW_networks and networks in set unRAW_networks on Dataset III-A. 72

5.1 Input tensor to $OSLN_{(S)}$, where $s_j \in S$. 80

6.1 Spelling accuracy of different P300 speller implementations when different number of sensors m is used to acquire EEG signals. . . . 99
6.2 max-ITR of different P300 speller implementations when different number of sensors m is used to acquire EEG signals. 99

7.1 Overview of how each chapter's contributions improve the performance and/or the efficiency of a P300 speller. 110

List of Abbreviations

ALS	Amyotrophic Lateral Sclerosis
AP	Action Potential
AUC	Area Under the Receiver Operating Characteristic
BCI	Brain Computer Interface
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
ECoG	Electrocorticography
EEG	Electroencephalography
EoCNN	Ensemble of Convolutional Neural Networks
ERD	Event Related Desynchronization
ERP	Event-Related Potential
FLD	Fisher's Linear Discriminants
ITR	Information Transfer Rate
LDA	Linear Discriminant Analysis
LFP	Local Field Potential
max-ITR	maximum ITR
MSE	Mean Squared Error
NN	Neural Network
OCLNN	One Convolution Layer Neural Network
OSLN	One Spatial Layer Network
OTLN	One Temporal Layer Network
PEoCNN	EoCNN with parameterized ensemble processing
ReLU	Rectified Linear Unit

SAE	Stacked Autoencoder
SGD	Stochastic Gradient Descent
SLES	Spatial Learning based Elimination Selection
SMAC	Sequential Model-based Algorithm Configuration
SNR	Signal to Noise Ratio
SSNR	Signal to Signal and Noise Ratio
SSVEP	Steady State Visual Evoked Potential
SVM	Support Vector Machine
SWLDA	Stepwise Linear Discriminant Analysis
II	BCI Competition II - Data set IIb
III-A	BCI Competition III - Data set II Subject A
III-B	BCI Competition III - Data set II Subject B

Chapter 1

Introduction

A Brain Computer Interface (BCI), also known as mind-machine interface, translates brain signals into computer commands, thereby building communication between the human brain and outside devices. In this way, human-beings can use only the brain to express their thoughts without any real movement. As a result, BCIs become an important communication pathway for the people who lose motor ability, such as patients with Amyotrophic Lateral Sclerosis (ALS) [SD06] or spinal-cord injury. In recent years, BCIs have also been popularly developed for healthy people, in application domains such as entertainments [GP⁺13], mental state monitoring [LTK13], virtual reality [CBJ16] as well as in IoT services [LL⁺14].

A BCI system consists of three components, as shown in Figure 1.1. The first component is the brain signal acquisition. In this component, the brain signals of a subject (person) are recorded by using a brain headset equipped with a number of sensors. The acquired brain signals are sent to the second component for brain signal processing and translation. In this component, a hardware/software platform is used to process and translate brain signals into computer commands. Then, in the last component, the translated commands, i.e., the control signals, are used to control the outside devices, e.g., a prosthesis [MPP08], a computer mouse [Spü15], a mobile phone [CCH⁺10], or a robot [BFL13].

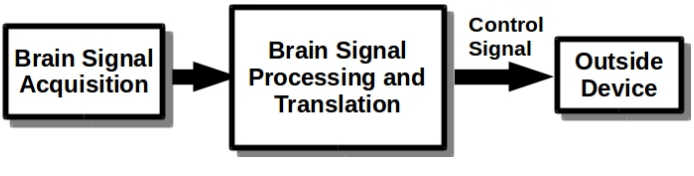


Figure 1.1: Workflow of a typical BCI.

Depending on the placement of the sensors, which are used to acquire brain signals in the brain headset, BCI systems can be categorized as invasive BCIs, semi-invasive BCIs, and non-invasive BCIs [Wal16]. In invasive BCIs, micro-sensor arrays are placed directly into the cortex [PHP10] to measure action potentials (APs) and local field potentials (LFPs). In semi-invasive BCIs, sensors are placed on the exposed surface of the brain in order to measure electrocorticography (ECoG) signals [SL11]. In non-invasive BCIs, sensors are placed on the scalp in order to acquire electroencephalography (EEG) signals [GS06]. In recent decades, EEG-based BCIs attract most of the BCI research due to their non-invasive, easy, and safe way of acquiring brain signals. EEG-based BCIs can be divided in four main categories [FRAG+12], namely P300-based BCIs [FD88], steady state visual evoked potential (SSVEP)-based BCIs [Her01], event related desynchronization (ERD)-based BCIs [PN01], and slow cortical potential-based BCIs [BKG+00]. Compared with the other categories of EEG-based BCIs, the P300-based BCIs have the following advantages. The P300-based BCIs are effective for almost every BCI user because the P300 signal, which is the target signal used in the P300-based BCIs, can be evoked in the brain of almost every human being [Els09]. In addition, the P300-based BCIs are relatively fast and straightforward to use. Moreover, the P300 signals work outstandingly well for BCI character spelling applications [GDS+09]. Therefore, the P300-based BCIs have attracted a lot of BCI researchers. As the benchmark for a P300-based BCI [FRAG+12], the P300 speller [FD88] has been the most-commonly investigated application of the P300-based BCI [FRAG+12]. Thus, this dissertation takes the P300 speller as the target BCI application.

1.1 Development Trends in P300-based Brain Computer Interface Systems

P300-based BCIs are still not used in human's daily life and remain in an experimental stage at research labs. In order to bring P300-based BCIs into practical use, currently, there are two development trends for P300-based BCI systems, i.e., to design high performance P300-based BCI systems and to design efficient P300-based BCI systems.

1.1.1 High Performance P300-based Brain Computer Interface Systems

The performance of a P300-based BCI system is the communication accuracy and the communication speed between the human brain and a computer. For example, for the P300 speller, the communication accuracy of such BCI system is the character spelling accuracy. The communication speed of such system is the Information Transfer Rate (ITR) [WRMP98]. The current performance of a P300-based BCI system is relatively low because the P300 signals are buried in a lot of noise and thus, the P300 signals have a very low Signal to Noise Ratio (SNR). This makes it difficult to detect P300 signals evoked in the human's brain, resulting in a low communication accuracy and speed of the P300-based BCI systems. P300-based BCI systems with such low performance are not acceptable for BCI users in their daily life. We take the P300 speller, the most-widely used application of the P300-based BCIs, as an example. Guy [GSB$^+$18] explored the usability of the current P300 spellers for disabled people with amyotrophic lateral sclerosis. This report shows that when using a current P300 speller, half of the subjects (persons) cannot spell characters with accuracy that is higher than 90%. To promote P300 spellers to be used in people's daily life, we should try our best to make the subjects spell characters with a P300 speller like the healthy people spell characters with their mouth. This means that we should try to make the subjects who use a P300 speller to achieve the character spelling accuracy that is (or close to) 100%. A P300 speller with accuracy that is much lower than 100% cannot be used in people's daily life. In addition, Guy's report [GSB$^+$18] also shows that when using the current P300 spellers, the mean number of characters correctly spelled by the subjects is 3.6 characters per minute. However, a healthy person is able to speak with around 120 characters per minute. Compared with 120 characters per minute, the communication speed of the current P300 spellers, i.e., 3.6 characters per minute, is far from what is needed to be used in people's daily life. Therefore, increasing the performance of the P300-based BCIs is a must in order to promote the P300-based BCIs into people's daily life.

To increase the performance of a P300-based BCI system, efforts are focused on the signal acquisition part and on the signal processing and translation part of a BCI system. In the signal acquisition part, researchers try to improve the recording quality of the EEG signals such that the signals, that contain P300 evoked potentials, have less noise. For example, Koka [KB07] has developed tripolar concentric sensors. These sensors use advanced engineering techniques to enhance the recording capability for brain signals. Unfortunately, the current signal recording techniques cannot provide high enough SNR for P300 signals, thereby not guaranteeing alone very good performance of a P300-based BCI system.

In recent years, massive efforts have been put in the signal processing and translation part of a P300-based BCI system. In order to increase the performance of the P300-based BCI system, a lot of studies have been done in terms of devising preprocessing, feature extraction, and classification methods for P300-based BCI systems. In terms of preprocessing EEG signals, different signal processing techniques are used, such as bandpass filtering [CG11], discrete-wavelet transform (DWT) [SS09], continuous wavelet transform (CWT) [Bos04]. In terms of feature extraction methods for P300-based BCIs, Rivet [RSAG09] uses an unsupervised algorithm to enhance P300 evoked potentials, Kulasingham [KVDS16] uses Stacked autoencoders (SAEs) to extract P300-related features. Researchers also try to remove artifacts in order to reduce the noise. For example, Gao [GZW10], Mennes [MWV$^+$10], and Gwin [GG$^+$10] propose signal processing methods to remove the artifacts caused by the muscle contraction, the eye movement, and the body movement, respectively. In terms of classification methods for P300-based BCIs, researchers have tried different classifiers, such as Support Vector Machine (SVM) [KMG$^+$04, RG08], Linear Discriminant Analysis (LDA) [JAB$^+$10], Fisher's Linear Discriminants (FLD) [SS09], Stepwise Linear Discriminant Analysis (SWLDA) [JK09], and neural networks (NN) [CG11, MG15, LWG$^+$18, SLS18], in order to improve the accuracy of detecting P300 signals. The rapid development of machine learning algorithms for P300-based BCIs boosts the performance improvement for P300-based BCI systems.

1.1.2 Efficient P300-based Brain Computer Interface Systems

P300-based BCI systems have been in an experimental stage at research labs for a long time. Traditional P300-based BCI systems, as shown in Figure 1.2, use a complex EEG headset which utilizes a large number of sensors for brain signal acquisition as well as they use a cumbersome computer for signal processing and translation. Even though such BCI systems may achieve high enough performance in some cases, such complex systems for P300-based BCIs cannot be used in people's daily life. This is because it is impossible for people who wear such complex headset and need such cumbersome computer to move freely everywhere they want. In order to bring P300-based BCIs into practical use, in recent years, researchers have been trying to develop efficient P300-based BCI systems. As shown in Figure 1.3[1], an efficient P300-based BCI uses a wireless EEG headset for signal acquisition. This headset utilizes a small number of sensors. In addition, such efficient BCI system uses a small mobile platform (e.g., mobile phone) for signal processing and translation. Since nowadays people use mobile phones almost every day and everywhere, it brings BCI users much convenience to use a mobile phone to process brain signals. Therefore, in recent years, a lot of research has been done for efficient P300-based BCI systems that use a wireless EEG headset for signal acquisition and a mobile phone for signal processing.

Concerning the wireless EEG headset, researchers have performed investigations to figure out the type of sensors as well as the number and the position of the sensors placed in the headset in order to build efficient P300-based BCI systems. Regarding the type of the sensors, traditional headsets, used in P300-based BCI systems, utilize wet sensors that operate with specially made conductive gels. The use of gels provides stable and high quality signal recording during a long-term use of a P300-based BCI system. However, the gels are sticky, which makes the BCI users' hair dirty and also

---

[1] This figure is taken from `https://www.emotiv.com/`.

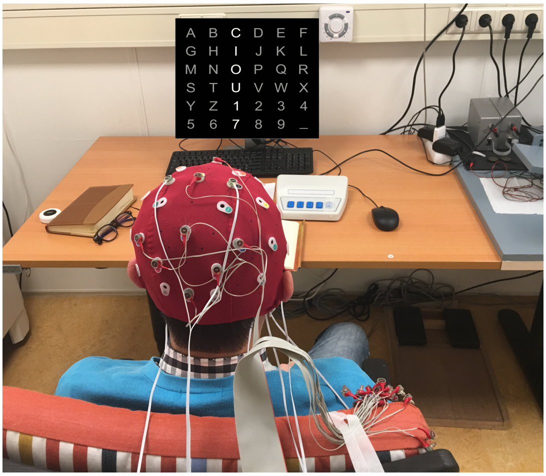


Figure 1.2: An example of a traditional P300-based BCI system.

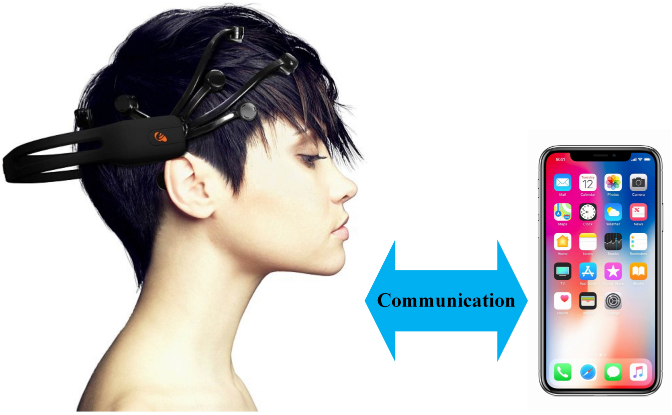


Figure 1.3: An example of an efficient P300-based BCI system.

makes users not comfortable. In addition, the preparation time of placing wet sensors on the BCI users' scalp is quite long. To make P300-based BCIs more convenient and user-friendly, researchers have developed dry sensors [GVF11, SRC$^+$12, CdBV$^+$14] for the acquisition of EEG signals. By using dry sensors, users do not need to use the conductive gels any more. To make the BCI users feel more comfortable, researchers also have developed non-contact sensors [HCP02, SDC07, ONB$^+$08] for EEG signal acquisition. Non-contact sensors are able to record EEG signals with a certain space between the brain skin and the headset. Unfortunately, dry sensors and non-contact sensors may impair the performance of the P300-based BCI systems because compared with wet sensors, dry sensors cannot provide the same high quality of recorded EEG signals, and non-contact sensors provide even lower quality of recorded EEG signals because non-contact sensors output a very small signal amplitude and they are very sensitive to artifacts [IS16].

In addition to the development of the aforementioned types of sensors, researchers also focus on reducing the number of sensors used in a P300-based BCI system while keeping the performance of this system acceptable [RG08, RSAG09, RCP$^+$10, CRC$^+$10, CRC$^+$11, RCS$^+$11, RCMM12, CRT$^+$14]. These studies propose sensor selection methods which select an appropriate sensor subset from an initial large set of sensors while keeping an acceptable BCI system performance. Such methods enable substantial reduction of the sensors needed to acquire EEG signals. The reduction of the number of sensors in a P300-based BCI system decreases the price of the EEG headset significantly, reduces the installation time of the P300-based BCI system, and also makes the users feel more comfortable. These advantages of the reduction of the sensors help promoting P300-based BCIs to be used in people's daily life.

After acquiring EEG signals from a wireless EEG headset, an efficient P300-based system uses a small mobile platform, such as a mobile phone, to process these signals. The mobile phone is an example of an embedded resource-constrained computing platform. Thus, the battery and memory of such platform are limited. As a result, the mobile phone cannot support the execution of signal processing algorithms with high complexity because such complex algorithms consume too much energy and memory where the amount of this consumption exceeds the limits of a mobile phone. Therefore, in order to build efficient P300-based BCI systems, signal processing algorithms with low complexity and acceptable performance are in urgent need. Such algorithms should be able to run on a mobile phone and consume a small amount of energy while keeping the system performance acceptable. In addition, in order to build energy-efficient P300-based BCI systems, techniques developed in the embedded system field can be used. Such techniques for energy-efficient task scheduling [LSWS16, CS16, NS17] and energy-efficient application mapping [LSCS15, SLS16] help the mobile phone to work energy-efficiently when used in a P300-based BCI system.

1.2 Problem Statement

The important development trends, described in Section 1.1, bring new opportunities to develop P300-based BCI systems. However, they also come with several issues when designing such systems. In this dissertation, we focus on several issues arisen by the aforementioned development trends in the contexts of the performance and the efficiency of the P300-based BCI systems. The specific problems, we address in this dissertation, are formulated as follows.

1.2.1 Problem 1

As discussed in Section 1.1.1, the performance of the P300-based BCI systems is very important to bring these BCIs into people's daily life. Since the P300 speller is the benchmark and the most-commonly investigated application of the P300-based BCIs, we focus on how to improve the performance of the P300 speller. In order to improve the performance of the P300 speller, previous research on P300 spellers uses traditional machine learning methods for the detection of P300 signals and the inference of characters in the P300 speller. The traditional machine learning methods use manually-designed signal processing techniques for feature extraction as well as classifiers like Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). Unfortunately, manually-designed feature extraction and traditional classification techniques have the following problems: 1) they can only learn the features that researchers are focusing on but lose or remove other underlying features; 2) brain signals have subject-to-subject variability, which makes it possible that methods performing well on certain subjects (with similar age or occupation) may not give a satisfactory performance on others. These problems limit the potential of manually-designed feature extraction and traditional classification techniques for further P300 detection accuracy, character spelling accuracy, and Information Transfer Rate (ITR)[2] improvements for the P300 speller.

Convolutional Neural Networks (CNNs) have the advantage of automatically extracting P300-related features from raw EEG signals. Thus, they can learn not only some features we know but also some features which are important and unknown to us. Automatically learning from raw EEG signals has better ability to achieve good results which are invariant to different subjects (persons). Thus, CNNs are able to boost the full potential of recognizing P300 signals, thereby overcoming the aforementioned shortcomings of traditional machine learning methods.

Therefore, in recent years, researchers have started to design (deep) CNNs for P300-based BCIs [CG11, MG15, LWG$^+$18] and achieved better P300 detection, accuracy, character spelling accuracy, and ITR than traditional techniques. However, these CNNs have some limitations in increasing the P300 detection accuracy, the character spelling accuracy, and ITR for the P300 speller. These CNNs first use a spatial convolution layer to learn P300-related spatial features from raw signals. Then, they use several temporal convolution layers to learn P300-related temporal features from the abstract temporal signals generated by the spatial convolution layer (the first layer).

---

[2] For the detailed description of ITR please refer to Section 2.4.3.

In this way, the input to the temporal convolution layers is the abstract temporal signals instead of raw temporal signals. These abstract temporal signals in the feature maps lose raw temporal information. Losing raw temporal information means losing important temporal features because the nature of P300 signals is the positive voltage potential in raw temporal information, see Figure 2.9 explained in Section 2.4.1, as well as many important P300-related features are also embodied in raw temporal information [Pol07]. As a result, these CNNs cannot learn temporal features well. This leads to issues such as: 1) these CNNs prevent further P300 detection accuracy, character spelling accuracy, and ITR improvements for the P300 speller, thereby impairing the performance of the P300 speller; 2) these CNNs have high network complexity to achieve competitive P300 detection accuracy, character spelling accuracy, and ITR for the P300 speller, thereby impairing the efficiency of the P300 speller. Thus, the first problem addressed in this dissertation is:

Problem 1: How can we design a CNN which achieves high P300 detection accuracy, character spelling accuracy, and ITR for the P300 speller and has low network complexity?

1.2.2 Problem 2

P300 spellers have been in an experimental stage at research labs for a long time. As discussed in Section 1.1.2, P300 spellers are still not used in people's daily life because the efficiency of these P300-based BCI systems is low, even though these systems may achieve high enough performance in some cases. Some reasons for this low efficiency are: 1) Current popular EEG headsets in the BCI systems used for the P300 speller utilize a large number of sensors to achieve high spelling accuracy. The price of the EEG headset is significantly high when the number of sensors is large because a lot of sensors require a complicated electrode cap and a lot of amplifier channels. 2) Utilizing a large number of sensors makes the P300 speller to consume a lot of energy, which is unacceptable for a battery-powered mobile BCI system. Such system utilizes a wireless EEG headset and a resource-constrained hardware platform for data processing. A large number of sensors increases the amount of the data needed to be recorded and processed, thereby increasing the energy consumption of the wireless EEG headset and the hardware platform. This does not allow a mobile P300 speller to work for a long time period on a single battery charge; 3) Utilizing a large number of sensors strengthens the user's discomfort and increases the installation time of the P300 speller.

To address the aforementioned issues caused by the utilization of a large number of sensors, sensor selection methods could be used to select an appropriate sensor subset from an initial large set of sensors while keeping acceptable spelling accuracy. So, a good sensor selection method should enable substantial reduction of

the sensors needed to acquire brain signals. Therefore, good sensor selection methods are in urgent need for designing comfortable, cheap, and energy-efficient P300 spellers and for promoting such P300 spellers into the human's daily life. Sensor selection methods for the P300 speller have been studied in recent years. For example, [RG08, RSG$^+$09, CRC$^+$10, CRC$^+$11] utilize a backward elimination algorithm as a sensor selection strategy. These works propose different ranking functions to evaluate and eliminate sensors such as the P300 signal detection accuracy, the P300 spelling accuracy [CRC$^+$11], the C_{cs} score [RG08], the Signal to Signal and Noise Ratio (SSNR) [RSG$^+$09, CRC$^+$10, CRC$^+$11], the Area Under the Receiver Operating Characteristic (AUC) [CRT$^+$14]. Alternatively, [CG11] and [LWG$^+$18] directly select the important sensors for a given user by analyzing the weights of a trained CNN. Unfortunately, the aforementioned sensor selection methods cannot select an appropriate sensor subset such that they can further reduce the number of sensors used to acquire EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used. As a consequence, the cost, energy consumption, and discomfort of a P300 speller are still unacceptably high when using the aforementioned sensor selection methods to design and configure P300 spellers. Therefore, the second problem addressed in this dissertation is:
Problem 2: How can we design a sensor selection method which is able to further reduce the number of sensors needed to acquire EEG signals while keeping the character spelling accuracy the same as the accuracy achieved when the initial large sensor set is used?

1.3 Research Contributions

In this section, we summarize the research contributions of this dissertation by addressing the research problems outlined in Section 1.2.

Contribution 1: Proposing a CNN architecture which has low complexity and achieves high P300 detection accuracy, character spelling accuracy, and ITR for the P300 speller.

To address Problem 1 in Section 1.2.1, we propose a simple, yet effective CNN architecture, called One Convolution Layer Neural Network (OCLNN), for the P300 speller. This CNN has only one convolution layer which is the first layer of the network. This layer performs both the spatial convolution and the temporal convolution at the same time, thereby learning very useful P300-related features from both raw temporal information and raw spatial information. Our OCLNN exhibits very low network complexity because it uses only one convolution layer and does not use fully-connected layers before the output layer. We perform experiments on three benchmark datasets and compare our results with those in previous research works that report the best results. The comparison shows that our proposed CNN can increase the P300 signal detection accuracy with up to 14.23% and the character spelling accuracy with up to 35.49%. The comparison also shows that our proposed CNN achieves comparable ITR with the related BN3 method [LWG$^+$18]. Moreover, our CNN achieves higher ITR compared to other state-of-the-art related methods [CG11, MG15, RG08, Bos04]. However, our OCLNN still has certain limitations to extract some important features related to P300 signals. OCLNN extracts P300-related spatial and temporal features at the same time in its single convolution layer, thereby extracting only P300-related joint spatial-temporal features through the spatial-temporal convolution. OCLNN does not extract P300-related separate temporal features and separate spatial features. These separate temporal features and separate spatial features have proven to be very important for the P300 speller [FTM$^+$88, Pol07, PNCB11, HVE06].

Contribution 2: Proposing an ensemble of different CNNs, we have devised, for the P300 speller.

Our OCLNN proposed in **Contribution 1** has the limitation that it cannot extract separate temporal features and separate spatial features related to P300 signals. Adding some temporal or spatial convolution layers following the first spatial-temporal convolution layer of OCLNN is a potential method to enable OCLNN to learn P300-related separate spatial or separate temporal features. Nevertheless, such potential method cannot learn P300-related separate temporal or spatial features well due to the loss of raw information. The raw information loss happens because the input to these added temporal or spatial convolution layers for OCLNN is the abstract signals generated by the first spatial-temporal convolution layer instead of raw signals. To address properly the aforementioned limitation of OCLNN (proposed in **Contribution 1**) , we propose an ensemble of two novel CNNs, we have devised, together with OCLNN in order to learn well the aforementioned P300-related separate spatial and separate temporal features, which are not extracted by OCLNN, together with the spatial-temporal features extracted by OCLNN. Our proposed ensemble of CNNs is called Ensemble of Convolutional Neural Networks (EoCNN). Our proposed two novel CNNs used in EoCNN are called One Spatial Layer Network (OSLN) and One Temporal Layer Network (OTLN), respectively. OSLN and OTLN has only one convolution layer. OTLN performs the temporal convolution in the first layer to learn P300-related separate temporal features. OSLN performs the spatial convolution in the first layer to learn P300-related separate spatial features. In this way, the input to OSLN and OTLN is raw signals, thus these two novel CNNs are able to learn features from raw signals. As a consequence, OTLN and OSLN can learn well P300-realted separate temporal features and separate spatial features, respectively. Our EoCNN uses the ensemble of OSLN and OTLN together with OCLNN, thereby extracting more useful P300-related features than OCLNN alone. As a result, our EoCNN can achieve higher P300 signal detection accuracy, character spelling accuracy, and ITR for P300 speller than OCLNN. Experimental results on three benchmark datasets show that our proposed EoCNN is able to increase the P300 signal detection accuracy, the character spelling accuracy, and the ITR achieved by OCLNN with up to 4.32%, 5%, and 6.05 bits/min, respectively. Also, our proposed EoCNN outperforms other related methods with a significant P300 signal detection accuracy improvement up to 18.55%, a significant character spelling accuracy improvement up to 38.72%, and a significant ITR improvement up to 21.75 bits/min. In terms of network complexity, the complexity of our EoCNN is lower than the complexity of the CNN in [MG15], and higher than the complexity of OCLNN and the CNNs in [CG11, LWG$^+$18].

Contribution 3: Proposing a CNN-based method for sensor reduction in the P300 speller.

To address Problem 2 in Section 1.2.2, we propose a novel CNN-based sensor selection method, called Spatial Learning based Elimination Selection (SLES). Compared with the state-of-the-art sensor selection methods [RG08, RSAG09, RCP$^+$10, CRC$^+$10, CRC$^+$11, RCS$^+$11, RCMM12, CRT$^+$14], our SLES is able to further reduce the number of sensors needed to acquire EEG signals in the P300 speller while keeping the character spelling accuracy the same as the accuracy achieved when an initial large set of sensors is used. Our SLES uses a novel parameterized CNN, we have devised, to evaluate and rank the sensors during the sensor selection process. This method features an iterative, parameterized, backward elimination algorithm to eliminate and select sensors. The parameter configured in this algorithm controls the training frequency of the CNN and the number of sensors to eliminate in every iteration. We perform experiments on three benchmark datasets and compare the minimal number of sensors selected by our SLES method and other selection methods needed to acquire brain signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large set of sensors is used. The results show that, compared with the minimal number of sensors selected by other methods, our method can reduce this number with up to 44 sensors.

Contribution 4: Proposing an improved ensemble of CNNs for the P300 speller with a small number of sensors.

As a result of **Contribution 2**, our EoCNN is able to achieve higher spelling accuracy and ITR compared to other state-of-the-art methods for the P300 speller. As a result of **Contribution 3**, our SLES method can reduce the number of sensors needed to acquire EEG signals in a EoCNN-based P300 speller while keeping the character spelling accuracy and the ITR the same as the character spelling accuracy and the ITR achieved by EoCNN when an initial large set of sensors is used in the P300 speller. We call the character spelling accuracy and the ITR, achieved by EoCNN for the P300 speller with a large number of sensors (e.g., 64 sensors), the state-of-the-art character spelling accuracy and ITR of the P300 speller. The experimental results mentioned in **Contribution 3** also show that in most cases, in order to preserve the state-of-the-art character spelling accuracy and ITR, we need to use more than 16 sensors to acquire EEG signals in the EoCNN-based P300 speller. Unfortunately, popular low-complexity and relatively cheap (affordable) BCI systems utilize a small number of sensors for the acquisition of EEG signals. Typically, such small number of sensors is less than or equal to 16 sensors. For example, BCI systems such as MUSE [MUS], EMOTIV Insight [Ins], Quick-8 [Qui], B-Alert X10 [B-A], EMOTIV EPOC+ [EMO], and OPEN BCI Mark IV [Mar] utilize only 4, 5, 8, 10, 14, and 16 sensors, respectively. Therefore, it is a challenge to achieve the state-of-the-art character spelling accuracy and ITR of the P300 speller with popular low-complexity and relatively cheap BCI systems that use a small number of sensors, i.e., less than or equal to 16 sensors, to acquire EEG signals.

To address the aforementioned challenge, we perform a study on our EoCNN (**Contribution 2**) as well as the three CNNs used in EoCNN, i.e., OTLN, OSLN, and OCLNN, for the P300 speller with different number of sensors in order to find the reason why EoCNN cannot achieve the state-of-the-art character spelling accuracy and ITR for a P300 speller with a small number of sensors. This study reveals that the reason for this is that EoCNN has the problem of putting equal importance on OSLN, OTLN, and OCLNN, when combining the outputs from these three CNNs for the P300 speller, irrespective of the number of sensors used to acquire EEG signals. In order to solve this problem of EoCNN, we propose an improved EoCNN for the P300 speller called PEoCNN. In PEoCNN, first, we parameterize the process of combining the outputs from OSLN, OTLN, and OCLNN. Then, we use the Sequential Model-based Algorithm Configuration (SMAC) [HHLB11] to automatically find and set values for the parameters depending on the number of sensors utilized in the P300 speller. In this way, PEoCNN is able to adapt/configure the importance of using the outputs from OSLN, OTLN, and OCLNN for the P300 speller depending on the number of sensors that are utilized. Experiments on three benchmark datasets show that when using our PEoCNN for the P300 speller, the state-of-the-art character spelling accuracy and ITR can be achieved in a BCI system with less than or equal to 16 sensors to acquire EEG signals.

1.4 Dissertation Outline

In this section, we give an outline of this dissertation:

Chapter 2 introduces some background information on Convolutional Neural Networks (CNNs), the P300 signal, the P300 speller, the Information Transfer Rate (ITR), and the datasets used in this dissertation.

Chapter 3 - 6 describe in details the contributions introduced in Section 1.3. Each chapter is organized in a self-contained way. That is, each chapter has its specific introduction, related work, proposed method, experimental evaluation, and conclusions.

Chapter 3 presents our proposed simple, yet effective, CNN architecture for the P300 signal detection and P300-based character spelling. This chapter is based on the following publication:

- **Hongchang Shan**, Yu Liu, and Todor Stefanov,
  "A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface",
  In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, pp. 1604-1610, Stockholm, Sweeden, July 13-19, 2018.

Chapter 4 presents our proposed ensemble of CNNs for the P300 signal detection and P300-based character spelling. This chapter is based on the following publication:

- **Hongchang Shan**, Yu Liu, and Todor Stefanov,
  "Ensemble of Convolutional Neural Networks for P300 Speller in Brain Computer Interface",
  In *Proceedings of the 28th International Conference on Artificial Neural Networks (ICANN'19)*, pp. 376-394, Munich, Germany, September 17-19, 2019.

Chapter 5 presents our proposed sensor reduction method for the P300 speller. This chapter is based on the following publications:

- **Hongchang Shan**, and Todor Stefanov,
  "SLES: A Novel CNN-based Method for Sensor Reduction in P300 Speller,"
  In *Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'19)*, pp. 3026-3031, Berlin, Germany, July 23-27, 2019.

- **Hongchang Shan**, and Todor Stefanov,
  "A Novel Sensor Selection Method based on Convolutional Neural Network for P300 Speller in Brain Computer Interface",
  The 56th ACM/IEEE Design Automation Conference (DAC'19) WIP session, Las Vegas, NV, USA, June 2-6, 2019.

Chapter 6 presents our proposed improved ensemble of CNNs for a P300 Speller with a small number of sensors. This chapter is based on the following publication:

- **Hongchang Shan**, Yu Liu, and Todor Stefanov,
  "An Empirical Study on Sensor-aware Design of Convolutional Neural Networks for P300 Speller in Brain Computer Interface,"
  In *Proceedings of "12th IEEE International Conference on Human System Interaction (IEEE HSI'19)"*, pp. 5-11, Richmond, Virginia, USA, June 25-27, 2019

Chapter 7 ends this dissertation by providing summary and conclusions regarding the work presented in this dissertation.

Chapter 2

Background

In this chapter, to better understand the contributions of this dissertation, we introduce some background information on machine learning, neural networks, Convolutional Neural Networks (CNNs), P300 signals, P300 spellers, the performance assessment of a P300 speller, and the datasets used in this dissertation.

In this dissertation, our proposed methods for P300-based BCIs are mainly based on CNNs. A CNN is a specific kind of a neural network. A neural network is a specific machine learning model in the machine learning field. Thus, first, we briefly introduce what machine learning is in Section 2.1. Then, we describe how a neural network works in Section 2.2. After that, we introduce what a CNN is in Section 2.3. The introductory text of each of the aforementioned sections is excerpts from well-known books with small modifications. For example, Section 2.1 is based on [Qiu], Section 2.2.1 and Section 2.2.2 are based on [Hay94], Section 2.2.3 and Section 2.3 are based on [Nie15].

The aim of this dissertation is to research and develop high performance and efficient P300-based BCI systems. Thus, we also introduce some background information on P300-based BCIs in Section 2.4.

Finally, in Section 2.5, we describe the datasets used in the dissertation to assess the performance of our proposed methods for P300-based BCIs.

2.1 Machine Learning

Machine learning is a subfield of artificial intelligence. Machine learning is the science of getting computers to learn from data in an autonomous manner [Sam67, M^+97].

Let us take an example to show how the machine learning works and also explain some other terminologies used in the machine learning field. Suppose that now we need to select good apples in a fruit market. How does the machine learning works to select good apples?

First, we takes some apples from the market. We list 3 features: color, shape, and size of each apple. The color, shape, and size are called the features that are related to apples. Then, we mark each apple with labels. For example, the label for each apple can be the label "good" or the label "bad". Labeled features and their corresponding labels constitute a dataset. Typically, there are two kinds of dataset, i.e., training dataset and test dataset. A machine learning method learns from the training dataset. The test dataset is used to assess the performance of this machine learning method.

We use a 3-dimension vector $X = [x_1, x_2, x_3]$ to denote a vector constructed by the aforementioned apple's features, where x_1, x_2, and x_3 denote the color, shape, and size of an apple, respectively. Here X is called a feature vector. We use a 2-dimension vector $y = [y_1, y_2]$ to denote a vector constructed by the aforementioned labels for an apple, where y_1 denotes the label "good" and y_2 denotes the label "bad". We use D to denote a training dataset. D is shown in Equation (2.1), where F denotes that there are in total F apples, which means there are F feature vectors in the training dataset; $X^{(i)}, i \in \{1, ..., F\}$ denotes the ith feature vector in the training dataset, and $y^{(i)}, i \in \{1, ..., F\}$ denotes the corresponding label for $X^{(i)}$.

$$D = \{(X^{(1)}, y^{(1)}), (X^{(2)}, y^{(2)}), ..., (X^{(F)}, y^{(F)})\} \tag{2.1}$$

For the given training dataset D, we want to get a computer to automatically find a function $f(X, \theta)$ to build a mapping from the feature vector X to the label y, where θ is a set of parameters of the function $f(\cdot)$. The function $f(X, \theta)$ is called a machine learning model. By using an algorithm A, we can find a set of parameters θ^* that is able to make the function $f(X, \theta^*)$ build the mapping from the feature vector X to the label y from the training dataset D. This process is called learning or training. The algorithm A used in the learning or training process to find θ^* is called a learning algorithm.

After we find $f(X, \theta^*)$ from the training dataset, when we buy new apples next time, based on the features (that constitute the feature vector) of the new apples X^*, we can use the aforementioned trained model $f(X, \theta^*)$ to predict the labels (i.e., good apples or bad apples) for these new apples X^*.

To summarize the aforementioned introduction to the machine learning, we show the workflow of a machine learning method in Figure 2.1. This figure shows that the input to a machine learning method is a feature vector X, the output of this machine learning method is the label y. A machine learning model $f(X, \theta)$ is used in the machine learning method. By using a learning algorithm A and the training dataset D, the machine learning method finds a set of parameters θ^* that makes the function

$f(X, \theta^*)$ build the mapping from the feature vector X to the label y. After this, the machine learning method gets a trained model $f(X, \theta^*)$. Then, for a new input X^*, the trained model $f(X, \theta^*)$ can predict a label for this new input X^*.

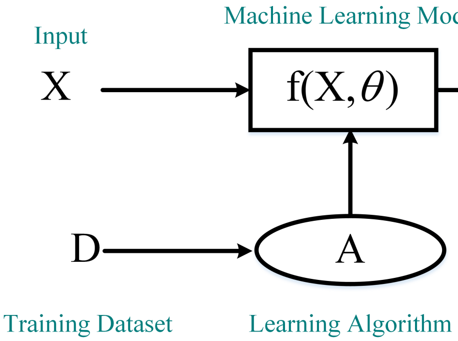


Figure 2.1: The workflow of machine learning.

2.2 Neural Network

A neural network is a specific machine learning model used in the machine learning method (introduced in Section 2.1). A neural network is made up of neurons . Therefore, we first introduce what a neuron is in a neural network in Section 2.2.1. Then, we describe how neurons constitute a neural network in Section 2.2.2. Finally, we introduce the learning algorithm used for neural networks.

2.2.1 Neurons

In this section, we introduce how a neuron works. First, we describe the model of a neuron in Section 2.2.1.1. Then, we introduce some functions used in the model of the neuron in Section 2.2.1.2.

2.2.1.1 Model of a Neuron

A neuron is an information processing unit which is fundamental to the operation of a neural network. Figure 2.2 shows the model of a neuron. We call this neuron neuron r since a neural network is constructed by many neurons (see Figure 2.4 and Figure 2.5 in Section 2.2.2). In Figure 2.2, neuron r takes several signals, namely, i_1, i_2, ..., i_m as inputs and produces a single output o_r. From this picture, we can see that a neuron has the following three basic elements:

1) A set of connecting links. Each of these links is characterized by a weight w_{rj}, $j \in [1, m]$. An input signal i_j is connected to neuron r by multiplying the weight w_{rj};

2) An adder. This adder sums the input signals (i_1, i_2, ..., i_m), weighted by the corresponding weights mentioned above;

3) An activation function. This activation function limits the amplitude of the output signal o_r to be in a certain range. Typically, the range of the output of a neuron is limited to $[0, 1]$ or $[-1, 1]$.

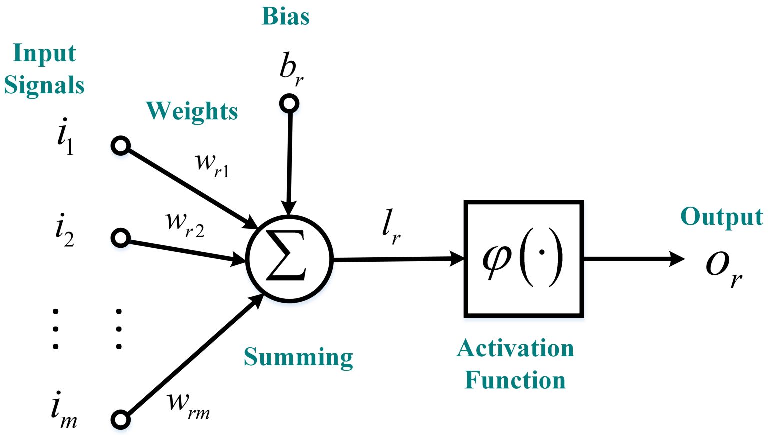


Figure 2.2: The model of a neuron.

Figure 2.2 shows that the model of a neuron also includes an externally applied parameter called bias. The bias is denoted by b_r in this model of neuron r. The bias b_r is used to increase or decrease the input of the activation function.

In mathematical terms, we can model neuron r by using Equations (2.2), (2.3) and (2.4), where i_1, i_2, ..., i_m are the input signals to neuron r; w_{r1}, w_{r2}, ..., w_{rm} are the weights of neuron r; b_r is the bias; $\varphi(\cdot)$ is the activation function; and o_r is the output of neuron r.

$$u_r = \sum_{j=1}^{m} w_{rj} i_j \tag{2.2}$$

$$l_r = u_r + b_r \tag{2.3}$$

$$o_r = \varphi(l_r) \tag{2.4}$$

2.2.1.2 Types of Activation Functions

The aforementioned activation function, denoted by $\varphi(\cdot)$, defines the output of neuron r. Here, we introduce some basic activation functions:

1) Threshold Function. The threshold function is given by Equation (2.5), where $\varphi(\cdot)$ denotes the activation function; and l_r denotes the input to the activation function $\varphi(\cdot)$, and l_r is defined using Equation (2.3).

$$\varphi(l_r) = \begin{cases} 1 & if \quad l_r \geq 0 \\ \\ 0 & if \quad l_r < 0 \end{cases} \tag{2.5}$$

2) Sigmoid Function. The sigmoid activation function is given by Equation (2.6), where a is used to control the output of the sigmoid function. Note that the output of a sigmoid function is in a continuous range $[0, 1]$ while the output of the threshold function is ether 1 or 0.

$$\varphi(l_r) = \frac{1}{1 + exp(-al_r)} \tag{2.6}$$

3) Rectified Linear Unit (ReLU) Function. The ReLU function is given by Equation (2.7). ReLU is by far the most commonly used activation function in CNNs.

$$\varphi(l_r) = max(0, l_r) \tag{2.7}$$

4) Softmax Function. The Softmax function is given by Equation (2.8), where p denotes that there are p neurons in total in a layer of a neural network. The layer of a neural network is described in Section 2.2.2.

$$\varphi(l_r) = \frac{e^{l_r}}{\sum_{n=1}^{p} e^{l_n}} \tag{2.8}$$

2.2.2 The Architecture of a Neural Network

Neurons constitute a neural network. In this section, we describe the architecture of a neural network that is constructed by the neurons. For better readability, we simplify the model of a neuron shown in Figure 2.2 with the graph shown in Figure 2.3. This means that the graph shown in Figure 2.3 denotes neuron r with input signals i_1, i_2,

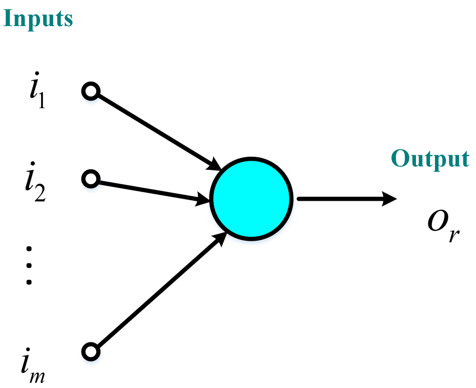


Figure 2.3: Architectural graph to model a neuron.

..., i_m and an output o_r. How neuron r works in this graph is given by Equations (2.2), (2.3) and (2.4) (For details please see Section 2.2.1.1).

In a neural network, the neurons are organized in the form of layers. In the simplest form of a neural network, we have an input layer of input signals that projects onto an output layer of neurons. We call this neural network the single-layer neural network. Here "single-layer" refers to the output layer of the network. We do not count the input layer of this network because there is no computation performed in the input layer. Figure 2.4 shows an example of a single-layer neural network. In this example, this single-layer neural network has four input signals and has one output layer of two neurons that produce outputs.

Typically, a neural network has more than one layer. Compared with a single-layer neural network, a neural network with more complex architecture has several layers of neurons. Such network is called a multi-layer neural network. In addition to the output layer and the input signals, a multi-layer neural network has one or more hidden layer of neurons. The function of the hidden layer of neurons is to intervene between the external input signals and the outputs of the network. By adding one or more hidden layers for the network, the network is enabled to extract higher-order features related to the input signals. The ability of extracting higher-order features by the hidden layers of neurons is particularly valuable when the size of the input layer is large. Figure 2.5 shows an example of a multi-layer neural network with one hidden layer and one output layer. In this example, this multi-layer neural network has 4 input signals, one hidden layer with 3 neurons, and the output layer with 2 neurons. The input signals are the input to the hidden layer of this network. The output signals of this hidden layer are the input to the output layer of the network. Typically, the input to the neurons in each layer of a multi-layer neural network is the output signals of its preceding layer only. The neural network shown in Figure 2.5 is also called a fully-connected neural network in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer. Here a node denotes an input signal or a neuron in the graph shown in this figure.

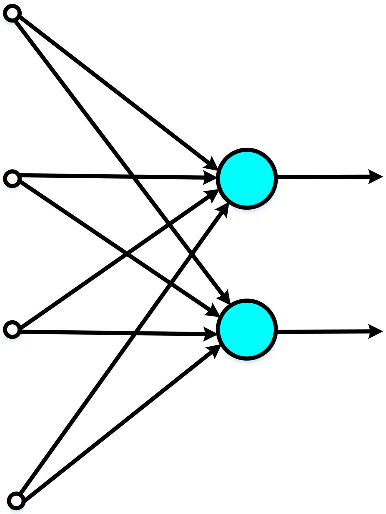


Figure 2.4: An example of a single-layer neural network.

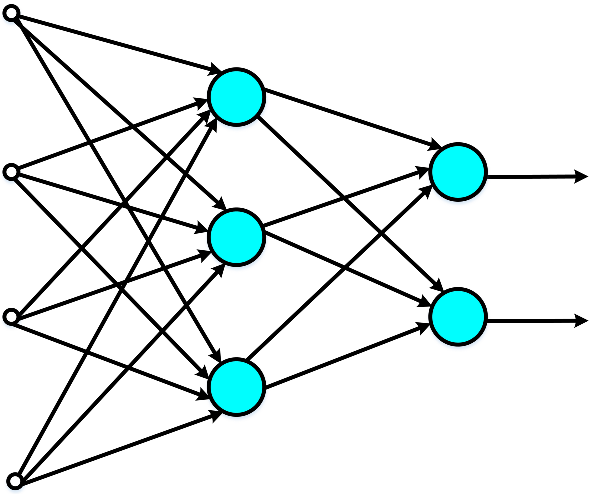


Figure 2.5: An example of a multi-layer neural network with one hidden layer and one output layer.

2.2.3 Learning Process of a Neural Network

As discussed in Section 2.1, when we use a neural network as a machine learning model, we need to use a learning algorithm to find a set of parameters, i.e., the weights and biases of all neurons in the neural network, that make this neural network be able to map input X to label y in the training dataset. However, the learning algorithm is not able to calculate the perfect weights and biases for a neural network. Instead, the learning process of a neural network is regarded as an optimization problem, where the learning algorithm is used to explore the space of possible sets of weights and biases for the neural network. We use a function to evaluate a candidate solution (i.e. a set of weights and biases for the neural network). This function is called a cost function or a loss function. For example, we can define a cost function as given in Equation (2.9). In this equation, X denotes the input vector in the training dataset and $X^{(i)}$ is the ith input feature vector in the training dataset. y denotes the desired output of the network and $y^{(i)}$ is the desired output of the network when the input to the network is $X^{(i)}$. $f_{NN}(\cdot)$ denotes the neural network. W denotes all the weights in the neural network. B denotes all the biases in the network. F denotes the total number of input feature vectors. This loss function is called the Mean Squared Error (MSE) cost function. From this cost function, we can see that $C(W, B)$ is non-negative. When the cost $C(W, B)$ becomes very small, i.e., $C(W, B)$ is close to 0, $f_{NN}(X, W, B)$ is approximately equal to y. This means that the learning algorithm has found very good weights W and biases B such that the neural network with these weights and biases approximately maps the input of this network X to the desired output of the network y. In contrast, when the cost $C(W, B)$ is large, this means that $f_{NN}(X, W, B)$ is not equal to y, showing that our neural network with the weights W and the biases B cannot map well the input of this network X to the output of the network y. Now, the cost function that is commonly-used for a neural network is called the cross-entropy cost function. The cross-entropy cost function is given in Equation (2.10). In this equation, E denotes that there are E neurons in the output layer of a neural network. $y_j^{(i)}$ denotes the desired output of the jth neuron in the output layer of a neural network when the input to the network is $X^{(i)}$. $f_{NN_j}(X^{(i)}, W, B)$ denotes the actual output of the jth neuron in the output layer of a neural network when the input to the network is $X^{(i)}$.

$$C(W, B) = \frac{1}{F} \sum_{i=1}^{F} \|y^{(i)} - f_{NN}(X^{(i)}, W, B)\|^2 \tag{2.9}$$

$$C(W, B) = -\frac{1}{F} \sum_{i=1}^{F} \sum_{j=1}^{E} [y_j^{(i)} log(f_{NN_j}(X^{(i)}, W, B)) + (1 - y_j^{(i)}) log(1 - f_{NN_j}(X^{(i)}, W, B))] \tag{2.10}$$

From the aforementioned description, we can see that the objective of the learning algorithm is to minimize the cost $C(W, B)$. More specifically, we seek to find a set of weights W and biases B that minimize this cost as much as possible. For better readability, we use $C(v)$ to denote a cost function. $C(v)$ can be a function of many parameters such as v = v_1, v_2, ..., v_h. Suppose C is a function of just two parameters v_1, v_2. Figure 2.6 shows an example of function C with v_1 and v_2. Our object is to find v_1, v_2 where C achieves its minimum. For the simple function shown in Figure 2.6, we can use calculus to try to find the minimum analytically. We could compute derivatives and then try using them to find places where C is an extremum. However, the cost function of a neural network can have many more parameters and be much more complex because a neural network contains much more parameters, i.e., the weights and biases of all neurons in the network. For example, very large neural networks have cost functions that depend on billions of weights and biases. It is impossible to use calculus to minimize the cost function.

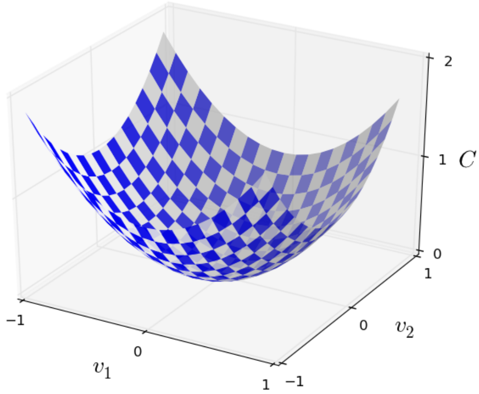


Figure 2.6: An example of a cost function C with two parameters v_1 and v_2.

To solve the aforementioned minimization problem, we can use a method, called gradient descent. We use an analogy to explain how the gradient descent method works to solve this minimization problem. This analogy is shown in Figure 2.7. As shown in this figure, we can think of our cost function as a kind of a valley and imagine a ball rolling down the slope of the valley. When the ball reaches the bottom of this valley, this means we find the minimum of the cost function. Then, the problem comes to how we make a rule that makes the ball roll down to the bottom of the valley. We can use the calculus to describe the move of a ball with a small amount $\triangle v_1$ in the v_1

direction and a small amount $\triangle v_2$ in the v_2 direction by using Equation (2.11). We need to find $\triangle v_1$ and $\triangle v_2$ that make $\triangle C$ negative (negative $\triangle C$ means that the ball is rolling down into the valley). We define that $\triangle v$ is equal to $(\triangle v_1, \triangle v_2)^T$ as shown in Equation (2.12), where T is the transpose operation that turns row vectors into column vectors in a matrix. We define that the gradient of C, denoted by $\bigtriangledown C$, is equal to the vector of partial derivatives $(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2})^T$, as shown in Equation (2.13). With these definitions, Equation (2.11) can be rewritten to be Equation (2.14). In order to make $\triangle C$ negative, we can make $\triangle v$ to be equal to $-\eta \bigtriangledown C$ as shown in Equation (2.15), where η is a small, positive parameter, called the learning rate. Then, by combing Equation (2.14) and Equation (2.15), $\triangle C = -\eta \bigtriangledown C \cdot \bigtriangledown C = -\eta \| \bigtriangledown C\|^2$. Because $\| \bigtriangledown C\|^2 \geq 0$ and $\eta > 0$, $\triangle C \leq 0$. This means that C will always decrease and never increase. Thus, we use Equation (2.15) to define the rule of how to move the ball in the gradient descent algorithm. This means that we use Equation (2.15) to compute a value $\triangle v$ and then move the position of the ball v to a new position v' by the amount of $\triangle v$, as shown in Equation (2.16). Then we will use updated rule again to make another movement of the ball. By keeping doing this, we can decrease C until we reach the (approximate) minimum of the cost function C.

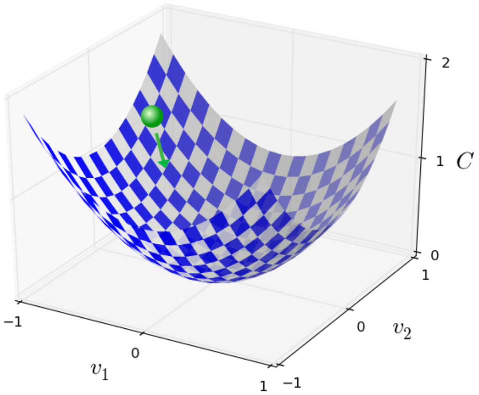


Figure 2.7: The analogy of using gradient descent to minimize a cost function.

$$\triangle C \approx \frac{\partial C}{\partial v_1} \triangle v_1 + \frac{\partial C}{\partial v_2} \triangle v_2 \tag{2.11}$$

$$\triangle v = (\triangle v_1, \triangle v_2)^T \tag{2.12}$$

$$\bigtriangledown C = (\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2})^T \tag{2.13}$$

$$\triangle C \approx \bigtriangledown C \cdot \triangle v \tag{2.14}$$

$$\triangle v = -\eta \bigtriangledown C \tag{2.15}$$

$$v \rightarrow v' = v - \eta \bigtriangledown C \tag{2.16}$$

The aforementioned discussion describes how the gradient descent method works when the cost function C has two parameters. When C is a function of h parameters, i.e., v_1, v_2, ..., v_h, $\bigtriangledown C$ is calculated using Equation (2.17). We repeatedly apply the rule shown in Equation (2.18) until we reach the (approximate) minimum of the cost function C.

$$\bigtriangledown C = (\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}, ..., \frac{\partial C}{\partial v_h})^T \tag{2.17}$$

$$v \rightarrow v' = v - \eta \bigtriangledown C \tag{2.18}$$

In fact, for a neural network, v is constituted by all weights W=w_1, w_2, ..., w_d and all biases B=b_1, b_2, ..., b_g. Therefore, to update the weights of a neural network, we repeatedly apply the rule shown in Equation (2.20) to reach the (approximate) minimum of the cost function C. In Equation (2.20), $\bigtriangledown C_W$ is calculated using Equation (2.19). Also, to update the biases of a neural network, we repeatedly apply the rule shown in Equation (2.22) to reach the (approximate) minimum of the cost function C. In Equation (2.22), $\bigtriangledown C_B$ is calculated using Equation (2.21).

$$\triangledown C_W = (\frac{\partial C}{\partial w_1}, \frac{\partial C}{\partial w_2}, ..., \frac{\partial C}{\partial w_d})^T \tag{2.19}$$

$$W \rightarrow W' = W - \eta \triangledown C_W \tag{2.20}$$

$$\triangledown C_B = (\frac{\partial C}{\partial b_1}, \frac{\partial C}{\partial b_2}, ..., \frac{\partial C}{\partial b_g})^T \tag{2.21}$$

$$B \rightarrow B' = B - \eta \triangledown C_B \tag{2.22}$$

Now, researchers often use the gradient descent method with momentum, which is called the momentum-based gradient descent method. The momentum technique modifies the gradient descent method in two ways. Firstly, the momentum technique introduces a notion of "velocity" for the parameters we optimize. The gradient descent method changes the "velocity" of the parameters, not (directly) the "position" of the parameters, and only indirectly affects the "position" of the parameters. Secondly, the momentum technique introduces a friction term, which can gradually reduce the "velocity" of the parameters. In mathematical terms, the momentum-based gradient descent method replaces the updating rule for W (given in Equation (2.20) used in the gradient descent method without momentum) with a new updating rule given in Equation (2.23), and (2.24), where V_w denotes the aforementioned "velocity" for W, and μ denotes the aforementioned friction term and is called the momentum parameter. Also, the momentum-based gradient descent method replaces the updating rule for B (given in Equation (2.22) used in the gradient descent method without momentum) with a new updating rule given in Equation (2.25), and (2.26), where V_b denotes the aforementioned "velocity" for B.

$$V_w \rightarrow V_w' = \mu V_w - \eta \triangledown C_W \tag{2.23}$$

$$W \rightarrow W' = W + V_w' \tag{2.24}$$

$$V_b \rightarrow V_b' = \mu V_b - \eta \triangledown C_B \tag{2.25}$$

$$B \rightarrow B' = B + V_b' \tag{2.26}$$

One problem of the learning process of a neural network is called overfitting. Overfitting happens when a neural network learns the details of the training data to the extent that it negatively impacts the performance of this neural network on new data. This means that random fluctuations in the training data is learned by the neural network. Unfortunately, the learned random fluctuations cannot apply on new data, thereby negatively impacting the generalizing ability of the network.

In order to reduce the overfitting, one commonly-used technique, called weight decay, is utilized. The weigh decay technique modifies the updating rule for weights W and does not change the updating rule for biases B in the gradient descent method. The gradient descent method with weight decay replace the updating rule for W (given in Equation (2.20) used in the gradient descent method without weight decay) with a new updating rule given in Equation (2.27). In Equation (2.27), λ is called the weight decay parameter; F denotes the total number of input feature vectors and η is the learning rate. The updating rule for B used in the gradient descent method with weight decay is the same as the updating rule for B (given in Equation (2.22)) used in the gradient descent method without weight decay.

$$W \rightarrow W' = (1 - \frac{\eta\lambda}{F})W - \eta \bigtriangledown C_W \tag{2.27}$$

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is a specific kind of neural network. In recent years, CNNs have been the most commonly-used neural networks to recognize images.

When using a fully-connected neural network (introduced in Section 2.2.2) to recognize images, the fully-connected neural network has the problem that it uses a large number of parameters to recognize images. Before introducing this problem, let us first describe how to use a neural network to recognize images. We take the handwritten digit recognition as an example of image recognition. The handwritten digit recognition is to recognize what digit (e.g., 1,3, 6,...) is for an image of a handwritten digit. From the discussion in Section 2.1 and Section 2.2.3, we can see that when using the machine learning method to recognize a handwritten digit, we need to develop a machine learning model. Here, we use a neural network, denoted by $f_{NN}(X, W, B)$, as a machine learning model, where f_{NN} denotes a neural network; W denotes all the weights in the neural network; B denotes all the biases in the network. X is called a tensor and denotes the inputs to the neural network. For example, if X is an image with 640 × 480 pixels, we call X a (640 × 480) tensor. We train this neural network $f_{NN}(\cdot)$ with the training dataset that consists of handwritten digits with their corresponding labels (e.g., 1, 3, 6). As introduced in Section 2.2.3, the gradient descent is used to find the weights W and biases B that make the neural network $f_{NN}(X, W, B)$ build the (approximate) mapping from the handwritten digits to the labels. Then, for a new handwritten digit, the trained neural network can predict a label for this new handwritten digit.

Up to this point, we have known how to use a neural network to recognize images of handwritten digits. Now let us introduce the reason of why the fully-connected neural network has the problem of using a large number of parameters to recognize images. In the aforementioned example of the handwritten digit recognition, the input is an image of a handwritten digit. This image is 28 × 28 pixel image. This means that the number of the input signals in the input layer of a fully-connected neural network is 784 = 28 × 28. Suppose a simple fully-connected neural network which architecture is a 2-layer network with one hidden layers and one output layer. We suppose that each hidden layer has 10 neurons and the output layer of this neural network has 10 neurons. The number of parameters (i.e., all weights and biases) of this neural network is (784 × 10+10) + (10 × 10 + 10) = 7960. Unfortunately, such simple fully-connected neural network cannot work well to recognize handwritten digits. Suppose a more complex fully-connected neural network which architecture is a 3-layer network with two hidden layers and one output layer. Each hidden layer of this network has 50 neurons, and the output layer still has 10 neurons. The number of parameters of this network is 42290. From this example, we can see that with the increase of the number of hidden neurons and hidden layers, the number of the parameters of a fully-connected layer is dramatically increased. This means that when we design a fully-connected neural network that can be useful for image recognition, the number of the parameters of such network will be large. The large number of parameters dramatically increases the time of training such fully-connected neural network because in the training process, the gradient descent algorithm will need quite a long time to find a large number of parameters that make this network (approximately) maps the handwritten digits to the labels (For details of the training process of a neural network, please see Section 2.2.3).

To address the aforementioned problem of the fully-connected neural network, Convolutional Neural Network is proposed to recognize images. The name "Convolutional Neural Network" indicates that this neural network utilizes a mathematical operation called convolution. We will introduce what the convolution operation is in Section 2.3.1. Then, we introduce the characteristics of a CNN in Section 2.3.2. Finally, we introduce the architecture of a CNN in Section 2.3.3.

2.3.1 The Convolution Operation

The convolution operation is an important operation in analytical mathematics. In this section, we introduce the 2-dimension convolution operation because the 2-dimension convolution operation is widely used for image recognition and also used in our proposed CNN-based methods for P300-based BCIs in this dissertation.

The 2-dimension convolution operation is defined by Equation (2.28), where $\otimes$ denotes the convolution operation. Z, X, and K are 2-dimension matrices. X denotes the input matrix to the convolution operation; Z denotes the outputs of the convolution operation; K is called the kernel of the convolution operation. (k_1, k_2) is called the kernel size. (s_1, s_2) is called the stride. In Equation (2.28), $Z(i, j)$ denotes the datum in the ith row and the jth column of the matrix Z; $X(i, j)$ denotes the datum in the ith row and the jth column of the matrix X; and $K(m, n)$ denotes the datum in the mth row and the nth column of the matrix K.

$$\begin{aligned} Z(i,j) &= (X \otimes K)(i,j) \\ &= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} X((i-1)s_1 + 1 + m, (j-1)s_2 + 1 + n) K(m,n) \end{aligned} \tag{2.28}$$

2.3.2 The Characteristics of Convolutional Neural Network

A CNN has three characteristics, i.e., the receptive field, the parameter sharing, and the pooling layers. We introduce these three characteristics in turn with the following.

1) **Receptive Field**. Firstly, we introduce what a receptive field in a CNN is. We still take the handwritten digit recognition as an example to describe the receptive field. When using a CNN to recognize the handwritten digits, the input to a CNN is a 28 × 28 pixel image. Let us take this input as a 28 × 28 square of input neurons. The value of each input neuron is the value of the corresponding pixel in the image. We take a small region of the input neurons to connect each hidden neuron in the first hidden layer of the CNN. For example, we can take a 5 × 5 region, containing 25 input neurons that corresponding to 25 input pixels, to connect to each hidden neuron in the hidden layer of the CNN. The region, which contains several neurons in a layer and is connected to a neuron in the next layer, is called the receptive field. Each connection which connects the receptive field with the hidden neuron represents a weight. One hidden neuron uses one bias. In this way, a hidden neuron is used to learn the features from a receptive field of input neurons. We then slide the receptive field across the entire input image. For example, first, the receptive field in the top-left corner of the input image is connected to the top-left hidden neuron in the first hidden layer. Then, we slide the receptive field over by one pixel (i.e., by one input neuron) to the right

to connect to the second hidden neuron in the first hidden layer. For each receptive field, there is a corresponding hidden neuron connected with this receptive field in the hidden layer. In this way, we build a hidden layer, and this hidden layer is called a feature map.

2) **Parameter Sharing**. Secondly, we introduce the parameter sharing in the CNN. Here, the parameter denotes the weights and biases used in a CNN. As described above, each hidden neuron in the hidden layer uses one biases and some wights that are connected to the corresponding receptive field. The parameter sharing means that all the neurons in a hidden layer of a CNN use the same weights and bias. The characteristic of parameter sharing in CNN is able to significantly reduce the number of parameters. Thus, the parameter sharing can solve the problem in the fully-connected neural networks, i.e., the fully-connected neural networks use a large number of parameters when used to recognize a image.

3) **Pooling Layers**. Finally, we introduce the pooling layers in the CNN. Pooling layers are often used after the convolution layers. The pooling operation in a pooling layer converts each feature map from the convolution layer into a condensed feature map by summarizing a region of neurons in the feature map. For example, the pooling operation, called max-pooling, takes the maximum in a region of neurons as a unit in the pooling layer. A CNN often uses more than one feature map, and the pooling operation is applied to each feature map, separately. Thus, if there are 5 feature maps, the pooling layer will output 5 condensed feature maps.

2.3.3 The Architecture of Convolutional Neural Network

After introducing the characteristics of the CNN, we describe the architecture of a CNN. The architecture of a CNN consists of three kinds of layers, i.e., convolution layers, pooling layers and fully-connected layers. A convolution layer performs the convolution operation introduced in Section 2.3.1. A pooling layer performs the pooling operation described in Section 2.3.2. A fully-connected layer performs the fully-connected operation shown in Section 2.2.2. Figure 2.8 shows an example of the architecture of a CNN used to recognize the handwritten digits. In this example, the input to the CNN is a 28×28 pixel image of a handwritten digit. The CNN has one convolution layer, one pooling layer, and one fully-connected layer. The input to the convolution layer is 28×28 input neurons. The convolution layer outputs 3 feature maps. These three feature maps are the input to the pooling layer. The pooling layer apply the pooling operation and outputs 3 condensed feature maps. These condensed feature maps are the input to the fully-connected layer. This fully-connected layer has 10 neurons that correspond to the 10 possible labels (e.g., 0, 1, 2, 3, ..., 9) of the handwritten digits. This fully-connected layer connects each of the 10 neuron with the condensed feature maps generated from the pooling layer.

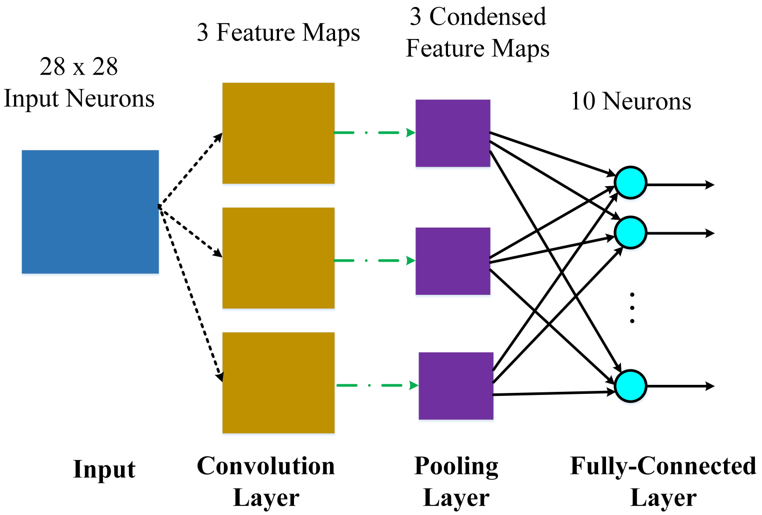


Figure 2.8: An example of the architecture of a CNN used for the handwritten digit recognition.

2.4 P300-based Brain Computer Interface

In this dissertation, we focus on P300-based BCIs. The P300 signal is the target signal used in a P300-based BCI and the P300 speller is the benchmark and the most commonly-used application of a P300-based BCI. First, we introduce some background information on the P300 signal and the P300 speller in Section 2.4.1 and Section 2.4.2, respectively. Then, we describe the metrics to assess the performance of the P300 speller in Section 2.4.3.

2.4.1 P300 Signal

Chapman and Bragdon first discovered the P300 signal in 1964 [CB64]. The P300 signal is a kind of an event-related potential (ERP). The P300 signal, recorded in EEG, occurs with a positive deflection in voltage at a latency about 300ms after a rare stimulus, as shown in Figure 2.9. The P300 signal is also called P3 wave because it is the third major positive peak in the late sensory and the late positive component [Pic92].

The P300 signal is an endogenous evoked potential because the evoking of a P300 signal does not have any relationship with the physical attributes of a stimulus, but has a relationship with a subject's (person's) reaction to the stimulus. The P300 signal is usually elicited using the oddball paradigm. In this paradigm, the target stimulus appears in low probability while the non-target stimulus appears in high probability.

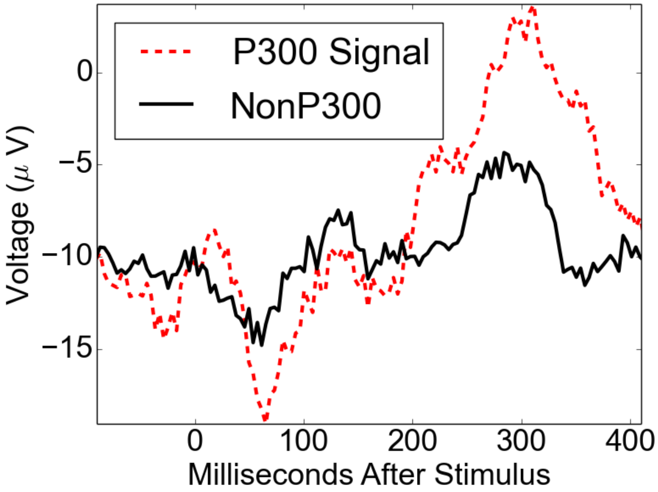


Figure 2.9: P300 signal.

The subject in this paradigm is detecting a rare target stimulus among the non-target stimuli. The P300 signal is only able to be evoked in the subject's brain when this subject detects the rare target stimulus.

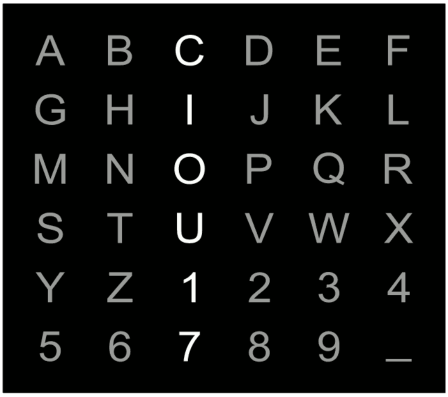
The P300 signal is typically measured most strongly by the electrodes covering the parietal lobe. The amplitude of a P300 signal varies with the rareness of the target stimulus. The latency of a P300 signal varies with the difficulty of discriminating the target stimulus from the non-target stimuli. For example, the typical latency of a P300 signal evoked in a young healthy adult is about 300ms while the latency of a P300 signal evoked in subjects (persons) with decreased cognitive ability is longer than 300ms. Due to its reproducibility and ubiquity, the P300 signal is a common choice for psychological tests in both the clinic and laboratory.

From the aforementioned description of the P300 signal, we can infer when the subject detects a target stimulus by detecting the evoked P300 signal in the subject's brain signals. The detection of P300 signals from brain signals can be considered as a binary classification problem. There are two classes in this classification problem: one class corresponds to the presence of a P300 signal within a certain time period while the second class corresponds to the absence of a P300 signal within the time period. If we use E to denote a classifier (e.g., a CNN as introduced in Section 2.3) to classify the P300 signal, and X to denote the subject's brain signals within a certain time period, the P300 signal detection process can be expressed as Equation (2.29). In this equation, P^1 denotes the probability, predicted by this classifier, of having a P300 signal in this time period, P^0 denotes the probability, predicted by this classifier, of not having a P300 signal in this time period.

$$E(X) = \begin{cases} 1 & if \quad P^1 > P^0 \\ 0 & otherwise \end{cases} \tag{2.29}$$

2.4.2 P300 Speller

Farwell and Donchin developed the first P300-based BCI character speller in 1988 [FD88]. The subject in the experiment is presented with a 6 by 6 character matrix (see Figure 2.10) and he focuses his attention on a target character he wants to spell. All rows and columns in this matrix are intensified successively and randomly but separately. Each row or column intensification lasts for time period t_1, followed by a blank time period of the matrix t_2. Two out of twelve intensifications contain the target character, i.e., one target row and one target column. As a result, the target row/column intensification becomes a rare stimulus to the subject. A P300 signal is then evoked by this rare stimulus. By detecting the P300 signal, we can infer which row or column the subject is focused on. By combing the row and column positions, we can infer the target character position. After the inference of one character, the matrix is blank for time period t_3 to inform the subject that the current character is completed and to focus on the next character.



A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	1	2	3	4
5	6	7	8	9	_

Figure 2.10: P300 speller character matrix.

Assume that one epoch includes 12 intensifications, in which there exist one target row intensification and one target column intensification. Then, in theory, one epoch is sufficient to infer one target character. However, in practice, since the P300 signal has a very low Signal to Noise Ratio (SNR) and is also influenced by artifacts, one epoch can hardly be sufficient to infer one target character correctly. As a result, in practice, experimenters use many epochs to help the subject spell one character. The detailed calculation for determining the position of the target character is given in Equations (2.30), (2.31), and (2.32), where $P^1_{(i,j)}$ denotes the probability of the presence of a P300 signal in the jth intensification and the ith epoch, $Sum_{(j)}$ denotes the sum of the probabilities for the jth intensification when using k epochs, $index_{col}$ denotes the column index of the target character in the matrix in Figure 2.10, and $index_{row}$ denotes the row index of the target character. When $j \in [1, 6]$, j denotes a column intensification. When $j \in [7, 12]$, j denotes a row intensification. Equation (2.30) cumulates the probabilities of having a P300 signal evoked by intensification j over k epochs. In Equation (2.31), we assign the index of the maximum $Sum_{(j)}$ to $index_{col}$ when $j \in [1, 6]$. This equation finds the index of the column intensification, with the maximum sum of probabilities, to have evoked a P300 signal. This index is the column position of the target character when using k epochs. In Equation (2.32), the row position of the target character when using k epochs is calculated in the same way as in Equation (2.31). The position of the target character in the matrix in Figure 2.10 is the coordinate formed by the target row position and the target column position.

$$Sum_{(j)} = \sum_{i=1}^{k} P^1_{(i,j)} \tag{2.30}$$

$$index_{col} = \underset{1 \leq j \leq 6}{argmax} \left\{Sum_{(j)}\right\} \tag{2.31}$$

$$index_{row} = \underset{7 \leq j \leq 12}{argmax} \left\{Sum_{(j)}\right\} \tag{2.32}$$

2.4.3 Performance Assessment of P300 Speller

As indicated in Chapter 1, in this dissertation, we use the P300 speller as the benchmark application of a P300-based BCI. As the benchmark application of a P300-based BCI, we need metrics to assess the performance of the P300 speller. More specifically, we need metrics to assess the communication accuracy and the communication speed of the P300 speller. As typically done in related research works for the P300 speller, we use the character spelling accuracy to assess the communication accuracy of the P300 speller as well as we use the Information Transfer Rate (ITR) to assess the communication speed of the P300 speller.

To calculate the character spelling accuracy of the P300 speller, we use Equation (2.33). In this equation, $acc_{char(k)}$ denotes the character spelling accuracy when using k epochs for each character (see Section 2.4.2), $N_{tc(k)}$ denotes the number of correctly inferred characters when using k epochs for each character, and S_c denotes the number of all characters.

$$acc_{char(k)} = \frac{N_{tc(k)}}{S_c} \tag{2.33}$$

In addition to using the character spelling accuracy to assess the communication accuracy of the P300 speller, we also use the Information Transfer Rate (ITR) for the assessment of the communication speed of the P300 speller. ITR has been the most commonly applied metric to assess the communication speed of P300-based BCIs [WW12, LWG16, NRS17, IKV18]. ITR has been introduced by Shannon and Weaver [SW49]. It is calculated by Equation (2.34) [WRMP98], where $acc_{char(k)}$ is calculated using Equation (2.33) and N_{cla} is the number of classes. Here, we have 36 characters to spell (see Figure 2.10), so N_{cla} =36. T_k denotes the time needed to spell a character when using k epochs. T_k is calculated using Equation (2.35), where t_1, t_2, and t_3 are the time periods described in the first paragraph of Section 2.4.2. For more detailed explanation of Equation (2.34), please refer to [WRMP98].

$$ITR_k = \frac{60(acc_{char(k)} \log_2(acc_{char(k)}) + (1 - acc_{char(k)}) \log_2(\frac{1-acc_{char(k)}}{N_{cla}-1}) + \log_2(N_{cla}))}{T_k} \tag{2.34}$$

$$T_k = t_3 + 12 \times (t_1 + t_2) \times k \quad 1 \leq k \leq 15 \tag{2.35}$$

Here, we calculate the theoretical maximum ITR when the datasets described in Section 2.5 are used in this dissertation because we want to compare the ITR achieved by our methods with the theoretical maximum ITR. When using the datasets described in Section 2.5, N_{cla}=36, t_1=100ms, t_2=75ms, and t_3=2.5s (for details please refer to Section 2.5). The theoretical maximum ITR is achieved when we use the least time to spell a character and achieve the highest spelling accuracy. The least time to spell a character means that we use only one epoch to spell a character. i.e., k=1. Thus, the least time we use to spell a character is T_1= 2.5s + 12 × (0.1s+0.075s) = 4.6s. The highest spelling accuracy is 100% (i.e., $acc_{char(1)} = 1$). As a result, the theoretical maximum ITR when using the datasets described in Section 2.5 is $\frac{60 \log_2(36)}{4.6}$ bits/min = 67.43 bits/min.

2.5 Datasets

This dissertation uses three benchmark datasets, namely, BCI Competition II - Data set IIb [Bla03] as well as BCI Competition III - Data set II Subject A and Subject

B [Bla08]. Since many methods for P300-based BCIs use these three benchmark datasets, we can fairly compare the character spelling accuracy and ITR, introduced in Section 2.4.3, achieved by our CNN-based P300 speller with the character spelling accuracy and ITR achieved by other state-of-the-art methods for the P300 speller. Here, we give a short description of the three datasets.

BCI Competition II - Data set IIb and BCI Competition III - Data set II Subject A and Subject B are provided by the Wadsworth Center, NYS Department of Health. They are recorded with the BCI2000 platform [SMH$^+$04], using the P300 speller described in Section 2.4.2. EEG signals are collected from 64 sensors at a sampling frequency of 240Hz. One intensification lasts for time period t_1=100ms, followed by a blank time period of the matrix t_2=75ms. The experiment uses 15 epochs for each character. Each character epoch is represented by 12 sets of signal samples. One set of signal samples, as shown in Figure 2.11, is a (N, C) matrix. In this matrix, each row has $N = T_s \times F_s$ (F_s is the signal sampling frequency) signal samples in the time period between 0 and T_s posterior to the beginning of each intensification, and each column has the signals samples taken at the same time from all C sensors used in the EEG headset. After each sequence of 15 epochs, the matrix is blank for time period t_3=2.5s to inform the subject that the current character is completed and to focus on the next character.

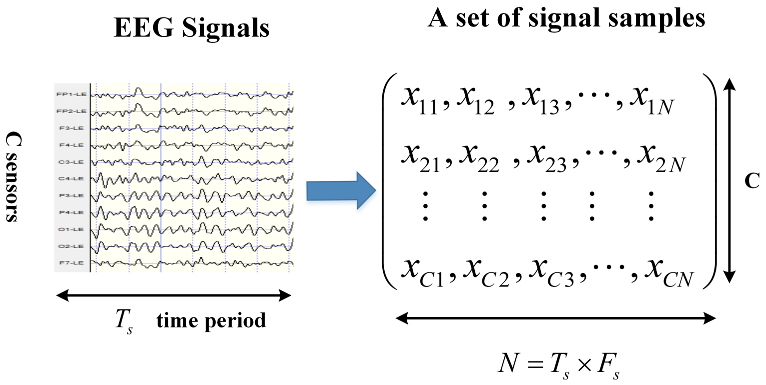


Figure 2.11: An example of a set of signal samples, where F_s is the signal sampling frequency

In BCI Competition II - Data set IIb, there is one subject with separate training and test datasets. The training dataset has 42 characters and the test dataset has 31 characters. In each character epoch, represented by 12 sets of signal samples, 2 sets have a P300 signal and 10 sets do not have a P300 signal. So, the training dataset has 42 × 15 × 2 = 1260 sets of signal samples labelled "P300", and there are 42 × 15 ×

10 = 6300 sets labelled “non-P300”. The test dataset has 930 sets of signal samples labelled “P300” and 4650 sets labelled “non-P300”.

In BCI Competition III - Data set II, there are two subjects. We call them Subject A and Subject B. For each subject, the training dataset has 85 characters and the test dataset has 100 characters. So, the training dataset has 2550 sets of signal samples labelled “P300” and 12750 sets labelled “non-P300”. The test dataset has 3000 sets of signal samples labelled “P300” and 15000 sets labelled “non-P300”.

Table 2.1 shows the number of P300s/non-P300s for each dataset. II denotes BCI Competition II - Data set IIb, III-A denotes BCI Competition III - Data set II Subject A, and III-B denotes BCI Competition III - Data set II Subject B.

Table 2.1: Number of P300s/non-P300s for each dataset.

Dataset	Train		Test	
	P300	non-P300	P300	non-P300
II	1260	6300	930	4650
III-A	2550	12750	3000	15000
III-B	2550	12750	3000	15000

Chapter 3

A Simple Convolutional Neural Network for P300 Signal Detection and Character Spelling

Hongchang Shan, Yu Liu, and Todor Stefanov,
"A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface,"
In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)*, pp. 1604-1610, Stockholm, Sweeden, July 13-19, 2018.

The P300 speller has been the benchmark and the most-commonly used application of P300-based BCI systems [FRAG$^+$12]. Previous research on the P300 signal detection and character spelling in the P300 speller uses traditional machine learning methods, namely manually-designed signal processing techniques for feature extraction as well as classifiers like Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). It focuses on enhancing P300 potentials [RSAG09], extracting useful features [Bos04], choosing the most relevant EEG sensors [CRC$^+$11], or removing artifacts caused by the muscle contraction [GZW10], the eye movement [MWV$^+$10] and the body movement [GG$^+$10]. Unfortunately, manually-designed feature extraction and traditional classification techniques have the following problems: 1) they can only learn the features that researchers focus on but lose or remove other underlying features; 2) brain signals have subject-to-subject variability, which makes it possible that methods performing well on certain subjects (with similar age or occupation) may not give a satisfactory performance on others. These problems limit the potential of manually-designed feature extraction and traditional classification techniques for further P300 signal detection and character spelling accuracy improvements.

In recent years, deep learning, especially using Convolutional Neural Networks (CNNs), has achieved significant performance improvements in the computer vision field [KSH12, SZ14, HZRS16]. Deep CNNs have the advantage of automatically learning feature representations from raw data[1]. They can learn not only something we know but also something important and unknown to us. Automatically learning from raw data has better ability to achieve good results which are invariant to different subjects. Thus, CNNs are able to boost the full potential of recognizing BCI signals, overcoming the aforementioned shortcomings of traditional machine learning methods.

Therefore, in recent years, researchers have started to design (deep) CNNs for P300-based BCIs [CG11, MG15, LWG$^+$18] and achieved better P300 signal detection and character spelling accuracy than traditional techniques. However, these CNNs have some limitations in increasing the P300 signal detection and character spelling accuracy. These CNNs first use a spatial convolution layer to learn P300-related spatial features from raw signals. Then, they use several temporal convolution layers to learn P300-related temporal features from the abstract temporal signals generated by the spatial convolution layer (the first layer). In this way, the input to the temporal convolution layers is the abstract temporal signals instead of raw temporal signals. In fact, raw temporal signals are more important to learn P300-related temporal feature. Therefore, these CNN architectures cannot learn P300-related temporal features well and this leads to problems that: 1) they prevent further P300 signal detection and character spelling accuracy improvements; 2) they require high network complexity to achieve competitive accuracy, which prevents the use of these CNNs for practical mobile-based BCIs [WWJ11, CFF16].

To solve the problems mentioned above, we propose a simple, yet efficient CNN architecture which can capture feature representations from both raw temporal and raw spatial information. The network complexity is significantly reduced while increasing the P300 signal detection accuracy, character spelling accuracy, and the communication speed. The novel contributions of this chapter are the following:

- We propose a CNN architecture with only one convolution layer. Our CNN is able to better learn P300-related features from both raw temporal information and raw spatial information. Our CNN exhibits much lower network complexity compared to other state-of-the-art CNNs [CG11, MG15, LWG$^+$18] for the P300 speller.

- We perform experiments on three benchmark datasets and compare our results

---

[1] In this dissertation, we use “raw data, information, or signals” to denote the data that is only pre-processed (e.g., bandpass filtering and normalization) but not abstracted by a feature extraction method (e.g., a CNN).

with those in previous research works that report the best results. The comparison shows that our proposed CNN can increase the P300 signal detection accuracy with up to 14.23% and the character spelling accuracy with up to 35.49%. The comparison also shows that our proposed CNN achieves comparable communication speed with the related BN3 method [LWG+18]. Moreover, our CNN achieves higher communication speed compared to other state-of-the-art related methods [CG11, MG15, RG08, Bos04].

The rest of this chapter is organized as follows: Section 3.1 describes the related work. Section 3.2 presents our proposed CNN. Section 3.3 compares the complexity, the P300 signal detection accuracy, the character spelling accuracy, and the communication speed between the proposed CNN and other methods for P300 signal detection and character spelling. Section 3.4 ends this chapter with conclusions.

3.1 Related Work

The general architecture of the CNNs for P300-based BCI [CG11, MG15, LWG+18] uses the input tensor[2] ($N \times C$) shown in Figure 3.1, where N denotes the number of temporal signal samples and C denotes the number of sensors used for EEG signal recording and obtaining the samples. This architecture has three stages. In the first stage, it performs convolution along space to learn P300-related spatial features. In the second stage, it performs convolution along time to learn P300-related temporal features. In the final stage, it uses fully-connected layers to make accurate correlation between learned features and a particular class.

Cecotti [CG11] is the first to propose the aforementioned architecture. Let us call his architecture CCNN. Table 3.1 shows the detailed architecture of CCNN. The first column in the table describes the sequence of layers. The second column describes the operation in a layer. The third column describes the kernel size of the convolution operation[3] in the convolution layers. The last column describes the number of feature maps/neurons in a layer.

Liu [LWG+18] improves CCNN by combining Batch Normalization [IS15] and Droupout [SHK+14] techniques (see Table 3.2). This CNN is named BN3 in [LWG+18]. BN3 uses the Batch Normalization operation in two layers: one is in Layer 1 and the other is in Layer 3. BN3 also employs Dropout in the fully-connected layers to reduce overfitting[4]. Before the output layer, BN3 uses two fully-connected layers instead of one for better generalization and accumulation of features.

---

[2] The notion of a tensor is introduced in the second paragraph in Section 2.3.

[3] The convolution operation and the notion of the kernel size are introduced in Section 2.3.1.

[4] The problem of overfitting is introduced in Section 2.2.3.

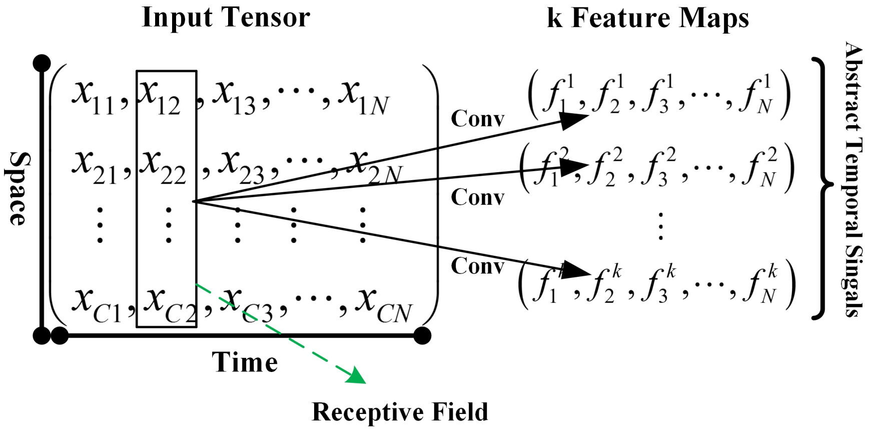


Figure 3.1: Abstraction of the raw signals in the spatial convolution layer in current CNNs. x denotes a signal sample in the input tensor. f denotes a datum in a feature map. Every column in the input tensor contains a set of C signal samples. These samples come from C sensor at a certain sampling time point. The spatial convolution operation converts each column of spatial data (receptive field) from the input tensor into an abstract datum in a feature map.

Manor [MG15] proposes a deep CNN achitecture for the P300-based BCI. Let us call his architecture CNN-R. It is shown in Table 3.3. CNN-R improves CCNN by using a deeper and wider network architecture. It uses a smaller kernel size for the temporal convolution operations but more layers for these temporal convolution operations (see Layer 2 and Layer 3 in Table 3.3). It also uses two fully-connected layers before the output layer. In addition, CNN-R uses more feature maps for the convolution layers and more neurons for the fully-connected layers. For such complex network, CNN-R uses Pooling (see Section 2.3.2) as well as Dropout to reduce overfitting.

The problem of the aforementioned CNNs is that they learn P300-related temporal features from abstract signals instead of raw signals, which makes these CNNs not able to learn P300-related temporal features well. P300-related temporal features are extracted by the temporal convolution layers of these CNNs. The input to these temporal convolution layers is the feature maps generated by the spatial convolution layer (the first layer). These feature maps are abstract temporal signals instead of raw signals because this spatial convolution layer converts each receptive field (see Section 2.3.2) of raw signals into an abstract datum in a feature map, as shown in Figure 3.1. These abstract temporal signals in the feature maps lose raw temporal information. Losing raw temporal information means losing important temporal features because the

Table 3.1: CCNN architecture.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(1,C)	10
2	Convolution	(13,1)	50
3	Fully-Connected	—	100
Output	Fully-Connected	—	2

Table 3.2: BN3 architecture.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Batch Norm	—	—
	Convolution	(1,C)	16
2	Convolution	(20,1)	16
	Batch Norm	—	16
3	Fully-Connected	—	128
	Dropout	—	128
4	Fully-Connected	—	128
	Dropout	—	128
Output	Fully-Connected	—	2

nature of P300 signals is the positive voltage potential in raw temporal information (see Figure 2.9 explained in Section 2.4.1) as well as many important P300-related features are also embodied in raw temporal information [Pol07]. As a result, these CNNs cannot learn P300-related temporal features well. Due to this problem, the aforementioned CNNs have to use a deeper and wider network architecture to learn temporal features better and achieve competitive accuracy. As a result, these CNNs exhibit high complexity.

In contrast, our novel CNN architecture performs both spatial convolution and temporal convolution in the first layer instead of performing only spatial convolution as in the aforementioned CNNs. Thus, the input to this convolution layer in our CNN is raw signals. In this way, the data used to extract P300-related temporal features is raw signals instead of the abstract signals in the aforementioned CNNs. As a result, our CNN is able to learn P300-related feature representations from raw temporal information and at the same time, it can also learn P300-related spatial features. Therefore, our CNN learns P300-related temporal features better. By learning in this

Table 3.3: CNN-R architecture.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(1,C)	96
	Pooling	(3,1)	96
2	Convolution	(6,1)	96
	Pooling	(3,1)	96
3	Convolution	(6,1)	96
4	Fully-Connected	—	2048
	Dropout	—	2048
5	Fully-Connected	—	4096
	Dropout	—	4096
Output	Fully-Connected	—	2

way, our CNN can achieve better P300 signal detection and character spelling accuracy (see Section 3.3.3 and Section 3.3.4) with only one convolution layer and without fully-connected layers before the output layer, which reduces the network complexity significantly (see Section 3.3.2).

3.2 Proposed Convolutional Neural Network

In this section, we introduce our novel CNN. We call it One Convolution Layer Neural Network (OCLNN). First, in Section 3.2.1, we describe the input to the network. Then, in Section 3.2.2, we describe our proposed network architecture. Finally, in Section 3.2.3, we explain how we train the network.

3.2.1 Input to the Network

The input to OCLNN is the input tensor ($N \times C$) shown in Figure 3.2 and Figure 3.3, where C denotes the number of sensors used for EEG signal recording and obtaining the samples. N denotes the number of temporal signal samples. Here N= $T_s \times F_s$, where T_s denotes the time period between 0 and T_s posterior to the beginning of each row/column intensification (see Section 2.4.2 and Section 2.5), and F_s denotes the signal sampling frequency.

Figure 3.2 shows that a set of signal samples from the EEG signals, introduced in Section 2.5, is preprocessed to obtain the input tensor which is used as the input to our proposed CNN. In such set of signal samples, the temporal signal samples are

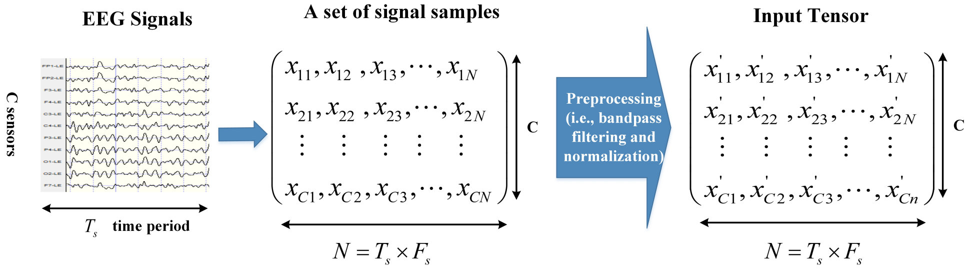

EEG Signals
C sensors
T_s time period
A set of signal samples
$x_{11}, x_{12}, x_{13}, \cdots, x_{1N}$
$x_{21}, x_{22}, x_{23}, \cdots, x_{2N}$
$x_{C1}, x_{C2}, x_{C3}, \cdots, x_{CN}$
C
$N = T_s \times F_s$
Preprocessing (i.e., bandpass filtering and normalization)
Input Tensor
$x'_{11}, x'_{12}, x'_{13}, \cdots, x'_{1N}$
$x'_{21}, x'_{22}, x'_{23}, \cdots, x'_{2N}$
$x'_{C1}, x'_{C2}, x'_{C3}, \cdots, x'_{Cn}$
C
$N = T_s \times F_s$

Figure 3.2: Input tensor for our proposed OCLNN.

bandpass filtered between 0.1Hz and 20Hz to remove the high frequency noise. Then, the filtered samples are normalized using Equation (3.1), (3.2), and (3.3) to have zero mean and unit variance based on each individual pattern and for each sensor. Each individual pattern represents N signal samples in the time period between 0 and T_s posterior to the beginning of each intensification. Such normalization is a common practice for preprocessing input data to CNNs. The normalization helps the CNN to perform well for the P300 signal detection and character spelling [CG11].

$$x'_{ij} = \frac{x_{ij} - \varepsilon}{\delta} \tag{3.1}$$

$$\varepsilon = \frac{1}{N} \sum_{j=1}^{N} x_{ij} \tag{3.2}$$

$$\delta = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (x_{ij} - \varepsilon)^2} \tag{3.3}$$

3.2.2 Network Architecture

The architecture of OCLNN is described in Table 3.4 and illustrated in Figure 3.3. The first column in the table describes the sequence of layers. The second column describes the operation in a layer. The third column describes the kernel size of the convolution operation in the convolution layer. The last column describes the number of feature maps/neurons in a layer. We have 2 layers in total, i.e., Layer 1 and Layer Output.

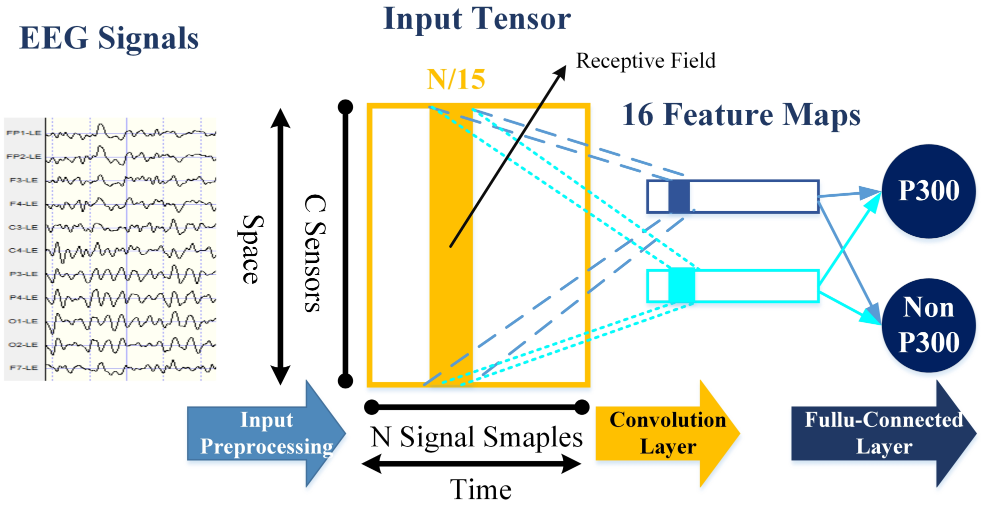


Figure 3.3: Illustration of OCLNN for P300 signal detection.

Table 3.4: OCLNN architecture.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	$(N/15,\mathrm{C})$	16
	Dropout	—	—
Output	Fully-Connected	—	2

In Layer 1, we segment the input tensor over the time domain into 15 parts and perform convolution operation on each part to learn features. Therefore, the kernel size of the convolution operation is $(N/15, C)$ and each receptive field (see the orange rectangle in Figure 3.3) of the input tensor is a tensor $(N/15, C)$ of signal samples. In the time domain, these signal samples come from a time period of $T_s/15$. In the space domain, these signal samples come from all C sensors. The convolution operation in this layer converts each receptive filed of data into an abstract datum in a feature map. In this way, this layer learns features from both raw temporal information and raw spatial information. The stride[5] used for the convolution operation in this layer is $(N/15, C)$. We use the Rectified Linear Unit (ReLU) function[6] as an activation function to model a neuron's output in this layer because a network with ReLUs is trained much faster than with other traditional activation functions [KSH12]. In this

---

[5] The notion of the stride is introduced in Section 2.3.1.

[6] The Rectified Linear Unit (ReLU) function is introduced in Section 2.2.1.2.

layer, we employ Dropout [SHK+14] to reduce overfitting. The Dropout rate is set to be 0.25. This layer generates 16 feature maps.

In Layer Output, OCLNN performs the fully-connected operation. There are two neurons in this layer. One neuron represents the class "P300" and the other neuron represents the class "non-P300". The fully-connected operation makes correlation between the feature maps from Layer 1 and the two classes. We employ the Softmax function[7] as an activation function for the neurons in this layer. The output of the Softmax function for class "P300" and class "non-P300" is denoted by $P^1_{(i,j)}$ and $P^0_{(i,j)}$, respectively. Therefore, $P^1_{(i,j)}$ represents the probability of having a P300 signal and $P^0_{(i,j)}$ represents the probability of not having a P300 signal at epoch i and intensification j. Thus, the detection of a P300 signal is defined by Equation (3.4), where $X_{(i,j)}$ is the input tensor to be classified and E_{ocl} denotes our OCLNN. By using Equation (3.4) to detect P300 signals, we can assess the performance of our proposed OCLNN in terms of the P300 signal detection accuracy (see Section 3.3.1 and Section 3.3.3).

$$E_{ocl}(X_{(i,j)}) = \begin{cases} 1 & if \quad P^1_{(i,j)} > P^0_{(i,j)} \\ 0 & otherwise \end{cases} \tag{3.4}$$

We use $P^1_{(i,j)}$, the output of OCLNN for class "P300", to calculate the position of the target character in the P300 speller character matrix. For the detailed calculation process please refer to Section 2.4.2, Equation (2.30), (2.31), and (2.32).

3.2.3 Training

The training of OCLNN is carried out by minimizing the cross-entropy cost function[8]. We use a Stochastic Gradient Descent (SGD) based learning algorithm, which is a modified version of the gradient descent based learning algorithm[9]. For more details on the SGD-based learning algorithm, please refer to [Bot10]. We use the SGD-based learning algorithm with momentum[10] and with weight decay[11]. The momentum parameter μ is set to 0.9. The weight decay parameter λ is set to 0.0005. The learning rate[12] η is fixed to 0.01. The aforementioned setup of the training parameters follows the suggestion in [SZ14] because [SZ14] has shown that when using these parameters to train a CNN on a training dataset, this trained CNN is able to achieve good performance on a test dataset. For details on the training process of a neural network please refer to Section 2.2.3.

---

[7] The Softmax function is given in Equation (2.8) introduced in Section 2.2.1.2.

[8] The cross-entropy cost function is given in Equation (2.10) introduced in Section 2.2.3.

[9] The gradient descent based learning algorithm is introduced in Section 2.2.3.

[10] The momentum technique is given in Equation (2.23), (2.24), (2.25), and (2.26) introduced in Section 2.2.3.

[11] The weigh decay technique is given in Equation (2.27) introduced in Section 2.2.3.

[12] The notion of the learning rate is introduced in Equation (2.15) in Section 2.2.3.

3.3 Experimental Evaluation

First, we introduce our experimental setup in Section 3.3.1. Then, we show the performance comparison between OCLNN and other related research works in terms of complexity (see Section 3.3.2), P300 signal detection accuracy (see Section 3.3.3) and character spelling accuracy (see Section 3.3.4). Finally, we compare the ITR of the P300 speller based on our OCLNN and other methods (see Section 3.3.5).

3.3.1 Experimental Setup

Our OCLNN is implemented using Keras [C$^+$15] with the Tensorflow [AA$^+$16] backend. The network is trained on an NVIDIA GeForce GTX 980 Ti GPU.

We train our OCLNN using each training dataset in Dataset II, III-A and III-B, described in Section 2.5, separately, thereby obtaining three different OCLNNs with different parameters (i.e., different weights and biases). The number of used sensors is 64 and the signal sampling frequency is 240 Hz (Section 2.5). Therefore, for the input to OCLNN (see Section 3.2.1), we have C = 64 and F_s = 240 Hz. T_s = 1000ms because we take each individual pattern to be the signal samples between 0 and 1000 ms posterior to the beginning of each intensification. Then, the number of temporal signal samples N = $T_s \times F_s$ = 240.

We run each of the three trained OCLNNs on the corresponding test dataset in Dataset II, III-A and III-B and calculate the P300 signal detection accuracy using Equation (3.5), the character spelling accuracy using Equation (2.33) (introduced in Section 2.4.3), and the Information Transfer Rate (ITR) using Equation (2.34) and (2.35) (introduced in Section 2.4.3) for each test dataset. In Equation (3.5), acc_{P300} denotes the P300 signal detection accuracy, N_{tp} denotes the number of truly classified P300s for a test dataset, N_{tn} denotes the number of truly classified non-P300s for the test dataset, and S_{pn} denotes the number of all P300s and non-P300s in the test dataset.

$$acc_{P300} = \frac{N_{tp} + N_{tn}}{S_{pn}} \tag{3.5}$$

For a fair comparison with CNN-R [MG15], we apply the bandpass filtering methods used for our OCLNN on CNN-R because we obtain low character spelling accuracy for CNN-R using the original filtering method in [MG15].

3.3.2 Complexity

In this section, we compare the complexity, in terms of the number of parameters and layers, of OCLNN with the networks CCNN [CG11], BN3 [LWG$^+$18], and CNN-R [MG15] briefly described in Section 3.1. The number of parameters is the number of weights and biases for all neurons in a network. We show the complexity in Table 3.5. The first row in the table lists the CNNs, we compare. The second row provides the number of parameters for each CNN. The third row shows the number of layers used in each CNN.

Table 3.5: Complexity comparison of different CNNs.

	OCLNN	CCNN	BN3	CNN-R
Parameters	16882	37502	39489	21950818
Layers	2	4	5	6

In terms of the number of parameters, OCLNN is much smaller than the other three CNNs. OCLNN has only 16882 parameters whereas CCNN has 37502 parameters[13], BN3 has 39489 parameters, and CNN-R has 21950818 parameters. Thus, the number of parameters for OCLNN is only 45%, 42%, and 0.07% of that for CCNN, BN3, and CNN-R, respectively.

In terms of number of layers used in a CNN, OCLNN has less layers than the other three CNNs. OCLNN has only 2 layers whereas CCNN has 4 layers, BN3 has 5 layers, and CNN-R has 6 layers. Thus, the number of layers in OCLNN is only 50%, 40%, and 33.33% of that in CCNN, BN3, and CNN-R, respectively.

3.3.3 P300 Signal Detection Accuracy

This section compares the P300 signal detection accuracies achieved by OCLNN with the accuracies achieved by CCNN, BN3, and CNN-R on Dataset II, III-A and III-B.

The P300 signal detection accuracy is shown in Table 3.6. The first row in the table lists the CNNs used for comparison. The second, third, and last row show the P300 signal detection accuracy of the different CNNs on Dataset II, III-A, III-B, respectively. The numbers are given in percentage (%) and calculated using Equation (3.5). An accuracy number in bold indicates the highest accuracy along a row. "–" in the table means that the accuracy is not reported in the reference paper describing the corresponding CNN.

Overall, OCLNN achieves the highest accuracies among all CNNs on Dataset II, III-A and III-B. It increases the P300 signal detection accuracies achieved by the other CNNs by up to 14.23%. For Dataset II, OCLNN achieves 92.41% P300 signal detection accuracy. The accuracy achieved by OCLNN is 7.97% and 6.12% higher than the accuracy achieved by BN3 and CNN-R, respectively. For Dataset III-A, OCLNN

---

[13] Cecotti [CG11] calculated the number of parameters erroneously for L_2. It should be $5N_s \times (13 \times N_s + 1)$ instead of $5N_s \times (13+1)$

Table 3.6: P300 signal detection accuracy of different CNNs on Dataset II, III-A and III-B.

	OCLNN	CCNN	BN3	CNN-R
P300 Accuracy on II	**92.41**	–	84.44	86.29
P300 Accuracy on III-A	**84.60**	70.37	75.13	73.06
P300 Accuracy on III-B	**86.40**	78.19	79.02	79.80

achieves 84.60% P300 signal detection accuracy. The accuracy achieved by OCLNN is 14.23%, 9.47%, and 11.54% higher than the accuracy achieved by CCNN, BN3, and CNN-R, respectively. For Dataset III-B, OCLNN achieves 86.40% P300 signal detection accuracy. The accuracy achieved by OCLNN is 8.21%, 7.38%, and 6.60% higher than the accuracy achieved by CCNN, BN3, and CNN-R, respectively.

3.3.4 Character Spelling Accuracy

This section compares the character spelling accuracies achieved by OCLNN and the accuracies achieved by CCNN, BN3, CNN-R, and ESVM [RG08] for Dataset III-A and III-B, as well as the character spelling accuracies achieved by OCLNN and the accuracies achieved by CCNN, BN3, CNN-R, and Bostanov [Bos04] for Dataset II. ESVM is the champion spelling method of BCI Competition III - Data set II. Bostanov is the champion spelling method of BCI Competition II - Data set IIb.

Table 3.7, 3.8, and 3.9 show the character spelling accuracies of different methods on Dataset II, III-A and III-B, respectively. The first column in a table lists the different methods we compare. Each row provides the character spelling accuracy of a method calculated by Equation (2.33) for different epoch numbers $k \in [1, 15]$. An accuracy number in bold indicates the highest accuracy along a column. "–" in a table means that the accuracy is not reported in the reference paper describing the corresponding method.

The goal of the aforementioned competitions (BCI Competition III and Competition II) is to compare which method is able to achieve the highest character spelling accuracy using all epochs (i.e., $k = 15$). For this goal, OCLNN is able to achieve the highest character spelling accuracy using all epochs for all three datasets in the two competitions. For Dataset III-A and III-B, OCLNN achieves 99% and 98% spelling accuracy for epoch number $k = 15$. For Dataset II, OCLNN only needs 3 epochs to achieve 100% spelling accuracy.

We also analyse the character spelling accuracies achieved by different methods for every epoch number $k \in [1, 15]$. Overall, in most cases, OCLNN achieves better

Table 3.7: Spelling accuracy achieved by different methods on Dataset II.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**77.42**	**90.32**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
CCNN	58.06	54.83	77.41	93.54	93.54	93.54	93.54	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**
CNN-R	70.97	83.87	93.55	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
BN3	**77.42**	74.19	80.65	83.87	93.55	96.77	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Bostanov	64.52	83.87	93.55	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 3.8: Spelling accuracy achieved by different methods on Dataset III-A.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**23**	**39**	56	63	**73**	**79**	**82**	**85**	**90**	**91**	**94**	**95**	**95**	**96**	**99**
CCNN	16	33	47	52	61	65	77	78	85	86	90	91	91	93	97
CNN-R	14	28	38	53	57	62	71	75	77	82	89	87	87	92	95
BN3	22	**39**	**58**	**67**	**73**	75	79	81	82	86	89	92	94	**96**	98
ESVM	16	32	52	60	72	–	–	–	–	83	–	–	94	–	97

Table 3.9: Spelling accuracy achieved by different methods on Dataset III-B.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	46	**62**	**72**	**79**	**84**	**87**	**89**	**93**	**94**	**96**	**97**	**97**	**97**	**98**	**98**
CCNN	35	52	59	68	79	81	82	89	92	91	91	90	91	92	92
CNN-R	36	46	66	70	77	80	86	86	88	91	94	95	95	96	96
BN3	**47**	59	70	73	76	82	84	91	**94**	95	95	95	94	94	95
ESVM	35	53	62	68	75	–	–	–	–	91	–	–	96	–	96

accuracies than the other methods. OCLNN increases the character spelling accuracies achieved by the other methods by up to 35.49%.

For Dataset II, OCLNN achieves the highest character spelling accuracies for every epoch number $k \in [1, 15]$ among all methods. Compared with the accuracies achieved by CCNN, CNN-R, BN3, and Bostanov, our OCLNN increases the accuracies with up to 35.49%, 6.45%, 19.35%, and 12.90%, respectively.

For Dataset III-A, when compared with methods CCNN, CNN-R, and ESVM, our OCLNN achieves the highest character spelling accuracies for every epoch number

$k \in [1, 15]$ among all the methods. OCLNN increases the accuracies by up to 14%, 12%, 18%, and 8% compared with the accuracies achieved by CCNN, CNN-R, and ESVM, respectively.

For Dataset III-B, when compared with methods CCNN, CNN-R, and ESVM, our OCLNN achieves the highest character spelling accuracies for every epoch number $k \in [1, 15]$ among all the methods. OCLNN increases the accuracies by up to 13%, 15%, 16%, and 11% compared with the accuracies achieved by CCNN, CNN-R, and ESVM, respectively.

When compared with BN3 on Dataset III-A and III-B, our OCLNN increases the accuracies by up to 8% considering epoch numbers 1, 2, and 5 to 15 on Dataset III-A as well as OCLNN increases the accuracies by up to 8% considering epoch numbers 2 to 15 on Dataset III-B. However, OCLNN decreases the accuracies for epoch numbers 3 and 4 on Dataset III-A and for epoch number 1 on Dataset III-B. This is because BN3 uses the Batch Normalization operation to improve the accuracies on smaller epoch numbers [LWG+18]. However, the Batch Normalization operation used in BN3 can only improve the accuracies on Dataset III-A and III-B. On Dataset II, BN3 achieves much worse results on smaller epoch numbers. In OCLNN, we do not use the Batch Normalization operation because we aim at a CNN with better potential to achieve higher accuracies across different datasets obtained from different subjects. The Batch Normalization operation is not very helpful to our OCLNN because it is more useful in deep CNNs [IS15] but our network has only 2 layers while BN3 has 5 layers. We have done experiments to obtain the spelling accuracy achieved by OCLNN when OCLNN uses and does not use the Batch Normalization operation. These experimental results are shown in Table 3.10, 3.11, and 3.12, where OCLNN-BN denotes that our OCLNN uses the Batch Normalization operation. Table 3.12 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is increased on epoch number k=1, 2, 3, 4, and 6. Unfortunately, Table 3.10 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is decreased on epoch number k= 3 and 4. Table 3.11 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is decreased on epoch number k= 8, 9, 11, 12, 13, and 15. These experimental results show that when used in our OCLNN, the Batch Normalization operation impairs the accuracies on Dataset II and III-A and only increases the accuracies on Dataset III-B. Therefore, in order to achieve higher accuracies across all three datasets obtained from different subjects, we abandon the Batch Normalization operation for our OCLNN.

3.3.5 Information Transfer Rate

This section compares the communication speed, i.e., Information Transfer Rate (ITR), of the P300 speller based on our OCLNN and other methods. ITR is calculated using

Table 3.10: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset II.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**77.42**	**90.32**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
OCLNN-BN	**77.42**	**90.32**	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 3.11: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset III-A.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**23**	**39**	**56**	**63**	**73**	**79**	**82**	**85**	**90**	**91**	**94**	**95**	**95**	**96**	**99**
OCLNN-BN	**23**	**39**	**56**	**63**	**73**	**79**	**82**	84	88	**91**	93	93	93	**96**	98

Table 3.12: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset III-B.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	46	62	72	79	**84**	87	**89**	**93**	**94**	**96**	**97**	**97**	**97**	**98**	**98**
OCLNN-BN	**48**	**64**	**73**	**80**	**84**	**88**	**89**	**93**	**94**	**96**	**97**	**97**	**97**	**98**	**98**

Equation (2.34) and (2.35) (introduced in Section 2.4.3). The ITR of the P300 speller based on our OCLNN and other methods for Dataset II, Dataset III-A and Dataset III-B is shown in Table 3.13, 3.14, and 3.15, respectively. In these tables, the different methods, we compare, are shown in the first column. The ITR for different epoch numbers $k \in [1, 15]$ is shown in each row of a table. A number in bold denotes that the number is the highest ITR along a row. "–" in a table denotes that the ITR cannot be calculated because the corresponding paper, describing the method, does not provide the spelling accuracy. The ITR is shown in bits/minute.

In the context of ITR, i.e, the communication speed of the P300 speller, we compare the maximum ITR achieved by each method because the maximum ITR represents the maximum communication speed achieved by a method. We call this maximum ITR max-ITR. Overall, our OCLNN achieves comparable max-ITR with BN3 and higher max-ITR than the other related methods, i.e., CCNN, CNN-R, Bostanov, and ESVM. Our OCLNN can increase the max-ITR achieved by CCNN, CNN-R, Bostanov, and ESVM with up to 15.7 bits/min.

For Dataset II, shown in Table 3.13, when compared with the max-ITR achieved

Table 3.13: The ITR of the P300 speller based on different methods on Dataset II.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**42.28**	37.74	35.25	28.46	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
CCNN	**26.58**	16.65	22.09	24.73	20.74	17.85	15.67	14.92	13.45	13.2	12.12	11.2	10.41	9.72	9.12
CNN-R	**36.68**	33.18	30.64	26.41	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
BN3	**42.28**	27.06	23.65	20.4	20.74	19.07	16.74	14.92	14.5	13.2	12.12	11.2	10.41	9.72	9.12
Bostanov	31.46	**33.18**	30.64	26.41	22.15	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12

Table 3.14: The ITR of the P300 speller based on different methods on Dataset III-A.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	5.77	9.64	13.11	12.78	**13.59**	13.32	12.44	11.78	11.74	10.91	10.63	10.02	9.32	8.88	8.89
CCNN	2.96	7.33	9.91	9.41	10.18	9.7	**11.21**	10.2	10.63	9.87	9.82	9.25	8.6	8.36	8.51
CNN-R	2.28	5.56	7.03	9.7	9.13	8.99	**9.82**	9.56	9.01	9.11	9.62	8.55	7.94	8.2	8.17
BN3	5.33	9.64	13.87	**14.1**	13.59	12.22	11.69	10.86	10	9.87	9.62	9.44	9.13	8.88	8.69
ESVM	2.96	6.96	11.65	11.82	**13.28**	–	–	–	–	9.29	–	–	9.13	–	8.51

Table 3.15: The ITR of the P300 speller based on different methods on Dataset III-B.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	18.32	**20.26**	19.62	18.45	17.15	15.68	14.32	13.82	12.71	12.06	11.3	10.44	9.71	9.27	8.69
CCNN	11.76	15.3	14.25	14.44	**15.47**	13.88	12.44	12.76	12.22	10.91	10.01	9.07	8.6	8.2	7.69
CNN-R	12.32	12.58	**17.05**	15.14	14.83	13.6	13.49	12.02	11.29	10.91	10.63	10.02	9.32	8.88	8.33
BN3	**18.97**	18.72	18.75	16.2	14.51	14.17	12.96	13.28	12.71	11.81	10.84	10.02	9.13	8.53	8.17
ESVM	11.76	**15.78**	15.43	14.44	14.2	–	–	–	–	10.91	–	–	9.51	–	8.33

by CCNN, CNN-R, and Bostanov, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and Bostanov with 15.7 bits/min, 5.6 bits/min, and 9.1 bits/min, respectively.

For Dataset III-A, shown in Table 3.14, when compared with the max-ITR achieved by CCNN, CNN-R, and ESVM, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and ESVM with 2.38 bits/min, 3.77 bits/min, and 0.31 bits/min, respectively.

For Dataset III-B, shown in Table 3.15, when compared with the max-ITR achieved by CCNN, CNN-R and ESVM, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and ESVM with 4.79 bits/min and 3.21 bits/min, and 4.48 bits/min, respectively.

When compared with BN3, on Dataset III-B, our OCLNN increases the max-ITR achieved by BN3 with 1.29 bits/min. On Dataset II, our OCLNN achieves the same max-ITR with BN3. However, on Dataset III-A, our OCLNN decreases the max-ITR achieved by BN3 with 0.51 bits/min. The reason is the following. On Dataset III-A, BN3 achieves the max-ITR 14.1 bits/min on epoch number $k = 4$ (see Table 3.14). On epoch number $k = 4$, our OCLNN achieves lower spelling accuracy than BN3 (see Table 3.8). The reason for this has been explained in the last paragraph in Section 3.3.4. Thus, our OCLNN cannot achieve the same max-ITR as BN3 does on Dataset III-A. The discussion above shows that overall, our OCLNN achieves comparable max-ITR with BN3. On the other hand, considering that the complexity of our OCLNN is much lower compared to the complexity of BN3 (see Table 3.5 in Section 3.3.2), our OCLNN could be considered better than BN3 for the P300 speller.

In Section 2.4.3, we have shown that when using Dataset II, III-A, and III-B, the theoretically achievable maximum ITR is 67.43 bits/min. The maximum ITR (i.e., max-ITR) achieved by our OCLNN is 42.28 bits/min, 13.59 bits/min, and 20.26 bits/min on Dataset II, III-A, and III-B, respectively. The maximum ITR achieved by our OCLNN still cannot reach the theoretically achievable maximum ITR. Thus, further research efforts are needed to find approaches to increase the maximum ITR of a P300 speller in order to bring it closer to the theoretically achievable maximum ITR.

3.4 Conclusions

In this chapter, we propose a simple CNN, called OCLNN, for P300 signal detection and character spelling in the P300 speller. Our CNN learns P300-related features better by performing both spatial convolution and temporal convolution in the first layer. Compared with the state-of-the-art CNNs for P300 signal detection, our CNN has only two layers and much smaller number of parameters, which reduces the complexity significantly. Experimental results on three datasets show that our CNN always increases the P300 signal detection accuracy and increases the character spelling accuracy in most cases, when compared with the state-of-the-art methods for P300 signal detection and character spelling. In terms of the communication speed, our OCLNN achieves comparable communication speed with BN3 and higher communication speed than the other state-of-art methods.

Chapter 4

Ensemble of Convolutional Neural Networks for P300 Signal Detection and Character Spelling

Hongchang Shan, Yu Liu, and Todor Stefanov,
"Ensemble of Convolutional Neural Networks for P300 Speller in Brain Computer Interface,"
In *Proceedings of the 28th International Conference on Artificial Neural Networks.*, pp., Munich, Germany, September 17-19, 2019.

In Chapter 3, we introduce our simple, yet effective OCLNN for the P300 signal detection and character spelling. This CNN solves the problems of the state-of-the-art CNNs for the P300 speller in [CG11, MG15, LWG+18] by performing the spatial convolution and temporal convolution in its first layer using raw signals. Unfortunately, OCLNN still has some limitations to extract some relevant and important features related to P300 signals. OCLNN performs the spatial convolution and the temporal convolution together, thereby realizing a joint spatial-temporal convolution in the first layer. This spatial-temporal convolution extracts only P300-related joint spatial-temporal features in its single convolution layer. OCLNN does not extract P300-related separate temporal features and separate spatial features. These separate temporal features and separate spatial features have proven to be very important for the P300 speller [FTM+88, Pol07, PNCB11, HVE06]. Adding several temporal or spatial convolution layers following the first spatial-temporal convolution layer of OCLNN is a potential method to enable OCLNN to learn P300-related separate spatial or separate temporal features. Nevertheless, such method cannot learn well P300-related separate temporal or spatial features due to the loss of raw information. The raw information loss happens because the input to these additional temporal or spatial convolution layers for OCLNN is the abstract signals generated by the first spatial-temporal convolution layer instead of raw signals.

In order to solve this problem of OCLNN, we proposes a novel network, which combines OCLNN with two other novel CNNs, we have devised, in order to learn well the aforementioned P300-related separate spatial and separate temporal features, which are not extracted by OCLNN, together with the spatial-temporal features extracted by OCLNN. The novel contributions of this chapter are the following:

- Each of the two novel CNNs has only one convolution layer. One of the novel CNNs performs only the temporal convolution in its convolution layer (the first layer) to learn P300-related separate temporal features. The other novel CNN performs only the spatial convolution in its convolution layer (the first layer) to learn P300-related separate spatial features. These two novel CNNs are able to learn well P300-related separate temporal and separate spatial features because the input to each of the two novel CNNs is raw signals. In addition, we propose a novel network which is an ensemble of these two novel CNNs and OCLNN. This network extracts more useful P300-related features than OCLNN alone and is able to achieve higher P300 signal detection accuracy and character spelling accuracy than OCLNN.

- Experimental results on three benchmark datasets show that our proposed ensemble of CNNs is able to increase the P300 signal detection accuracy, the character spelling accuracy, and the communication speed achieved by OCLNN with up to 4.32%, 5%, and 6.05 bits/min, respectively. Also, our proposed ensemble of CNNs outperforms other related methods with a significant P300 signal detection accuracy improvement up to 18.55%, a significant character spelling accuracy improvement up to 38.72%, and a significant communication speed improvement up to 21.75 bits/min. In terms of the network complexity, the complexity of our proposed ensemble of CNNs is lower than the complexity of the CNN in [MG15], and higher than the complexity of OCLNN and the CNNs in [CG11, LWG$^+$18].

The rest of the chapter is organized as follows: Section 4.1 presents our proposed network (ensemble of CNNs) for the P300 speller. Section 4.2 compares the complexity, the P300 signal detection accuacy, the character spelling accuracy, and the communication speed between our network and other related methods for the P300 speller. Section 4.3 analyzes our proposed two novel CNNs on extracting P300-related features, performs an ablation study on our proposed network, and discusses the importance of extracting P300-related features from raw signals. Section 4.4 ends this chapter with conclusions.

4.1 Proposed Network

This section introduces our proposed network for the P300 signal detection and character spelling in the P300 speller. We call our proposed network Ensemble of Convolutional Neural Networks (EoCNN). EoCNN combines two novel CNNs, we have devised, together with our proposed OCLNN presented in Chapter 3. We call these two novel CNNs as follows: One Spatial Layer Network (OSLN) and One Temporal Layer Network (OTLN).

4.1.1 Ensemble of Convolutional Neural Networks

The workflow of our EoCNN is shown in Figure 4.1. First, the EEG signals are preprocessed to construct the input tensor. For the details of the construction of the input tensor, please refer to Section 3.2.1. Then, the input tensor is sent to three different CNNs, i.e., OSLN, OTLN, and OCLNN. OSLN and OTLN are described in Section 4.1.2. OCLNN is our proposed simple CNN presented in Chapter 3. OSLN extracts P300-related separate spatial features. OTLN extracts P300-related separate temporal features. OCLNN extracts P300-related joint spatial-temporal features. Our EoCNN uses the ensemble of the outputs from OSLN, OTLN, and OCLNN for the P300 signal detection and character spelling in the P300 speller. The detection of P300 signals and the inference of characters by using EoCNN is introduced in Section 4.1.4.

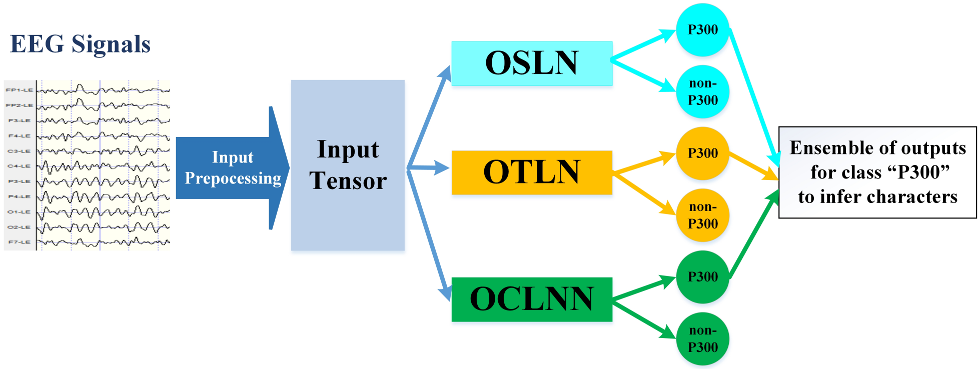


Figure 4.1: Workflow of our EoCNN

4.1.2 Proposed OSLN and OTLN

The architectures of our proposed OSLN and OTLN are described in Table 4.1 and Table 4.2, respectively. OSLN and OTLN are used in EoCNN (see Section 4.1.1), where OSLN is designed to learn P300-related separate spatial features and OTLN is designed to learn P300-related separate temporal features. Since only the convolution layer is different between OSLN and OTLN, below we describe the architectures of OSLN and OTLN together.

Table 4.1: OSLN architecture.

Layer	Operation	Kernel	Feature Maps or Neurons
1	Convolution Dropout	$(1,C)$ —	16 —
Output	Fully-Connected	—	2

Table 4.2: OTLN architecture.

Layer	Operation	Kernel	Feature Maps or Neurons
1	Convolution Dropout	$(N/15, 1)$ —	16 —
Output	Fully-Connected	—	2

Layer 1 of OSLN (see Table 4.1) performs the spatial convolution operation with the kernel size $(1, C)$. This convolution operation converts each receptive field of the signal samples into an abstract datum in a feature map. The signal samples in each receptive field are from all C sensors in the space domain and sampled at only one time point in the time domain. Therefore, this convolution operation extracts P300-related separate spatial features. We use the kernel size $(1, C)$ in order to make this layer to learn the spatial features from EEG signals acquired using all sensors. The reason for using all sensors is that it is more helpful to increase the spelling accuracy than using only part of all sensors [CG11, MG15, LWG+18, SLS18]. The input to this layer is raw signals, so this layer learns P300-related separate spatial features from raw signals. This layer generates 16 feature maps, which are the input to Layer Output of OSLN.

Layer 1 of OTLN (see Table 4.2) performs the temporal convolution operation with the kernel size $(N/15, 1)$. The temporal convolution operation converts each receptive field of the signal samples into an abstract datum in a feature map. The signal samples in each receptive field are sampled within a certain time period and are acquired from only one sensor. Therefore, this convolution operation extracts

P300-related separate temporal features. We use the kernel size (N/15, 1) because Chapter 3 has shown that 1/15 of the temporal signal samples is a proper receptive field for a CNN to learn P300-related temporal features. The input to this layer is raw signals, so this layer learns P300-related separate temporal features from raw signals. This layer generates 16 feature maps, which are the input to Layer Output of OTLN.

In both Layer 1 of OSLN and Layer 1 of OTLN, the activation function is the Rectified Linear Unit (ReLU) function. We employ Dropout [SHK$^+$14], with a rate of 0.4, to prevent OSLN and OTLN from overfitting (introduced in Section 2.2.3).

Layer Output of OSLN (see Table 4.1) and Layer Output of OTLN (see Table 4.2) are the same. Layer Output is a fully-connected layer with two neurons. These two neurons represent the class “P300” (the presence of a P300 signal) and the class “non-P300” (the absence of a P300 signal), respectively. The activation function used in this layer is the Softmax function which outputs the predicted probability for the “P300” class and the “non-P300” class.

OSLN and OTLN each uses only one convolution layer. OSLN uses only one convolution layer because it does not make sense to add more spatial convolution layers for OSLN. This CNN is designed to only learn P300-related spatial features from the EEG signals recorded with all C sensors in the first layer. If we add more spatial convolution layers after its first spatial convolution layer to learn P300-related spatial features, these added layers should learn spatial features from the abstract signals generated by the first spatial convolution layer. However, these abstract signals include only the time domain and do not have the space domain because the first convolution layer uses a receptive field including all C sensors. Thus, these abstract signals cannot be used to extract further spatial features. OTLN also uses only one convolution layer because one convolution layer is enough to extract useful P300-related separate temporal features (as shown and discussed later in Section 4.3.1).

4.1.3 Training

The training process used for our EoCNN is the same as the training process used for our OCLNN (proposed in Chapter 3). For the details of the training process, please refer to Section 3.2.3.

4.1.4 P300 Signal Detection and Character Spelling using EoCNN

We use the outputs of the Softmax function for class "P300" and class "non-P300" in Layer Output of OSLN, OTLN, and OCLNN to detect P300 signals and infer characters. For epoch i and for intensification j we use $P_{OS}^{1}(i,j)$ to denote the output of the Softmax function for class "P300" in OSLN, $P_{OS}^{0}(i,j)$ to denote the output of the Softmax function for class "non-P300" in OSLN, $P_{OT}^{1}(i,j)$ to denote the output of the Softmax function for class "P300" in OTLN, $P^0_{OT}(i,j)$ to denote the output of the Softmax function for class "non-P300" in OTLN, $P^1_{OCL}(i,j)$ to denote the output of the Softmax function for class "P300" in OCLNN, $P^0_{OCL}(i,j)$ to denote the output of the Softmax function for class "non-P300" in OCLNN. Therefore, for epoch i and for intensification j, $P^1_{OS}(i,j)$ denotes the predicted probability by OSLN for class “P300”; $P^0_{OS}(i,j)$ denotes the predicted probability by OSLN for class “non-P300”; $P^1_{OT}(i,j)$ denotes the predicted probability by OTLN for class “P300”; $P^0_{OT}(i,j)$ denotes the predicted probability by OTLN for class “non-P300”; $P^1_{OCL}(i,j)$ denotes the predicted probability by OCLNN for class “P300”; and $P^0_{OCL}(i,j)$ denotes the predicted probability by OCLNN for class “non-P300”.

We use Equation (4.1), (4.2), and (4.3) for the detection of P300 signals, where $P^1_{EoC}(i,j)$ denotes the predicted probability by EoCNN for class “P300”; $P^0_{EoC}(i,j)$ denotes the predicted probability by EoCNN for class “non-P300”; EoC denotes our EoCNN classifier; and $X_{(i,j)}$ denotes the input tensor to be classified. Equation (4.1) and (4.2) show the ensemble processing of the outputs from OSLN, OTLN, and OCLNN. Equation (4.3) shows the detection of a P300 signal. In this equation, $EoC(X_{(i,j)}) = 1$ means that EoCNN detects a P300 signal from the input tensor $X_{(i,j)}$ and $EoC(X_{(i,j)}) = 0$ means that EoCNN does not detect a P300 signal from this input tensor. After using Equation (4.3) to detect P300 signals, we can assess the performance of our proposed EoCNN in terms of the P300 signal detection accuracy (see Section 4.2.2).

$$P^1_{EoC}(i,j) = \frac{1}{3} \times (P^1_{OS}(i,j) + P^1_{OT}(i,j) + P^1_{OCL}(i,j)) \tag{4.1}$$

$$P^0_{EoC}(i,j) = \frac{1}{3} \times (P^0_{OS}(i,j) + P^0_{OT}(i,j) + P^0_{OCL}(i,j)) \tag{4.2}$$

$$EoC(X_{(i,j)}) = \begin{cases} 1 & if \quad P^1_{EoC}(i,j) > P^0_{EoC}(i,j) \\ 0 & otherwise \end{cases} \tag{4.3}$$

We use $P^1_{EoC}(i,j)$, the output of EoCNN for class “P300”, to calculate the position of the target character in the P300 speller character matrix. For the detailed calculation process, please refer to Section 2.4.2, Equation (2.30), (2.31), and (2.32).

4.2 Experimental Evaluation

The experimental setup used for the evaluation, presented in this section, is the same as the experimental setup used in Chapter 3 (for detailed description of the experimental setup please see Section 3.3.1). We first compare the complexity of our EoCNN with the complexity of other related CNNs for the P300 speller in Section 4.2.1. Then, we compare the P300 signal detection accuracy achieved by our EoCNN and other related methods in Section 4.2.2. Also, we compare the character spelling accuracy achieved by our EoCNN and other related methods in Section 4.2.3. Finally, we compare the ITR of the P300 speller based on our EoCNN and other related methods in Section 4.2.4.

4.2.1 Complexity

In this section, we compare the complexity of EoCNN, in terms of the number of parameters (explained in Section 3.3.2) and layers, with the complexity of the networks OCLNN (proposed and presented in Chapter 3), CCNN [CG11], BN3 [LWG+18], and CNN-R [MG15] briefly described in Section 3.1. Concerning the complexity of EoCNN, since EoCNN is the ensemble of OSLN, OTLN, and OCLNN, the number of the parameters of EoCNN is calculated as the sum of the number of parameters of OSLN, OTLN, and OCLNN, and the number of the layers used in EoCNN is calculated as the sum of the number of the layers used in OSLN, OTLN, and OCLNN. The complexity of different CNNs is shown in Table 4.3. The first row in the table lists the CNNs, we compare. The second row provides the number of parameters for each CNN. The third row shows the number of layers used in each CNN. Table 4.3 shows that the complexity of EoCNN, in terms of the number of parameters and layers, is higher than the complexity of OCLNN, CCNN, BN3 and lower than the complexity of CNN-R.

Table 4.3: Complexity of different CNNs.

	EoCNN	OCLNN	CCNN	BN3	CNN-R
Parameters	56598	16882	37502	39489	21950818
Layers	6	2	4	5	6

4.2.2 P300 Signal Detection Accuracy

This section compares the P300 signal detection accuracies achieved by EoCNN with the P300 signal detection accuracies achieved by OCLNN (proposed and presented in Chapter 3), CCNN [CG11], BN3 [LWG+18], and CNN-R [MG15] on Dataset II, III-A and III-B.

The P300 signal detection accuracies are shown in Table 4.4. The first row in the table lists the CNNs used for comparison. The second, third, and last row show the P300 signal detection accuracy of the different CNNs on Dataset II, III-A, and III-B, respectively. The numbers are given in percentage (%) and calculated using Equation (3.5). An accuracy number in bold indicates the highest accuracy along a row. “–” in the table means that the accuracy is not reported in the reference paper describing the corresponding CNN.

Table 4.4: P300 signal detection accuracy of different CNNs on Dataset II, III-A, and III-B.

	EoCNN	OCLNN	CCNN	BN3	CNN-R
P300 Accuracy on II	**94.76**	92.41	–	84.44	86.29
P300 Accuracy on III-A	**88.92**	84.60	70.37	75.13	73.06
P300 Accuracy on III-B	**89.65**	86.40	78.19	79.02	79.80

Overall, EoCNN achieves the highest accuracy among all CNNs on Dataset II, III-A, and III-B. It increases the P300 signal detection accuracies achieved by the other CNNs with up to 18.55%. Compared with OCLNN, EoCNN is able to increase the P300 signal detection accuracy achieved by OCLNN with 2.35%, 4.32%, and 3.25% on Dataset II, Dataset III-A, and Dataset III-B, respectively. Compared with the other related CNNs, on Dataset II, EoCNN is able to increase the P300 signal detection accuracy achieved by BN3 and CNN-R with 10.32% and 8.47%, respectively. On Dataset III-A, EoCNN is able to increase the P300 signal detection accuracy achieved by CCNN, BN3, and CNN-R with 18.55%, 13.79%, and 15.86%, respectively. On Dataset III-B, EoCNN is able to increase the P300 signal detection accuracy achieved by CCNN, BN3, and CNN-R with 11.46%, 10.63%, and 9.85%, respectively.

4.2.3 Character Spelling Accuracy

This section compares the character spelling accuracies achieved by our EoCNN and the accuracies achieved by OCLNN, CCNN, BN3, CNN-R, and ESVM [RG08] for Dataset III-A and III-B, as well as the character spelling accuracies achieved by EoCNN and the accuracies achieved by OCLNN, CCNN, BN3, CNN-R, and Bostanov [Bos04] for Dataset II.

The character spelling accuracy achieved by our EoCNN and other methods on Dataset II, Dataset III-A, and Dataset III-B is shown in Table 4.5, 4.6, and 4.7, respectively. In these tables, the different methods, we compare, are shown in the first column. The spelling accuracy for different epoch numbers $k \in [1, 15]$ is shown in each row of the table. A number in bold indicates that the accuracy achieved by the corresponding method is the highest among all methods. “–” denotes that the corresponding paper, describing the method, does not provide the accuracy number. The accuracy numbers in these tables are given in percentage (%). Overall, the spelling accuracy achieved by our EoCNN is higher than the spelling accuracy achieved by other methods in most cases. Our EoCNN increases the spelling accuracy achieved by other methods with up to 38.72%.

Table 4.5: Spelling accuracy achieved by different methods on Dataset II.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	**83.87**	**93.55**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
CCNN	58.06	54.83	77.41	93.54	93.54	93.54	93.54	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**
CNN-R	70.97	83.87	93.55	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
BN3	77.42	74.19	80.65	83.87	93.55	96.77	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**
OCLNN	77.42	90.32	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Bostanov	64.52	83.87	93.55	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 4.6: Spelling accuracy achieved by different methods on Dataset III-A.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	**23**	**39**	**61**	**68**	**76**	**81**	**84**	**86**	88	**93**	**95**	**98**	**97**	**99**	**99**
CCNN	16	33	47	52	61	65	77	78	85	86	90	91	91	93	97
CNN-R	14	28	38	53	57	62	71	75	77	82	89	87	87	92	95
BN3	22	**39**	58	67	73	75	79	81	82	86	89	92	94	96	98
OCLNN	**23**	**39**	56	63	73	79	82	85	**90**	91	94	95	95	96	**99**
ESVM	16	32	52	60	72	–	–	–	–	83	–	–	94	–	97

Table 4.5, 4.6, and 4.7 show that, when compared with OCLNN (proposed and presented in Chapter 3), the spelling accuracy achieved by EoCNN is higher than the spelling accuracy achieved by OCLNN in most cases. EoCNN is able to increase the character spelling accuracy achieved by OCLNN with up to 6.45%, 5%, 5% for Dataset II, Dataset III-A, and Dataset III-B, respectively. However, on epoch number $k = 9$ in Dataset III-A and on epoch number $k = 8$ in Dataset III-B, EoCNN decreases the spelling accuracy achieved by OCLNN. The reason for this is that EoCNN puts equal importance on OSLN, OTLN, and OCLNN in the ensemble processing of the

Table 4.7: Spelling accuracy achieved by different methods on Dataset III-B.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	**51**	**66**	**74**	**81**	**84**	**90**	**91**	92	**95**	**97**	**98**	**98**	**98**	**98**	**99**
CCNN	35	52	59	68	79	81	82	89	92	91	91	90	91	92	92
CNN-R	36	46	66	70	77	80	86	86	88	91	94	95	95	96	96
BN3	47	59	70	73	76	82	84	91	94	95	95	95	94	94	95
OCLNN	46	62	72	79	**84**	87	89	**93**	94	96	97	97	97	**98**	98
ESVM	35	53	62	68	75	–	–	–	–	91	–	–	96	–	96

outputs from OSLN, OTLN, and OCLNN. For more details of the explaination on this reason please refer to Chapter 6.

Table 4.5, 4.6, and 4.7 also show that, when compared with other related methods, for Dataset II, our EoCNN can increase the spelling accuracy achieved by CCNN, CNN-R, BN3, and Bostanov with up to 38.72%, 12.90%, 19.36%, and 19.35%, respectively. For Dataset III-A, our EoCNN can increase the spelling accuracy achieved by CCNN, CNN-R, BN3, and ESVM with up to 16%, 23%, 7%, and 10%, respectively. For Dataset III-B, our EoCNN can increase the accuracy achieved by CCNN, CNN-R, BN3, and ESVM with up to 16%, 20%, 8%, and 16%, respectively.

Moreover, our method is robust across different subjects. Table 4.5, 4.6, and 4.7 show that for all three subjects, our EoCNN achieves the highest spelling accuracy among all other methods in 43 out of 45 cases.

These experimental results also give some insights on how many epochs we should use for the spelling of one character in the P300 speller. The first insight is from the fact that, in Table 4.5, the spelling accuracy achieved by CCNN and BN3 on epoch number k=2 is lower than the spelling accuracy achieved by CCNN and BN3 on epoch number k=1. This shows that adding more epochs does not necessarily improve the spelling accuracy for the P300 speller. Such observation is also discussed in more details in [CG11]. The other insight is from the fact that in Dataset II, we need only 2 epochs to achieve a spelling accuracy which is higher than 90% while in Dataset III-A and Dataset III-B, in order to achieve a spelling accuracy higher than 90%, we need at least 10 epochs and 6 epochs, respectively. This indicates that we can use different number of epochs for different subjects to spell characters using the P300 speller. In this way, we can use a small number of epochs for a subject when using the P300 speller such that we can significantly decrease the time needed for a subject to spell a character while keeping an acceptable spelling accuracy.

4.2.4 Information Transfer Rate

This section compares the Information Transfer Rate (ITR) of the P300 speller based on our EoCNN and other methods. ITR is calculated using Equation (2.34) and (2.35) (introduced in Section 2.4.3). The ITR of the P300 speller based on our EoCNN and other methods for Dataset II, Dataset III-A, and Dataset III-B is shown in Table 4.8, 4.9, and 4.10, respectively. In these tables, the different methods, we compare, are shown in the first column. The ITR for different epoch numbers $k \in [1, 15]$ is shown in each row of the table. A number in bold denotes that the number is the highest ITR along a row. "–" in a table denotes that the ITR cannot be calculated because the corresponding method does not provide the spelling accuracy. The ITR is shown in bits/minute.

Table 4.8: The ITR of the P300 speller based on different methods on Dataset II.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	**48.33**	40.25	35.25	28.46	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
CCNN	**26.58**	16.65	22.09	24.73	20.74	17.85	15.67	14.92	13.45	13.2	12.12	11.2	10.41	9.72	9.12
CNN-R	**36.68**	33.18	30.64	26.41	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
BN3	**42.28**	27.06	23.65	20.4	20.74	19.07	16.74	14.92	14.5	13.2	12.12	11.2	10.41	9.72	9.12
OCLNN	**42.28**	37.74	35.25	28.46	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
Bostanov	31.46	**33.18**	30.64	26.41	22.15	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12

Table 4.9: The ITR of the P300 speller based on different methods on Dataset III-A.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	5.77	9.64	**15.03**	14.44	14.51	13.88	12.96	12.02	11.29	11.35	10.84	10.67	9.71	9.48	8.89
CCNN	2.96	7.33	9.91	9.41	10.18	9.7	**11.21**	10.2	10.63	9.87	9.82	9.25	8.6	8.36	8.51
CNN-R	2.28	5.56	7.03	9.7	9.13	8.99	**9.82**	9.56	9.01	9.11	9.62	8.55	7.94	8.2	8.17
BN3	5.33	9.64	13.87	**14.1**	13.59	12.22	11.69	10.86	10	9.87	9.62	9.44	9.13	8.88	8.69
OCLNN	5.77	9.64	13.11	12.78	**13.59**	13.32	12.44	11.78	11.74	10.91	10.63	10.02	9.32	8.88	8.89
ESVM	2.96	6.96	11.65	11.82	**13.28**	–	–	–	–	9.29	–	–	9.13	–	8.51

We compare the max-ITR[1] achieved by our EoCNN and other methods for the P300 speller. Overall, the max-ITR achieved by our EoCNN is higher than the max-ITR achieved by all other methods. Our EoCNN is able to increase the max-ITR achieved by other methods with up to 21.75 bits/min.

---

[1] The notion of max-ITR is introduced in Section 3.3.5.

Table 4.10: The ITR of the P300 speller based on different methods on Dataset III-B.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN	21.61	**22.4**	20.52	19.23	17.15	16.64	14.9	13.55	12.97	12.31	11.55	10.67	9.92	9.27	8.89
CCNN	11.76	15.3	14.25	14.44	**15.47**	13.88	12.44	12.76	12.22	10.91	10.01	9.07	8.6	8.2	7.69
CNN-R	12.32	12.58	**17.05**	15.14	14.83	13.6	13.49	12.02	11.29	10.91	10.63	10.02	9.32	8.88	8.33
BN3	**18.97**	18.72	18.75	16.2	14.51	14.17	12.96	13.28	12.71	11.81	10.84	10.02	9.13	8.53	8.17
OCLNN	18.32	**20.26**	19.62	18.45	17.15	15.68	14.32	13.82	12.71	12.06	11.3	10.44	9.71	9.27	8.69
ESVM	11.76	**15.78**	15.43	14.44	14.2	–	–	–	–	10.91	–	–	9.51	–	8.33

Table 4.8, 4.9, and 4.10 show that, when compared with OCLNN (proposed and presented in Chapter 3), the max-ITR achieved by our EoCNN is higher than the max-ITR achieved by our OCLNN on all three datasets. Our EoCNN increases the max-ITR achieved by our OCLNN with 6.05 bits/min, 1.44 bits/min, and 2.14 bits/min on Dataset II, III-A, and III-B, respectively.

Table 4.8, 4.9, and 4.10 also show that, when compared with other related methods for the P300 speller, for Dataset II, the max-ITR achieved by our EoCNN is higher than the max-ITR achieved by all other methods, i.e., CCNN, CNN-R, BN3, and Bostanov. Our EoCNN increases the max-ITR achieved by CCNN, CNN-R, BN3, and Bostanov with 21.75 bits/min, 11.65 bits/min, 6.05 bits/min, and 15.15 bits/min, respectively. For Dataset III-A, the max-ITR achieved by our EoCNN is higher than the max-ITR achieved by all other methods, i.e., CCNN, CNN-R, BN3, and ESVM. Our EoCNN increases the max-ITR achieved by CCNN, CNN-R, BN3, and ESVM with 3.82 bits/min, 5.21 bits/min, 0.93 bits/min, and 1.75 bits/min, respectively. For Dataset III-B, the max-ITR achieved by our EoCNN is higher than the max-ITR achieved by all other methods, i.e., CCNN, CNN-R, BN3, and ESVM. Our EoCNN increases the max-ITR achieved by CCNN, CNN-R, BN3, and ESVM with 6.93 bits/min, 5.35 bits/min, 3.43 bits/min, and 6.62 bits/min, respectively.

Our EoCNN increases the max-ITR achieved by our OCLNN, thereby bringing the max-ITR more closer to the theoretically achievable maximum ITR (introduced in Section 2.4.3). Unfortunately, the complexity, in terms of the number of parameters, of EoCNN is 3.35 times higher than the complexity of OCLNN (see Table 4.3). As described in Section 1.1.2, the low complexity of a CNN-based method for P300 character spelling is an important requirement to build efficient P300 spellers that can be used in people's daily life. Therefore, increasing further the complexity of our EoCNN-based P300 speller in order to further increase the max-ITR of the speller is not a suitable way to go in terms of efficiency. Thus, further research efforts are needed to find alternative ways to further increase the max-ITR without sacrificing the efficiency of the EoCNN-based P300 speller. For example, one possible alternative ways is to devise a better character matrix (Figure 2.10) which enables the reduction of the time periods t_1, t_2, and t_3 in the P300 speller experiment (see Section 2.4.2 and Equation (2.34) and (2.35)) . However, such psychology-related research direction is out of the scope of this thesis.

4.3 Discussions

In this section, first, we analyse our proposed OTLN and OSLN in terms of character spelling accuracy and discuss the influence of the number of convolution layers on extracting useful P300-related separate temporal features in Section 4.3.1. Then, we perform an ablation study on EoCNN to show that we need to combine all three CNNs (i.e., OSLN, OTLN, OCLNN) in EoCNN in order to achieve high spelling accuracy in Section 4.3.2. Finally, we explore the importance of extracting P300-related features from raw signals in Section 4.3.3.

In this section, all the experiments are performed by using the experimental setup described in Section 3.3.1. We draw similar conclusions from the experimental results of all datasets, i.e., Dataset III-A, Dataset III-B, and Dataset II. Thus, the experimental results are shown using only Dataset III-A in order to present our conclusions.

4.3.1 Analysis of Our Proposed OTLN and OSLN

First, we perform experiments to show the character spelling accuracy achieved by OTLN and OSLN, respectively. The experimental results are shown in Table 4.11. In this table, the different CNNs, we compare, are shown in the first column. The spelling accuracy for different epoch numbers $k \in [1, 15]$ is shown in each row of the table. A number in bold indicates that the corresponding CNN achieves the highest accuracy compared to all other CNNs. The accuracy numbers in this table are given in percentage (%). Table 4.11 shows that OTLN and OSLN both have good ability to achieve high spelling accuracy when OTLN and OSLN are used independently for P300 spelling. Thus, OTLN and OSLN are able to extract very useful P300-related separate temporal features and P300-related separate spatial features, respectively.

Then, we analyse whether OTLN needs more convolution layers to extract P300-related separate temporal features. In order to analyse the influence of the number of convolution layers on OTLN, we perform experiments to compare the spelling accuracy achieved by OTLN and other two CNNs called OTLN-3l and OTLN-6l. OTLN-3l and OTLN-6l use 3 and 6 convolution layers, respectively. These convolution layers use the same kernel size and generate the same number of feature maps as the convolution layer used in OTLN. The spelling accuracy achieved by OTLN, OTLN-3l and OTLN-6l is plotted in Figure 4.2. This figure shows that the spelling accuracy achieved by OTLN-3l and OTLN is almost the same. The spelling accuracy achieved by OTLN-6l is lower than the spelling accuracy achieved by OTLN. These experimental results show that using one convolution layer is enough to extract useful P300-related separate temporal features for P300 spelling. Using more convolution layers for the extraction of separate temporal features does not help increasing the spelling accuracy and may cause overfitting which decreases the spelling accuracy.

Table 4.11: Spelling accuracy achieved by OTLN, OSLN and EoCNN on Dataset III-A.

Network	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OTLN	21	34	51	65	69	73	76	81	85	85	88	92	92	93	95
OSLN	**24**	35	55	63	69	75	78	79	80	82	89	92	94	95	96
EoCNN	23	**39**	**61**	**68**	**76**	**81**	**84**	**86**	**88**	**93**	**95**	**98**	**97**	**99**	**99**

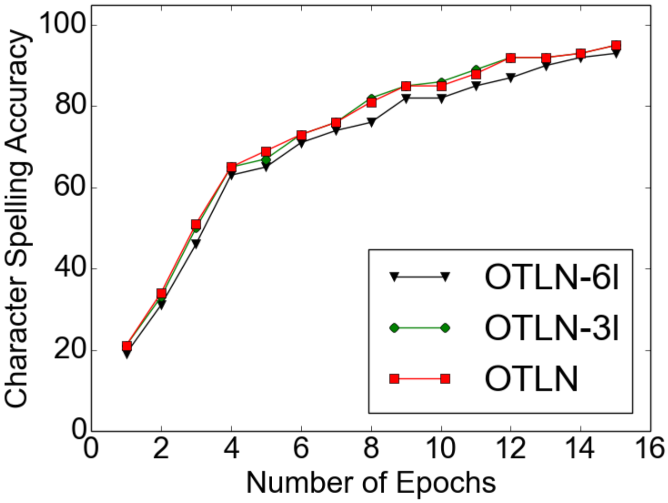


Figure 4.2: Spelling accuracy achieved by OTLN, OTLN-3l and OTLN-6l on Dataset III-A.

4.3.2 Ablation Study on EoCNN

We perform an ablation study on EoCNN to show that we need to combine all three CNNs (i.e., OSLN, OTLN, OCLNN) in EoCNN in order to achieve high spelling accuracy. We first remove a CNN from EoCNN. Then, we perform experiments to show the spelling accuracy achieved by the ensemble of the two CNNs left in EoCNN. In this way, we want to show the importance of each separate CNN in EoCNN for character spelling in the P300 speller. The experimental results are shown in Table 4.12. In this table, "-" indicates that we remove a given CNN from EoCNN. For example, "EoCNN-OSLN" indicates that we remove OSLN from EoCNN. The experimental results show that after removing any of the individual CNNs from EoCNN, the spelling accuracy achieved by the ensemble of the two CNNs left is lower compared with the spelling accuracy achieved by EoCNN when none of the individual CNNs is removed. This shows that we need to combine all three CNNs (i.e., OSLN, OTLN, OCLNN) in EoCNN in order to achieve high spelling accuracy. The experimental results from Table 4.12 also give us some insights. For example, in most cases, the spelling accuracy achieved by EoCNN-OTLN is higher than the spelling accuracy achieved by EoCNN-OSLN. This shows that P300-related spatial features are more important than P300-related temporal features on increasing the spelling accuracy. This is because a large number of sensors (i.e., 64 sensors) are used to acquire EEG signals in the P300 speller. When using a large number of sensors for the acquisition of EEG signals, we need to put more importance on extracting P300-related spatial features in order to achieve high spelling accuracy (For more explanation, please see Chapter 6).

Table 4.12: Spelling accuracy achieved by EoCNN after removing a separate CNN.

Network	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EoCNN-OTLN	**23**	**39**	58	67	75	**81**	82	**86**	86	91	93	96	96	97	**99**
EoCNN-OSLN	22	36	57	66	73	79	80	84	**89**	92	92	95	95	97	98
EoCNN-OCLNN	22	35	55	67	75	79	80	82	83	89	90	93	95	97	98
EoCNN	**23**	**39**	**61**	**68**	**76**	**81**	**84**	**86**	88	**93**	**95**	**98**	**97**	**99**	**99**

4.3.3 Exploration on the Importance of Extracting P300-related Features from Raw Signals

We explore the importance of extracting P300-related temporal features from raw signals. We consider two sets of networks. These two sets of networks are called "RAW_networks" and "unRAW_networks", respectively. RAW_networks include networks EoCNN, OCLNN, EoCNN-OSLN, EoCNN-OTLN, and EoCNN-OCLNN. All the networks in set RAW_networks extract P300-related temporal features from only raw signals. unRAW_networks include networks CCNN, CNN-R and BN3. All the networks in set unRAW_networks extract P300-related temporal features from abstract signals. We perform experiments to show the spelling accuracy achieved by each network in set RAW_networks and the spelling accuracy achieved by each network in set unRAW_networks.

The experimental results are shown in Figure 4.3. In this figure, the spelling accuracy achieved by the networks in set RAW_networks and the spelling accuracy achieved by the networks in set unRAW_networks are plotted in different shapes and colors. This figure shows that in most cases, the spelling accuracy achieved by the networks in set RAW_networks is higher than the spelling accuracy achieved by the networks in set unRAW_networks. This fact indicates that extracting P300-related temporal features from raw signals is able to achieve higher spelling accuracy than extracting P300-related temporal features from abstract signals.

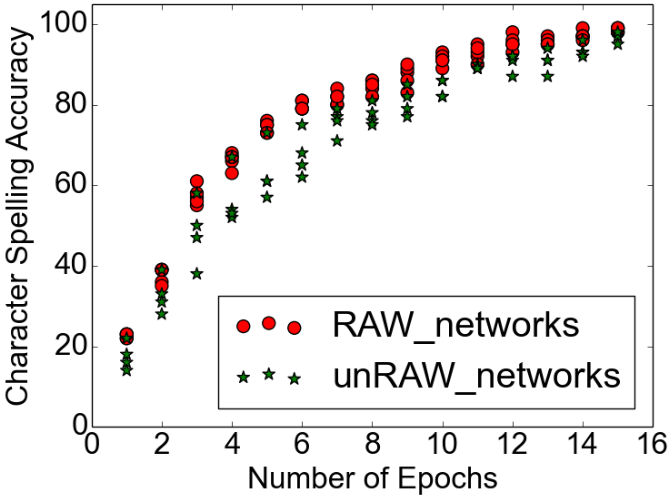


Figure 4.3: Spelling accuracy achieved by networks in set RAW_networks and networks in set unRAW_networks on Dataset III-A.

4.4 Conclusions

In this chapter, we propose a novel and effective network, called EoCNN, for the P300 signal detection and character spelling in the P300 speller. Our EoCNN uses an ensemble of three different CNNs for P300 spelling. These three CNNs extract different useful P300-related features. Experimental results on three datasets show that our EoCNN increases the P300 signal detection accuracy, the character spelling accuracy, and the ITR achieved by OCLNN (proposed and presented in Chapter 3) and other related methods for the P300 speller. In addition, our EoCNN is robust across different subjects.

Unfortunately, the complexity of our EoCNN is only lower than the complexity of CNN-R, and higher than the complexity of OCLNN, CCNN, and BN3. Thus, when compared to CNN-R, we should use our EoCNN for the P300 speller because our EoCNN has lower complexity and achieves higher P300 signal detection accuracy, character spelling accuracy, and ITR than CNN-R. When compared with OCLNN, CCNN, and BN3, if the hardware platform used in an efficient P300-based BCI system cannot support the high complexity of EoCNN, we need to choose a network among OCLNN, CCNN, and BN3 to be used for the P300 speller. In this case, we should use OCLNN because OCLNN is better than CCNN and BN3 for the P300 speller (For detailed explanation on why OCLNN is better than CCNN and BN3 for the P300 speller, please see Section 3.3.5). If the hardware platform used in an efficient P300-based BCI system can support the complexity of EoCNN, we should use EoCNN in such P300-based BCI system because EoCNN is able to achieve higher P300 signal detection accuracy, character spelling accuracy, and ITR than OCLNN, CCNN, and BN3 for the P300 speller.

Chapter 5

A Novel Sensor Selection Method based on Convolutional Neural Network for P300 Speller

Hongchang Shan, and Todor Stefanov,
"SLES: A Novel CNN-based Method for Sensor Reduction in P300 Speller,"
In *Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'19)*, Berlin, Germany, July 23-27, 2019.

Hongchang Shan, and Todor Stefanov,
"A Novel Sensor Selection Method based on Convolutional Neural Network for P300 Speller in Brain Computer Interface",
The 56th ACM/IEEE Design Automation Conference (DAC'19) WIP session, Las Vegas, NV, USA, June 2-6, 2019.

P300 spellers are still not used in human's daily life and remain in an experimental stage at research labs. Some of the reasons for this situation are : 1) Current popular EEG headsets in BCI systems, used for P300 spellers, utilize a large number of sensors to achieve high spelling accuracy. For example, the BCI systems Brain Products ActiCHamp [Act], g.HIamp [g.H], and Biosemi ActiveTwo [Bio18] utilize up to 160, 256, and 280 sensors, respectively. The price of the EEG headset is significantly high when the number of sensors is large because a lot of sensors require a complicated electrode cap and a lot of amplifier channels. For example, a 280-sensor BCI system (e.g., BioSemi ActiveTwo) costs around 87000 dollars while a 14-sensor BCI system (e.g., EMOTIV EPOC+ [EMO]) costs 799 dollars; 2) Utilizing a large number of sensors makes the P300 speller to consume a lot of power, which is unacceptable for a battery-powered mobile BCI practical system. Such system utilizes a wireless EEG headset and a resource-constrained hardware platform for data processing. A large number of sensors increases the amount of the data needed to be recorded and processed, thereby increasing the power consumption of the wireless BCI headset and the hardware platform. This does not allow a mobile practical P300 speller to work for a long time period on a single battery charge; 3) Utilizing a large number of sensors strengthens the user's discomfort and increases the installation time of the P300 speller.

To address the aforementioned problems caused by the utilization of a large number of sensors, sensor selection methods could be used to select an appropriate sensor subset from an initial large set of sensors while keeping acceptable spelling accuracy. So, a good sensor selection method should enable substantial reduction of the sensors needed to acquire brain signals. Therefore, good sensor selection methods are in urgent need for designing comfortable, cheap, and power-efficient P300 spellers and for promoting such P300 spellers into the human's daily life. Sensor selection methods for the P300 speller have been studied in recent years. For example, [RG08] [RSG$^+$09] [CRC$^+$10] [CRC$^+$11] utilize a backward elimination algorithm as a sensor selection strategy. These works propose different ranking functions to evaluate and eliminate sensors such as the P300 signal detection accuracy, the P300 spelling accuracy [CRC$^+$11], the C_{cs} score [RG08], Signal to Signal and Noise Ratio (SSNR) [RSG$^+$09] [CRC$^+$10] [CRC$^+$11], Area Under the Receiver Operating Characteristic (AUC) [CRT$^+$14]. Alternatively, [CG11] and [LWG$^+$18] directly select the important sensors for a given user by analysing the weights of a trained neural network. Unfortunately, the aforementioned sensor selection methods cannot select an appropriate sensor subset such that they can further reduce the number of sensors used to acquire brain signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used. As a consequence, the cost, power consumption, and discomfort of a P300 speller are still unacceptably high when using the aforementioned sensor selection methods to design and configure P300 spellers. In order to further reduce the cost and power consumption of a P300 speller, we propose an effective sensor selection method based on a specific novel CNN, i.e., the OSLN, we have devised and presented in Chapter 4. The novel contributions of this chapter are the following:

- We parameterize the OSLN with the number of sensors used for the acquisition of EEG signals. Our sensor selection method uses this parameterized CNN to evaluate and rank the sensors during the sensor selection process. This method features an iterative parameterized backward elimination algorithm to eliminate and select sensors. The parameter, configured in this backward elimination algorithm, controls the training frequency of the CNN and the number of sensors to eliminate in every iteration.

- We perform experiments on three benchmark datasets and compare the minimal number of sensors selected by our proposed method and other selection methods needed to acquire EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used. The results show that, compared with the minimal number of sensors selected by other methods, our method can reduce this number with up to 44 sensors.

The rest of the chapter is organized as follows. Section 5.1 describes the related work. Section 5.2 presents our proposed sensor selection method. Section 5.3 describes the experimental setup and the experimental results on the comparison of the minimal number of sensors selected by our proposed method and other sensor selection methods to acquire brain signals for the P300 speller. Section 5.4 discusses how the number of sensors eliminated in an iteration influences the performance of our proposed method as well as how the CNN network architecture influences the sensor selection process. Section 5.5 ends the chapter with conclusions.

5.1 Related Work

In this section, we describe the related works on sensor selection methods for the P300 speller in BCI.

[RG08] [CRC$^+$11] utilize a backward elimination algorithm as a sensor selection strategy. Different ranking functions are proposed to evaluate and eliminate sensors. These ranking functions include the P300 detection accuracy, the average spelling accuracy across different epochs [CRC$^+$11], the C_{cs} score [RG08], Signal to Signal and Noise Ratio (SSNR) [CRC$^+$11], and Area Under the Receiver Operating Characteristic (AUC) [CRT$^+$14]. In order to select a sensor subset, the backward elimination algorithm either eliminates one sensor [CRC$^+$11] or a group of sensors [RG08] in each iteration of the algorithm. Starting with a set of n sensors in an iteration, the backward elimination algorithm removes each sensor in the current sensor set and evaluates the resulting subsets with $(n-1)$ sensors using the aforementioned ranking functions. The sensor or the group of sensors which removal maximizes the ranking score is eliminated. In contrast to these methods, we proposes a novel ranking function (see Section 5.2.3) based on the OSLN we have devised and presented in Chapter 4. Experimental results (see Section 5.3.2) show that our sensor selection method is able to select a sensor subset with smaller number of sensors needed to acquire the EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used, compared with the sensor subset selected by the aforementioned sensor selection methods. Therefore, our sensor selection method can further reduce the cost and power consumption of the P300 speller.

[CG11] and [LWG$^+$18] propose CNN-based classifiers for character spelling in the P300 speller. By analysing the weights of the spatial convolution layer of their trained CNNs, they determine which sensors are more important in the sensor set. This can be a potential sensor selection method for the P300 speller. However, the problem of such potential method is that it loses important information needed for proper sensor selection. The aforementioned CNNs have multiple convolution layers. The information needed for proper sensor selection is distributed over the weights of all convolution layers. In [CG11] and [LWG$^+$18], only the weights of the first layer are used for analysis and sensor selection because the weights of the other convolution layers can hardly be used for sensor selection (for the detailed explanation of the reason for this please refer to Section 5.4.2). Thus, the aforementioned methods cannot use all the information available for proper sensor selection. In contrast to the aforementioned CNNs, our proposed OSLN has only one convolution layer and this layer performs the spatial convolution operation. All the information needed for sensor selection is captured by the weights of this single spatial convolution layer. Moreover, our CNN has similar ability to extract very useful P300-related features compared to the aforementioned CNNs (see Table 4.11 and Table 3.8). We analyse the weights of the single spatial convolution layer in our CNN to select sensors. Thus, our method uses all the information available for proper sensor selection compared to the aforementioned methods. As a result, our method can select more appropriate sensor subsets and further reduce the minimal number of sensors needed to acquire brain signals without losing spelling accuracy. For more detailed discussion see Section 5.4.2.

5.2 Our Sensor Selection Method

In this section, we present our novel iterative sensor selection method for the P300 speller. We call it Spatial Learning based Elimination Selection (SLES).

5.2.1 Spatial Learning based Elimination Selection

Our SLES method is described in Algorithm 1. The symbols used in Algorithm 1 and their corresponding descriptions are listed in Table 5.1. The input of SLES is the initial sensor set S and the parameter E_s. The output of SLES is a set of selected sensor subsets SUB. For each iteration in Algorithm 1, SLES trains $OSLN_{(S)}$ with the input signals recorded with the sensors in sensor set S (see Line 2 in Algorithm 1). $OSLN_{(S)}$ (described in Section 5.2.2) is the parameterized version of the OSLN (proposed in Chapter 4) with S as a parameter. After training $OSLN_{(S)}$, the ranking scores $score_j$ for all sensors s_j in sensor set S are calculated (Line 3-4) using

$OSLN_{(S)}$ and Equation (5.2) explained in Section 5.2.3. The sensor with the minimal score is found and removed from sensor set S (Lines 6-7). This reduced sensor set S is the selected sensor subset in this iteration (Line 8). The input parameter E_s controls the training frequency of $OSLN_{(S)}$ (Line 1) and the number of sensors to eliminate after training $OSLN_{(S)}$ (Line 5).

Table 5.1: The symbols used in Algorithm 1.

Symbol	Description
S	Sensor set.
s_j	The jth sensor in S.
C	Number of sensors in the initial sensor set.
SUB	A set of selected sensor subsets.
sub_m	A selected sensor subset with m sensors.
$OSLN_{(S)}$	The novel parameterized CNN given in Section 5.2.2
E_s	Number of sensors to eliminate in an iteration.
$score_j$	The ranking score for s_j.
s_{remove}	The sensor to remove.

5.2.2 Parameterized OSLN

In this section, we describe in details the $OSLN_{(S)}$ (used in Algorithm 1), which is the parameterized version of the OSLN, proposed and presented in Chapter 4.

5.2.2.1 Input Tensor

The input to $OSLN_{(S)}$ is the tensor ($N \times |S|$) shown in Figure 5.1. S is the sensor set used in Algorithm 1. x_{ji} denotes the ith temporal signal sample in the time domain and this signal sample is recorded with sensor s_j in sensor set S in the space domain. $OSLN_{(S)}$ is parameterized by S because the input tensor to $OSLN_{(S)}$ is constructed by the EEG signal samples acquired using the sensors in sensor set S and S is changed in each main iteration of Algorithm 1 (see Line 7). N denotes the number of temporal

Algorithm 1: Proposed SLES algorithm.

```
Input: Set $S$ = {$s_1$, $s_2$, ..., $s_j$, ... $s_C$}, $E_s$;
Output: Set $SUB$ = {$sub_1$, $sub_2$, ... , $sub_m$, ... $sub_{C-1}$};
for $1 \leq k \leq C/E_s$ do
    Train a $OSLN_{(S)}$ with the input signals recorded using $S$;
    for $s_j \in S$ do
        Calculate $score_j$ using $OSLN_{(S)}$ and Equation (5.2);
    for $1 \leq m \leq E_s$ do
        $s_{remove}$ = $\underset{s_j \in S}{argmin}$ {$score_j$};
        $S \leftarrow S - s_{remove}$;
        $sub_{(C-E_s*(k-1)-m)} \leftarrow S$;
```

signal samples. These temporal signal samples are preprocessed in the same way as explained in Section 3.2.1 of Chapter 3.

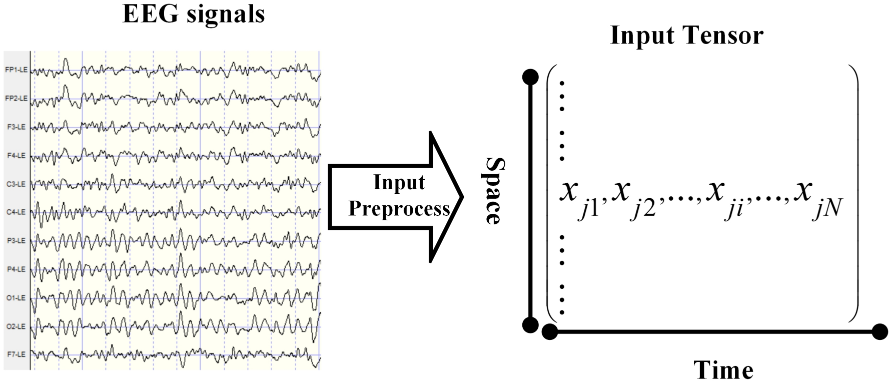


Figure 5.1: Input tensor to $OSLN_{(S)}$, where $s_j \in S$.

5.2.2.2 Network Architecture

Table 5.2 shows the details of the $OSLN_{(S)}$ architecture. The first column shows the name of the layers. The second column shows the operation performed in the corresponding layer. The third column shows the kernel size in the convolution layer. The fourth column shows how many feature maps or neurons are utilized in the convolution or fully-connected layer. The difference between $OSLN_{(S)}$ and OSLN (see Table 4.1 in Section 4.1.2) is Layer 1, i.e., the convolution layer. Thus, we describe only Layer 1 of $OSLN_{(S)}$ in this section.

Table 5.2: $OSLN_{(S)}$ architecture.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(1,$\|S\|$)	16
	Dropout	—	—
2	Fully-Connected	—	2

In Layer 1, $OSLN_{(S)}$ performs a spatial convolution operation to extract the spatial features related to P300 signals from the input tensor. The detailed calculation in this convolution operation is shown in Equation (5.1), where f_{ki} denotes the ith datum in the kth feature map. w_{kj} denotes the jth weight of the filter and this filter outputs abstract data for the kth feature map. The activation function we utilize in this layer is the Rectified Linear Unit (ReLU). In this layer, we utilize Dropout in order to prevent the network from overfitting. In this layer, we do not use bias in the convolution operation, thus all the learned features are captured by the weights w_{kj}. This layer outputs 16 feature maps in total.

$$f_{ki} = \sum_{s_j \in S} x_{ji} w_{kj} \tag{5.1}$$

5.2.3 Ranking Function

Our proposed novel ranking function used in SLES is given in Equation (5.2), where $score_j$ is the ranking score for sensor s_j used in Algorithm 1. w_{kj} are the weights described in Equation (5.1). These weights are obtained from the trained $OSLN_{(S)}$, described in Section 5.2.2.2 and used in Algorithm 1. Note that we take the absolute value of the weights in Equation (5.2) because weights with a large negative value also indicate that the corresponding sensors are important in sensor set S. We use the absolute values of the weights from the spatial convolution layer (i.e., Layer 1 in Table 5.2) of the trained $OSLN_{(S)}$ in the ranking function to rank the sensors in the sensor set because [CG11] and [LWG^{+}18] have shown that analyzing the weights of the spatial convolution layer from trained CNNs for the P300 speller is a potential method to determine which sensors are more important in the sensor set. For details, please refer to [CG11] and [LWG^{+}18].

$$score_j = \sum_{k=1}^{16} |w_{kj}| \tag{5.2}$$

We have proposed three CNNs with one convolution layer, i.e., OSLN, OCLNN (proposed and presented in Chapter 3), and OTLN (proposed and presented in Section 4.1.2 of Chapter 4). Our ranking function to rank the sensors in the sensor set is based on the parameterized OSLN instead of a parameterized version of OCLNN as well as instead of a parameterized version of OTLN. The reason for this is explained in Section 5.4.2.

5.3 Experimental Evaluation

In this section, we present the experiments, we have performed, in order to compare the minimal number of sensors selected by our method and other methods to acquire EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used. We first introduce our experimental setup and then we present and analyse the obtained experimental results.

5.3.1 Experimental Setup

To perform the experiments, we use 3 different implementations of the P300 speller: the OCLNN-based P300 speller (proposed and presented in Chapter 3), the EoCNN-based P300 speller (proposed and presented in Chapter 4), and the SVM-based P300 speller [RG08]. We want to confirm the robustness of our SLES method by showing that our method is effective for different P300 speller implementations.

We compare our SLES method with 12 other sensor selection methods. These methods are summarized in Table 5.3. In this table, the first row gives the names of the different methods. The second row describes the sensor elimination algorithms used in the methods, where BE-1 denotes a backward elimination algorithm which eliminates one sensor at a time; BE-4 denotes a backward elimination algorithm which eliminates 4 sensors at a time; "–" denotes that the corresponding method does not use a backward elimination algorithm. The last row indicates the ranking functions used in the methods, where P300 denotes the P300 detection accuracy; Char denotes the average character spelling accuracy across all epochs; AUC denotes Area Under the Receiver Operating Characteristic [CRT$^+$14]; C_{cs} denotes the ranking score proposed in [RG08]; SSNR denotes Signal to Signal and Noise Ratio [CRC$^+$11]; CCNN and BN3 denote that the corresponding method selects sensors by analysing the weights obtained from the trained networks CCNN [CG11] and BN3 [LWG$^+$18], respectively.

Table 5.3: Methods compared with SLES.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Algo.	BE-1	BE-1	BE-1	BE-1	BE-1	BE-4	BE-4	BE-4	BE-4	BE-4	–	–
Function	P300	Char	AUC	C_{cs}	SSNR	P300	Char	AUC	C_{cs}	SSNR	CCNN	BN3

We compare the minimal number of sensors selected by the different methods to acquire EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set is used. We use the training dataset of Dataset II, III-A and III-B (described in Section 2.5) as the preliminary dataset to perform sensor selection using the different sensor selection methods to select sensor subsets for the corresponding subject. More specifically, for our SLES method, this preliminary dataset is used to train a $OSLN_{(S)}$ and calculate $score_j$ in each iteration of our SLES method (see Algorithm 1). For the sensor selection methods C_1 , C_2 , C_3 , C_4 , C_5 , C_6 , C_7 , C_8 , C_9 , and C_{10} (see Table 5.3), this preliminary dataset is used to calculate the P300 detection accuracy (for C_1 and C_6), the average character spelling accuracy across all epochs (for C_2 and C_7), AUC (for C_3 and C_8), the C_{cs} score (for C_4 and C_9), and the SSNR score (for C_5 and C_{10}). For the sensor selection methods C_{11} and C_{12} (see Table 5.3), this preliminary dataset is used to train CCNN (for C_{11}) and BN3 (for C_{12}) in order to analyze the weights of the trained CCNN and BN3 to select appropriate sensor subsets from the initial sensor set. After using different sensor selection methods to select sensor subsets, we calculate the spelling accuracy of the aforementioned OCLNN-based P300 speller, EoCNN-based P300 speller, and SVM-based P300 speller with the selected sensor subsets. The training datasets of Dataset II, III-A and III-B are used to train the classifier used in the aforementioned P300 speller implementations with the selected sensor subsets. Then, the test dataset of Dataset II, III-A and III-B are used to calculate the spelling accuracy of the aforementioned P300 speller implementations with the selected sensor subsets. The spelling accuracy is calculated using Equation (5.3). In this equation, $acc^{m}_{char(k)}$ denotes the spelling accuracy when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m number of sensors. $N^{m}_{tc(k)}$ denotes the number of truly predicted characters when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m number of sensors, and S_c denotes the number of all characters in the evaluation dataset. After the evaluation of the spelling accuracy, the minimal number of sensors needed to acquire EEG signals for epoch k is calculated as m_{min}, where m_{min} is the minimal $m \in [1, 63]$ which makes $acc^{m}_{char(k)} >= acc^{64}_{char(k)}$.

$$acc^{m}_{char(k)} = \frac{N^{m}_{tc(k)}}{S_c} \tag{5.3}$$

The setup for our SLES algorithm (Algorithm 1) is the following. The input to SLES is S = $\{s_1, s_2, ..., s_j, ...\ s_C\}$ and E_s. We set C=64 because the datasets used in the experiments are recorded with 64 sensors. We set E_s=4. For detailed discussion why E_s=4 see Section 5.4.1. SLES uses $OSLN_{(S)}$ as the ranking function. $OSLN_{(S)}$ uses the input tensor ($N \times |S|$). N = 240 because the signal sampling frequency is 240 Hz and we take each individual pattern to be the signal samples between 0 and 1000 ms posterior to the beginning of each intensification.

5.3.2 Experimental Results

Table 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12 show the minimal number of sensors selected by the different sensor selection methods to acquire EEG signals while keeping the spelling accuracy the same as the accuracy achieved when the initial large sensor set of all 64 sensors is used. The first column in the tables lists the different selection methods we compare. Each row provides the minimal number of sensors selected by a method to acquire EEG signals for different epoch numbers $k \in [1, 15]$. A number in bold indicates that the minimal number of sensors selected by the corresponding method is the lowest among all methods. Overall, the minimal number of sensors selected by our SLES method is lower than the minimal number of sensors selected by all other methods in most cases. SLES is able to reduce the minimal number of sensors selected by other methods with up to 44 sensors.

For the P300 speller with our CNN-based classifiers, i.e., OCLNN and EoCNN, (see Table 5.4, 5.5, 5.6, 5.7, 5.8, and 5.9), in 83 out of 90 cases, the minimal number of sensors selected by our SLES is lower than the minimal number of sensors selected by all other methods. Our SLES is able to reduce the minimal number of sensors selected by other methods with up to 44 sensors. The largest reduction occurs when comparing the minimal number of sensors selected by SLES with the minimal number of sensors selected by C_8 on epoch number $k = 7$ for Dataset III-A using the OCLNN-based P300 speller.

For the P300 speller with the SVM-based classifier (see Table 5.10, 5.11 and 5.12), in 41 out of 45 cases, the minimal number of sensors selected by our SLES is lower than the minimal number of sensors selected by all other methods. Our SLES is able to reduce the minimal number of sensors selected by other methods with up to 40 sensors. The largest reduction occurs when comparing the minimal number of sensors selected by SLES with the minimal number of sensors selected by C_{12} on epoch number $k = 2$ for Dataset III-B.

Finally, our SLES method is robust because: 1) SLES is effective in reducing the number of sensors when the P300 speller is implemented with different classifiers. From Table 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12, we can see that no matter the P300 speller is implemented with CNN-based classifier or SVM-based classifier, the minimal number of sensors selected by SLES is lower than the minimal number of sensors selected by all other methods in most cases; 2) SLES is effective when it is used for different subjects, i.e., no matter that SLES is used with Dataset III-A, Dataset III-B or Dataset II, the minimal number of sensors selected by SLES is lower than the minimal number of sensors selected by all other methods in most cases.

Table 5.4: Minimal number of sensors selected by different methods for Dataset II. The P300 speller is implemented using the CNN-based classifier OCLNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**11**	**11**	**46**	**15**	**22**	11	**9**	**3**	**3**	**3**	**3**	4	**3**	**3**	**3**
C_1	32	30	47	39	38	22	16	6	6	6	6	6	6	7	6
C_2	17	20	55	19	34	15	12	6	5	3	4	**3**	6	7	7
C_3	18	18	47	18	30	10	12	6	5	**3**	5	6	6	6	6
C_4	24	27	49	17	28	**8**	10	7	6	7	4	**3**	**3**	5	6
C_5	40	33	48	41	38	35	20	9	8	9	9	8	9	9	8
C_6	48	33	49	47	32	30	28	28	20	15	10	20	10	15	15
C_7	44	32	49	48	31	27	27	27	22	18	10	10	8	10	10
C_8	44	36	50	38	33	32	25	25	10	10	10	10	10	17	12
C_9	45	35	44	34	34	34	25	25	10	17	18	17	15	15	15
C_{10}	48	35	49	44	40	30	33	29	21	19	20	20	20	18	17
C_{11}	25	25	54	24	24	14	15	15	10	10	12	17	15	15	15
C_{12}	29	27	59	25	31	22	15	18	13	13	15	18	19	21	18

5.4 Discussions

In this section, we discuss the configuration of input parameter E_s in SLES (see Algorithm 1). Also, we discuss the impact of different CNN architectures on selecting sensors.

Table 5.5: Minimal number of sensors selected by different methods for Dataset III-A. The P300 speller is implemented using the CNN-based classifier OCLNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**17**	**50**	**50**	**25**	**40**	**35**	**20**	**50**	**60**	**20**	**50**	**35**	**21**	**16**	35
C_1	27	64	58	59	55	64	43	58	64	29	57	49	29	29	53
C_2	29	64	64	60	59	59	36	60	**60**	31	56	56	31	28	60
C_3	28	64	64	60	64	60	55	64	64	36	64	55	39	39	46
C_4	53	64	63	64	64	64	60	64	61	61	64	56	55	37	61
C_5	30	64	56	64	55	64	43	63	64	31	64	57	36	33	55
C_6	30	63	64	64	64	64	57	64	64	61	64	53	53	53	59
C_7	44	57	64	64	64	64	62	64	64	55	64	53	59	52	51
C_8	49	59	60	63	64	64	64	63	64	60	64	56	56	54	56
C_9	22	64	56	55	58	64	56	56	64	34	52	48	27	36	**34**
C_{10}	34	63	64	64	64	64	59	64	64	61	64	58	61	53	59
C_{11}	25	64	56	55	58	64	56	56	64	34	54	48	27	36	39
C_{12}	28	64	64	56	64	64	58	56	64	41	64	53	39	36	41

5.4.1 Configuration of E_s in SLES

In this section, we show how we configure the input parameter E_s in SLES. We use only the preliminary dataset of Dataset III-A and use the OCLNN-based P300 speller implementation to show the experiments on how to tune E_s because we obtain the same E_s value when we perform experiments using all datasets and using all the aforementioned P300 speller implementations to tune E_s for SLES. We divide the preliminary dataset of Dataset III-A into two parts. The first part contains 60% of the preliminary dataset of Dataset III-A. The second part contains the left 40% of the preliminary dataset of Dataset III-A. The first part, i.e., the 60% of the preliminary dataset of Dataset III-A, is used to train $OSLN_{(S)}$ (see Section 5.2.2) while running SLES with different E_s configurations, i.e., E_s=1, 2, 4, 8, 16, 32 and 64. With each E_s configuration, SLES selects a set of sensor subsets for Dataset III-A. The second part, i.e., the left 40 % of the preliminary dataset of Dataset III-A, is used to evaluate the spelling accuracy of the aforementioned P300 speller implementation with the selected sensor subsets. The P300 spelling accuracy is calculated using Equation (5.3). Then, we

Table 5.6: Minimal number of sensors selected by different methods for Dataset III-B. The P300 speller is implemented using the CNN-based classifier OCLNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**25**	**17**	**31**	**27**	**26**	**21**	**27**	**32**	**37**	**32**	**31**	**29**	**34**	**29**	**23**
C_1	32	23	**31**	28	31	24	32	40	43	44	38	32	42	35	26
C_2	28	24	52	38	36	25	35	45	41	41	**31**	30	42	38	31
C_3	26	29	42	50	35	23	32	54	49	48	38	**29**	39	38	33
C_4	31	25	61	40	45	24	38	57	60	57	59	55	49	47	43
C_5	44	33	36	34	31	24	34	41	43	48	38	42	42	37	26
C_6	50	56	55	49	48	58	48	50	50	50	48	44	51	49	45
C_7	52	48	49	49	50	49	49	59	48	48	48	46	45	47	46
C_8	49	49	54	52	49	54	48	59	49	49	49	44	46	44	44
C_9	44	56	49	49	40	25	35	39	39	49	51	37	**34**	35	42
C_{10}	49	52	55	44	50	51	46	50	52	51	49	50	51	49	50
C_{11}	44	56	49	49	40	25	35	49	39	49	51	37	38	34	42
C_{12}	54	61	59	55	52	27	30	42	45	63	54	48	47	34	42

calculate the minimal number of sensors m_{min} for the different E_s configurations as described in Section 5.3.1. In this experiment, we divide the preliminary dataset of Dataset III-A into two parts, i.e., one part containing 60% of the preliminary dataset of Dataset III-A and one part containing the left 40% of this dataset because the majority of the researchers use this ratio to split a dataset [XMYR16, VELB18, LLJ$^+$18].

The experimental results are shown in Table 5.13. The first column in the table lists the different configurations of E_s in SLES. Each row provides the minimal number of sensors selected by SLES for different epoch numbers $k \in [1, 15]$. A number in bold indicates that the minimal number of sensors selected by SLES with the corresponding E_s configuration is the lowest compared with the minimal number of sensors selected by SLES with other E_s configurations. From Table 5.13, we can see that, in most cases, the minimal number of sensors selected by SLES with E_s=4 is the lowest compared to the minimal number of sensors selected by SLES with other E_s configurations. Therefore, we set E_s=4 when using SLES.

Table 5.7: Minimal number of sensors selected by different methods for Dataset II. The P300 speller is implemented using the CNN-based classifier EoCNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**10**	**16**	**28**	**19**	**22**	9	**8**	**3**	**3**	**3**	**3**	4	**3**	**3**	**3**
C_1	28	33	40	41	38	19	15	6	6	6	6	6	6	7	6
C_2	15	24	48	**19**	34	14	12	6	5	**3**	4	**3**	6	7	7
C_3	14	20	41	20	30	9	10	6	5	**3**	5	6	6	6	6
C_4	19	29	43	21	28	**7**	9	7	6	8	4	**3**	**3**	5	6
C_5	31	34	42	43	38	31	17	9	8	9	9	8	9	9	8
C_6	43	35	44	49	33	29	25	27	20	15	10	20	10	15	15
C_7	39	32	45	52	31	27	26	27	22	18	10	10	8	11	10
C_8	37	36	47	40	33	30	25	25	10	10	10	10	10	17	12
C_9	42	37	43	36	34	33	24	25	10	18	18	17	15	15	15
C_{10}	43	38	45	47	40	28	32	29	21	19	20	20	20	18	15
C_{11}	19	29	48	25	23	13	12	15	10	10	11	17	15	15	15
C_{12}	26	31	53	28	31	18	13	18	13	13	15	18	19	19	18

5.4.2 Exploring the Impact of the CNN Architecture on Sensor Selection

We perform experiments to explore the impact of different CNN architectures on the sensor selection process in order to address the issue mentioned in the third paragraph of Section 5.1 and the issue mentioned at the end of Section 5.2.3. The P300 speller implemention used for this experiment is the CNN-based classifier OCLNN. We use the preliminary dataset of Dataset III-A to train our OSLN, OTLN (proposed and presented in Section 4.1.2 of Chapter 4), OCLNN (proposed and presented in Chapter 3), CCNN [CG11], and BN3 [LWG$^+$18]. We select sensor subsets by directly analyzing the weights of the convolution layer of OSLN, OCLNN, and OTLN, as well as select sensor subsets by directly analyzing the weights of the first convolution layer of CCNN and BN3 (as done in [CG11] and [LWG$^+$18]). We use the evaluation dataset of Dataset III-A to evaluate the P300 spelling accuracy of the aforementioned P300 speller implementation with the selected sensor subsets. Then, we calculate the minimal number of sensors m_{min} selected by analysing the weights of OSLN, OCLNN, OTLN, CCNN, and BN3. For the detailed calculation of m_{min} see Section 5.3.1.

Table 5.8: Minimal number of sensors selected by different methods for Dataset III-A. The P300 speller is implemented using the CNN-based classifier EoCNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**18**	**39**	**37**	**21**	**41**	**38**	**20**	**39**	**44**	**23**	**46**	**33**	**21**	**16**	**31**
C_1	27	51	44	55	55	60	43	46	50	30	52	46	29	29	50
C_2	30	53	50	56	59	55	36	49	52	35	51	54	31	27	56
C_3	30	53	50	56	64	56	55	52	53	39	59	53	39	39	41
C_4	55	52	48	59	63	60	60	52	49	62	60	52	55	37	56
C_5	31	54	51	59	55	60	43	50	50	36	59	55	36	33	50
C_6	31	52	50	58	64	59	57	53	52	62	61	50	54	53	56
C_7	44	56	52	55	62	60	61	51	53	58	60	50	59	52	48
C_8	50	57	47	52	64	60	64	50	52	61	60	51	56	54	53
C_9	23	55	43	51	55	58	56	45	51	40	48	42	27	36	**31**
C_{10}	34	56	49	53	64	60	59	64	52	62	60	53	61	53	52
C_{11}	26	53	42	50	58	61	56	46	52	39	51	44	26	36	36
C_{12}	30	60	48	55	64	63	58	52	53	44	59	48	39	36	40

The experimental results are shown in Table 5.14. The first column in the table lists the different CNNs. Each row provides the minimal number of sensors selected by analysing the weights of the different CNNs for different epoch numbers $k \in [1, 15]$. A number in bold indicates that the minimal number of sensors selected by analysing the weights of the corresponding CNN is the lowest, compared to the minimal number of sensors selected by analysing the weights of other CNNs.

Table 5.14 shows that the minimal number of sensors selected by analysing the weights of our proposed one-convolution-layer CNNs, i.e., OSLN, OCLNN, as well as OTLN, is lower than the minimal number of sensors selected by analysing the weights of CCNN and BN3. The reason is the following. CCNN and BN3 have multiple convolution layers. The information needed for proper sensor selection is distributed over the weights of all convolution layers. In CCNN and BN3, only the weights of the first convolution layer are used for analysis and proper sensor selection because the weights of the other convolution layers can hardly be used for proper sensor selection. This is because the information for the importance of each sensor in the sensor set is stored in the input neurons of the input tensor to a CNN. This

Table 5.9: Minimal number of sensors selected by different methods for Dataset III-B. The P300 speller is implemented using the CNN-based classifier EoCNN.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**25**	**26**	**28**	**27**	**23**	**14**	**16**	**25**	**30**	**29**	33	**22**	30	**29**	**23**
C_1	31	30	**28**	28	27	18	23	34	37	40	37	29	39	35	26
C_2	28	32	48	38	32	20	24	38	35	39	**31**	27	39	38	31
C_3	26	40	38	50	33	21	22	47	42	45	36	26	37	38	33
C_4	31	31	57	40	42	20	25	50	53	52	56	40	40	47	43
C_5	44	39	33	34	29	22	26	34	36	46	38	36	38	37	26
C_6	50	60	50	49	45	42	37	45	45	49	46	38	47	49	45
C_7	52	55	48	49	46	39	38	52	40	47	47	41	41	45	46
C_8	49	55	51	52	48	45	37	53	42	46	45	37	41	44	44
C_9	44	62	46	49	38	23	24	35	33	45	50	32	**29**	36	42
C_{10}	50	60	52	44	49	41	35	44	46	48	49	41	47	49	50
C_{11}	44	60	48	49	37	23	25	36	35	42	49	30	35	34	41
C_{12}	54	64	56	55	45	26	28	42	41	51	52	38	40	34	42

input tensor is directly related to the first convolution layer of CCNN and BN3 by directly connecting each receptive field of the input neurons in the input tensor with each neuron in the first layer of CCNN and BN3. These connections are expressed by their corresponding weight in the first convolution layer. Thus, the weights of the first convolution layer of CCNN and BN3 can be used for analysis and sensor selection. However, the weights of the other convolution layers of CCNN and BN3 only express the connections of the neurons of the first convolution layer with the neurons of the other convolution layers. We can hardly build any direct relation between the input tensor (that stores the information for the importance of each sensor in the sensor set) with the neurons in the other convolution layers of CCNN and BN3 by using the weights of these layers. As a result, the weights of the other layers of CCNN and BN3 can hardly be used for analysis and proper sensor selection. Therefore, CCNN and BN3 cannot use all the information available for proper sensor selection. In contrast, our OSLN, OCLNN, OTLN have only one convolution layer. All the information needed for sensor selection is captured by the weights of the single convolution layer of OSLN, OCLNN, and OTLN. We analyse the weights of the single convolution layer

Table 5.10: Minimal number of sensors selected by different methods for Dataset II, The P300 speller is implemented using the SVM-based classifier ESVM [RG08].

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**13**	**10**	**36**	**20**	**31**	**16**	**8**	**4**	**3**	**5**	**3**	**5**	**3**	**6**	**3**
C_1	30	32	42	38	43	28	15	8	6	9	7	10	6	8	6
C_2	25	28	40	29	40	19	16	9	6	11	**3**	11	7	**6**	8
C_3	21	23	38	30	34	25	13	8	7	10	6	14	7	7	7
C_4	26	24	**36**	33	38	18	14	10	8	12	5	13	4	9	9
C_5	43	29	37	40	41	36	24	11	9	13	8	12	10	16	11
C_6	50	38	52	50	50	33	26	30	18	19	11	24	12	14	16
C_7	47	40	55	48	49	29	29	29	16	21	13	23	11	17	13
C_8	50	37	54	42	46	31	30	29	16	23	13	20	16	16	19
C_9	51	35	49	39	54	39	29	31	19	25	14	19	13	19	17
C_{10}	49	44	50	40	50	30	31	26	20	21	19	19	19	20	16
C_{11}	28	30	56	30	36	19	17	16	9	20	16	19	16	17	12
C_{12}	31	33	61	36	38	23	18	20	11	24	20	21	21	26	14

in OSLN, OCLNN, and OTLN to select sensors. Thus, OSLN, OCLNN, and OTLN use all the information available for proper sensor selection compared to CCNN and BN3. As a result, OSLN, OCLNN, and OTLN can select more appropriate sensor subsets and further reduce the minimal number of sensors needed to acquire EEG signals without losing spelling accuracy. The aforementioned discussion explains the issue mentioned in the third paragraph of Section 5.1.

The experimental results in Table 5.14 also explain why in Section 5.2.3, our ranking function to rank the sensors in the sensor set is based on OSLN instead of OCLNN and OTLN. When compared with OTLN, the minimal number of sensors selected by analyzing the weights of OSLN is lower than the minimal number of sensors selected by analyzing the weights of OTLN. Therefore, our ranking function in SLES is based on OSLN instead of OTLN. When compared with OCLNN, the minimal number of sensors selected by analyzing the weights of OSLN is comparable with the minimal number of sensors selected by analyzing the weights of OCLNN. However, the network complexity of OSLN (8,722 parameters) is only 51.67% of the complexity of OCLNN (16882 parameters). This means that the time of training the OSLN is much

Table 5.11: Minimal number of sensors selected by different methods for Dataset III-A, The P300 speller is implemented using the SVM-based classifier ESVM [RG08].

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**19**	**44**	**54**	**38**	**40**	**42**	33	56	**58**	**39**	**44**	**27**	**35**	**21**	**36**
C_1	31	56	58	49	45	57	44	**54**	64	43	55	51	46	32	56
C_2	32	53	57	48	44	59	47	55	64	43	50	53	47	32	55
C_3	29	53	**54**	48	47	56	35	55	63	45	52	57	42	36	49
C_4	56	50	53	44	42	55	**31**	**54**	60	56	49	51	43	37	57
C_5	31	55	58	50	45	49	43	60	64	46	58	48	47	33	57
C_6	35	60	61	57	54	56	52	64	64	53	62	59	57	50	62
C_7	43	59	61	55	49	57	52	64	64	50	62	58	55	50	62
C_8	47	54	64	57	48	54	55	62	64	59	57	52	56	49	63
C_9	33	55	60	51	48	62	59	63	64	56	53	54	39	46	49
C_{10}	33	62	62	59	53	57	57	64	64	54	61	60	59	52	55
C_{11}	29	61	59	54	55	59	56	58	64	46	53	45	41	40	37
C_{12}	35	64	63	56	59	61	60	59	64	51	58	47	44	46	43

smaller than the time of training the OCLNN. Thus, the speed of SLES with the ranking function based on OSLN is much higher than the speed of SLES with the ranking function based on OCLNN (see Line 2 to Line 4 in Algorithm 1). Therefore, our ranking function in SLES is based on OSLN instead of OCLNN.

5.5 Conclusions

In this chapter, we propose a novel sensor selection method, called SLES, for reducing the number of sensors needed to acquire EEG signals for a P300 speller without losing spelling accuracy. SLES uses an iterative parameterized backward elimination algorithm to eliminate and select sensors and it uses our novel $OSLN_{(S)}$ as a ranking function to evaluate the importance of a sensor. Our SLES is also robust across different P300 speller implementations and different subjects. Experimental results show that the minimal number of sensors selected by our SLES method is lower than the minimal number of sensors selected by other methods in most cases. Therefore, our SLES can further reduce the cost and power consumption of the P300 speller, thereby

Table 5.12: Minimal number of sensors selected by different methods for Dataset III-B, The P300 speller is implemented using the SVM-based classifier ESVM [RG08].

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLES	**18**	**19**	**34**	**29**	**17**	31	**39**	**41**	**22**	**36**	**24**	33	**33**	**26**	**27**
C_1	28	27	43	39	32	35	45	50	36	51	36	**30**	45	37	29
C_2	27	25	42	38	29	31	**39**	54	36	49	40	**30**	44	30	**27**
C_3	24	31	45	35	26	30	42	49	34	44	38	32	47	**26**	**27**
C_4	23	26	59	36	27	**29**	44	52	35	42	37	36	39	37	30
C_5	35	29	44	40	29	33	48	50	39	49	38	33	43	40	31
C_6	49	58	56	51	44	40	57	61	48	59	46	42	57	51	47
C_7	41	52	53	48	46	48	53	63	50	54	49	44	52	47	44
C_8	40	50	53	49	47	50	49	60	47	50	41	44	51	50	46
C_9	42	51	48	47	41	37	52	61	43	49	45	39	46	44	45
C_{10}	42	54	51	52	38	43	57	61	44	55	50	46	53	50	49
C_{11}	43	55	49	47	34	33	45	53	40	52	51	39	49	29	46
C_{12}	52	59	56	61	46	41	51	64	49	64	56	41	60	30	46

facilitating the utilization of P300 spellers into people's daily life.

Table 5.13: Minimal number of sensors selected by SLES with different E_s configurations.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E_s=1	27	56	58	57	50	60	44	53	63	30	55	33	24	19	46
E_s=2	18	**45**	52	34	43	46	28	**48**	58	26	47	30	20	16	**36**
E_s=4	**15**	46	**49**	**28**	**40**	**36**	**18**	**48**	59	**21**	**46**	**27**	**19**	**13**	38
E_s=8	19	54	**49**	31	41	48	26	**48**	**57**	25	**46**	34	20	**13**	37
E_s=16	22	54	55	42	45	56	39	53	63	29	53	35	25	20	37
E_s=32	25	59	57	47	49	58	48	54	63	32	54	37	28	23	41
E_s=64	27	60	60	48	50	60	55	55	64	36	59	42	31	24	44

Table 5.14: Minimal number of sensors selected by analysing different CNNs.

	Epochs														
CNN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OSLN	23	**60**	**55**	**49**	51	**60**	**53**	55	64	**31**	53	**46**	**27**	**32**	**39**
OCLNN	**22**	**60**	**55**	**49**	**49**	**60**	**53**	**53**	**63**	32	**52**	**46**	**27**	33	**39**
OTLN	23	62	56	51	53	62	**53**	56	64	34	53	**46**	**27**	33	**39**
CCNN	25	64	56	55	58	64	56	56	64	34	54	48	**27**	36	**39**
BN3	28	64	64	56	64	64	58	56	64	41	64	53	39	36	41

Chapter 6

An Improved Ensemble of Convolutional Neural Networks for P300 Speller with a Small Number of Sensors

Hongchang Shan, Yu Liu, and Todor Stefanov,
"An Empirical Study on Sensor-aware Design of Convolutional Neural Networks for P300 Speller in Brain Computer Interface,"
In *Proceedings of "12th IEEE International Conference on Human System Interaction (IEEE HSI'19)"*, pp. 5-11, Richmond, Virginia, USA, June 25-27, 2019.

In Chapter 4, we have presented our EoCNN which achieves higher spelling accuracy and ITR compared to other state-of-the-art methods for the P300 speller. In Chapter 5, we have presented our SLES method that can reduce the number of sensors needed to acquire EEG signals in our EoCNN-based P300 speller while keeping the character spelling accuracy and the ITR the same as the character spelling accuracy and the ITR achieved by EoCNN when an initial large set of 64 sensors is used in the P300 speller. We call the character spelling accuracy and the ITR, achieved by EoCNN for the P300 speller with a large number of sensors (i.e, 64 sensors), the state-of-the-art character spelling accuracy and ITR of the P300 speller. Table 5.7, 5.8, and 5.9 in Chapter 5 show that in most cases, in order to not lose the state-of-the-art character spelling accuracy and ITR of the P300 speller, we need to use more than 16 sensors to acquire EEG signals in the EoCNN-based P300 speller. Unfortunately, popular low-complexity and relatively cheap (affordable) BCI systems utilize a small number of sensors for the acquisition of EEG signals. Typically, such small number of sensors is less than or equal to 16 sensors. For example, BCI systems such as MUSE [MUS], EMOTIV Insight [Ins], Quick-8 [Qui], B-Alert X10 [B-A], EMOTIV EPOC+ [EMO], and OPEN BCI Mark IV [Mar] utilize only 4, 5, 8, 10, 14, and 16 sensors, respectively. Therefore, in this chapter, we present our research on how to achieve the state-of-the-art character spelling accuracy and ITR of the P300 speller with popular low-complexity and relatively cheap BCI systems that use a small number of sensors (i.e., less than or equal to 16 sensors) to acquire EEG signals. The novel contributions of this chapter are the following.

- We perform a study on EoCNN as well as the three CNNs used in EoCNN, i.e., OTLN, OSLN, and OCLNN, for the P300 speller with different number of sensors in order to find the reason why EoCNN cannot achieve the state-of-the-art character spelling accuracy and ITR for a P300 speller with a small number of sensors. This study shows that the reason for this is that EoCNN has the problem of putting equal importance on OSLN, OTLN, and OCLNN in the ensemble processing of the outputs from these three CNNs (see Section 4.1.4 as well as Equation (4.1) and (4.2)) for the P300 speller irrespective of the number of sensors used to acquire EEG signals.

- In order to solve the problem of EoCNN, mentioned in the above contribution, we propose an improved EoCNN for the P300 speller called PEoCNN. In PEoCNN, first, we parameterize the ensemble processing of the outputs from OSLN, OTLN, and OCLNN. Then, we use the Sequential Model-based Algorithm Configuration (SMAC) [HHLB11] to automatically find and set values for the parameters, used in the parameterized ensemble processing of PEoCNN, depending on the number of sensors utilized in the P300 speller. In this way, PEoCNN is able to adopt/configure the importance of using the outputs from OSLN, OTLN, and OCLNN for the P300 speller depending on the number of sensors that are utilized.

- Experiments on three benchmark datasets show that, when using our PEoCNN for the P300 speller, the state-of-the-art spelling accuracy can be achieved in a BCI system with less than or equal to 16 sensors to acquire EEG signals in most cases. In addition, the state-of-the-art max-ITR[1] of the P300 speller can be achieved in a BCI system with less than 16 sensors to acquire EEG signals.

The rest of this chapter is organized as follows. Section 6.1 presents our study on the EoCNN-based P300 speller with different number of sensors in order to analyze and find the reason why EoCNN cannot achieve the state-of-the-art spelling accuracy

---

[1] The notion of max-ITR is explained in Section 3.3.5.

and max-ITR for a P300 speller with a small number of sensors. Section 6.2 introduces our approach to solve the problem of EoCNN revealed by the aforementioned study. Section 6.3 describes the experimental evaluation of our approach to show that by using our approach, we are able to achieve the state-of-the-art character spelling accuracy and max-ITR of the P300 speller with less than or equal to 16 sensors to acquire EEG signals. Section 6.4 ends this chapter with conclusions.

6.1 Study on EoCNN-based P300 Speller with Different Number of Sensors

In this section, we perform a study on the EoCNN-based P300 speller with different number of sensors in order to find the reason why EoCNN cannot achieve the state-of-the-art character spelling accuracy and ITR for a P300 speller with a small number of sensors. In this study, we perform experiments to examine the character spelling accuracy and the max-ITR achieved by EoCNN as well as the three CNNs used in the EoCNN, i.e., OCLNN, OTLN, and OSLN, for the P300 speller with different number of sensors. First, we describe the experimental setup of this study in Section 6.1.1. Then, we show and analyze the experimental results of this study in Section 6.1.2.

6.1.1 Experimental Setup

In this study, we use four implementations of the P300 speller to perform the experiments: the EoCNN-based P300 speller, the OCLNN-based P300 speller, the OSLN-based P300 speller, and the OTLN-based P300 speller. In order to examine the character spelling accuracy and the max-ITR of the aforementioned four P300 speller implementations when different number of sensors are utilized to acquire EEG signals, we perform the following two steps:

Step 1. We select different appropriate sensor subsets, containing different number of sensors, from an initial large set of sensors to acquire EEG signals for a subject who uses a P300 speller. The subject in this study is the subject used to acquire the EEG signals in Dataset III-A. We call this subject Subject III-A. We apply our SLES method (proposed and presented in Chapter 5) to select different appropriate sensor subsets from an initial large set of 64 sensors for Subject III-A. Therefore, we can use the training dataset of Dataset III-A to apply our SLES sensor selection method. More specifically, for our SLES method, this training dataset is used to train the $OSLN_{(S)}$ and calculate $score_j$ in each iteration of our SLES method (see Algorithm 1 in Chapter 5). For the details of the setup for our SLES please refer to the last paragraph in Section 5.3.1.

Step 2. After using our SLES method to select different sensor subsets for Subject III-A, we calculate the spelling accuracy and the max-ITR of the aforementioned four P300 speller implementations with the selected sensor subsets using Dataset III-A. The training dataset of Dataset III-A is used to train the CNN-based classifiers used in the aforementioned P300 speller implementations with the selected sensor subsets. Then, the test dataset of Dataset III-A is used to calculate the spelling accuracy and max-ITR of the aforementioned P300 speller implementations with the selected sensor subsets. The spelling accuracy $acc^{m}_{char(k)}$ is calculated using Equation (5.3) in Section 5.3.1, where $acc^{m}_{char(k)}$ denotes the spelling accuracy achieved when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m sensors. The ITR ITR^{m}_{k} is calculated using Equation (6.1) and (2.35), where ITR^{m}_{k} denotes the ITR achieved when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m sensors; N_{cla} =36 because we have 36 possible characters to spell (see Figure 2.10); $acc^{m}_{char(k)}$ is calculated using Equation (5.3); and T_k is calculated using Equation (2.35). After the calculation of ITR^{m}_{k}, we calculate the max-ITR $maxITR^{m}$ using Equation (6.2), where $maxITR^{m}$ denotes the max-ITR achieved when the EEG signals are acquired with the selected sensor subset containing m sensors.

$$ITR^{m}_{k} = \frac{60(acc^{m}_{char(k)} \log_2(acc^{m}_{char(k)}) + (1 - acc^{m}_{char(k)}) \log_2(\frac{1-acc^{m}_{char(k)}}{N_{cla}-1}) + \log_2(N_{cla}))}{T_k} \tag{6.1}$$

$$maxITR^{m} = \max_{1 \le k \le 15} \{ITR^{m}_{k}\} \tag{6.2}$$

6.1.2 Experimental Results

The experimental results on the spelling accuracy and the max-ITR of the EoCNN-based P300 speller, the OSLN-based P300 speller, the OTLN-based P300 speller, and the OCLNN-based P300 speller when different number of sensors m is used to acquire Subject III-A's EEG signals are shown in Figure 6.1 and Figure 6.2, respectively. Figure 6.1 shows that, in most cases (with respect to the number of sensors m), the OCLNN-based P300 speller achieves higher spelling accuracy than the OTLN-based P300 speller and the OSLN-based P300 speller. When the number of sensors used to acquire EEG signals is between 1 and 36, the OTLN-based P300 speller achieves higher spelling accuracy than the OSLN-based P300 speller. When the number of sensors used to acquire EEG signals is between 37 and 64, the OSLN-based P300 speller achieves higher spelling accuracy than the OTLN-based P300 speller. Figure 6.2

shows that, in most cases (with respect to the number of sensors m), the OCLNN-based P300 speller achieves higher max-ITR than the OTLN-based P300 speller and the OSLN-based P300 speller. When the number of sensors used to acquire EEG signals is between 1 and 30, the OTLN-based P300 speller achieves higher max-ITR than the OSLN-based P300 speller. When the number of sensors used to acquire EEG signals is between 32 and 64, the OSLN-based P300 speller achieves higher max-ITR than the OTLN-based P300 speller.

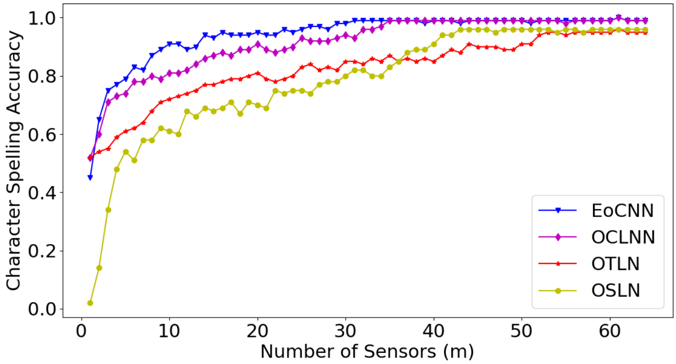


Figure 6.1: Spelling accuracy of different P300 speller implementations when different number of sensors m is used to acquire EEG signals.

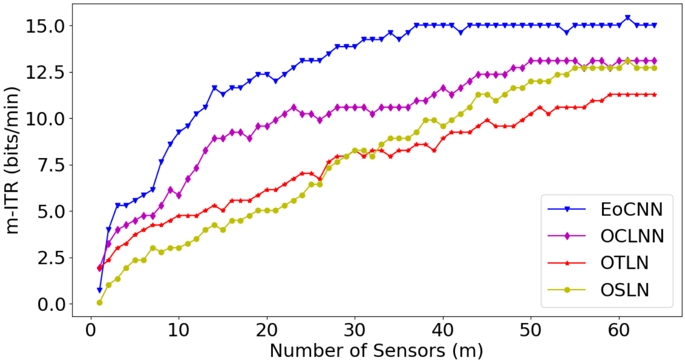


Figure 6.2: max-ITR of different P300 speller implementations when different number of sensors m is used to acquire EEG signals.

The aforementioned experimental results reveal that overall, the three CNNs, i.e., OCLNN, OSLN, and OTLN, have different importance and impact on the spelling accuracy and the max-ITR of a P300 speller depending on the number of sensors that are used to acquire EEG signals. This implies that when we use a CNN, which combines the outputs of OCLNN, OSLN, and OTLN, for a P300 speller, we should adopt/configure the importance of using the outputs from OCLNN, OSLN, and OTLN depending on the number of sensors that are used to acquire EEG signals. Unfortunately, the EoCNN (presented and proposed in Chapter 4) has the issue of putting equal importance on OSLN, OTLN, and OCLNN in the ensemble processing of the outputs from these three CNNs (see Section 4.1.4 as well as Equation (4.1) and (4.2)) for the EoCNN-based P300 speller irrespective of the number of sensors used to acquire EEG signals.

6.2 Our Solution Approach

In this section, in order to address the issue of EoCNN revealed in Section 6.1, we present our solution approach on how to make EoCNN adopt/configure the importance of using the outputs from OSLN, OTLN, and OCLNN for a P300 speller depending on the number of sensors used to acquire EEG signals. In our approach, first, we parameterize the ensemble processing of EoCNN as described in Section 6.2.1. Then, we find and set values for the parameters, that are used in the ensemble processing of EoCNN, depending on the number of sensors used to acquire EEG signals in the P300 speller as described in Section 6.2.2.

6.2.1 Parameterized Ensemble Processing

Our approach is based on EoCNN that is proposed and presented in Chapter 5. We use the architecture of EoCNN, i.e., the ensemble of OTLN, OSLN, and OCLNN, for the P300 speller (see Figure 4.1). However, the difference is in the ensemble processing of the outputs from OTLN, OSLN, and OCLNN. That is, EoCNN puts equal importance on OSLN, OTLN, and OCLNN in the ensemble processing of the outputs from OSLN, OTLN, and OCLNN (see Equation (4.1) in Section 4.1.4) irrespective of the number of sensors used to acquire EEG signals. In contrast, our approach here parameterizes the ensemble processing of the outputs from OSLN, OTLN, and OCLNN for the P300 speller in order to make this ensemble processing adaptable/configurable to the number of sensors used to acquire EEG signals.

Our parameterized ensemble processing of the outputs from OSLN, OTLN, and OCLNN is shown in Equation (6.3). We call EoCNN, with this parameterized ensemble processing, PEoCNN. In Equation (6.3), for epoch i and for intensification j, $P^1_{PEoC}(i,j)$ denotes the predicted probability by PEoCNN for class “P300”; $P^1_{OT}(i,j)$ denotes the predicted probability by OTLN for class “P300”; $P^1_{OS}(i,j)$ denotes the predicted probability by OSLN for class "P300"; $P^1_{OCL}(i,j)$ denotes the predicted probability by OCLNN for class "P300"; and p_1, p_2, and p_3 are three parameters that weight the importance of the predicted probability by OTLN, OSLN, OCLNN, respectively, for class "P300", in order to determine the output $P^1_{PEoC}(i,j)$ of PEoCNN. $p_1 \in [0,1]$, $p_2 \in [0,1]$, and $p_3 \in [0,1]$. In addition, we set the constraint p_1 + p_2 + p_3 = 1 to guarantee that $P^1_{PEoC}(i,j)$ is in the range [0, 1] because $P^1_{PEoC}(i,j)$ is the probability predicted by PEoCNN for class "P300".

$$P^1_{PEoC}(i,j) = p_1 \times P^1_{OT}(i,j) + p_2 \times P^1_{OS}(i,j) + p_3 \times P^1_{OCL}(i,j) \tag{6.3}$$

How we select appropriate values for p_1, p_2, and p_3, depending on the number of sensors used in the P300 speller for acquisition of EEG signals, is described in Section 6.2.2. After the selection of values for p_1, p_2, and p_3, we use $P^1_{PEoC}(i,j)$, i.e., the output of PEoCNN for class "P300", to calculate the position of the target character in the character matrix shown in Figure 2.10. For the detailed calculation process, please refer to Section 2.4.2, Equation (2.30), (2.31), and (2.32).

6.2.2 Parameter Configuration for Parameterized Ensemble Processing

As described in Section 6.2.1, in the parameterized ensemble processing of PEoCNN, there are three parameters, i.e., p_1, p_2, and p_3, that need to be configured (see Equation (6.3)). This section describes how we select appropriate values for these three parameters depending on the number of sensors used for the acquisition of EEG signals in the P300 speller.

For a given number of sensors m used to acquire EEG signals in the P300 speller, we select a set of appropriate values for p_1, p_2, and p_3 by looking at this selection problem as an optimization problem. We define this optimization problem as shown in Equation (6.4), where **p** is a vector of p_1, p_2, and p_3, i.e., $\mathbf{p} = [p_1, p_2, p_3]$; $Q(\mathbf{p})$ is the cost function. We define $Q(\mathbf{p})$ using Equation (6.5), where $maxITR_{ta}$ denotes the theoretically achievable maximum ITR of a P300 speller; $maxITR^m(\mathbf{p})$ denotes the max-ITR achieved by PEoCNN, configured with **p**, for the P300 speller when the EEG signals are acquired using the given m sensors. $maxITR^m(\mathbf{p})$ is calculated using Equation (6.6), where E denotes the total number of epochs used in a P300 speller. Here, E=15 because we use 15 epochs in the P300 speller (see Section 2.5). $ITR^m_k(\mathbf{p})$ denotes the ITR achieved by PEoCNN, configured with **p**, when k epochs are used for the P300 speller and the EEG signals are acquired using the given m sensors. $ITR^m_k(\mathbf{p})$ is calculated using Equation (6.7),where N_{cla} =36 because we have 36 possible characters to spell (see Figure 2.10); T_k is calculated using Equation (2.35); $acc^m_{char(k)}(\mathbf{p})$ denotes the character spelling accuracy achieved by PEoCNN, configured with **p**, when k epochs are used for the P300 speller and the EEG signals are acquired using the given m sensors. $acc^{m}_{char(k)}(\mathbf{p})$ is calculated using Equation (6.8), where $N^{m}_{tc(k)}(\mathbf{p})$ denotes the number of correctly inferred characters by PEoCNN configured with **p** when using k epochs for each character and the EEG signals are acquired using the given m sensors, and S_c denotes the number of all spelled characters.

$$\begin{gathered} \underset{\mathbf{p}}{Minimize}\ Q(\mathbf{p}) \\ subject\ to:\ p_1 + p_2 + p_3 = 1, 0 \leq p_1 \leq 1, 0 \leq p_2 \leq 1, and\ 0 \leq p_3 \leq 1 \end{gathered} \tag{6.4}$$

$$Q(\mathbf{p}) = maxITR_{ta} - maxITR^{m}(\mathbf{p}) \tag{6.5}$$

$$maxITR^{m}(\mathbf{p}) = \underset{1 \leq k \leq E}{max} \{ITR^{m}_{k}(\mathbf{p})\} \tag{6.6}$$

$$ITR^{m}_{k}(\mathbf{p}) = \frac{60(acc^{m}_{char(k)}(\mathbf{p}) \log_2(acc^{m}_{char(k)}(\mathbf{p})) + (1 - acc^{m}_{char(k)}(\mathbf{p})) \log_2(\frac{1 - acc^{m}_{char(k)}(\mathbf{p})}{N_{cla} - 1}) + \log_2(N_{cla}))}{T_k} \tag{6.7}$$

$$acc_{char(k)}(\mathbf{p}) = \frac{N_{tc(k)^{m}}(\mathbf{p})}{S_c} \tag{6.8}$$

Equation (6.4) shows that we define the selection of values for $\mathbf{p} = [p_1, p_2, p_3]$ as a single-objective optimization problem. In this optimization problem, we aim at finding **p** such that the cost function $Q(\mathbf{p})$ is minimized. By using this cost function, we aim at finding appropriate **p** to configure PEoCNN such that for a given m sensors used to acquire EEG signals, the max-ITR achieved by PEoCNN, i.e., $maxITR^{m}(\mathbf{p})$, is the closest possible to the theoretically achievable maximum ITR $maxITR_{ta}$. Typically, the ultimate goal of designing methods for the P300 speller is to increase the max-ITR in order to bring it closer to the theoretically achievable maximum ITR (for detailed discussion please see Section 3.3.5 and Section 4.2.4.). Thus, we define the cost function, shown in Equation (6.5), to find appropriate **p** to configure PEoCNN.

We use the Sequential Model-based Algorithm Configuration (SMAC) [HHLB11] as an optimization algorithm to solve the aforementioned single-objective optimization problem defined by Equation (6.4) because SMAC is currently one of the best-performing and versatile optimization algorithms for parameter configuration. For more details on SMAC please refer to [HHLB11]. In the optimization process of SMAC, the cost function $Q(\mathbf{p})$ is calculated as follows. We use a dataset to train PEoCNN configured by $\mathbf{p}$ selected by SMAC. The training process of PEoCNN is the same as the training process of EoCNN described in Section 4.1.3. Then, we run this trained PEoCNN on another dataset to calculate $Q(\mathbf{p})$ using Equation (6.5), (6.6), (6.7), and (6.8).

6.3 Experimental Evaluation

In this section, we present the experiments, we have performed, to determine and evaluate the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art character spelling accuracy and max-ITR. The goal is to demonstrate that: 1) by using our PEoCNN, we are able to achieve the state-of-the-art character spelling accuracy and max-ITR of the P300 speller when using less than or equal to 16 sensors to acquire EEG signals; 2) our solution approach, described in Section 6.2, is effective, i.e., the PEoCNN-based P300 speller needs less number of sensors to acquire EEG signals than the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy and max-ITR. First, we describe the experimental setup in Section 6.3.1. Then, in Section 6.3.2, we show and analyze the obtained experimental results for the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller.

6.3.1 Experimental Setup

We perform the following three steps in order to compare the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art character spelling accuracy and max-ITR.

Step 1. We select different appropriate sensor subsets, containing different number of sensors m, from an initial large set of sensors to acquire EEG signals for a subject who uses a P300 speller. The subjects in our experiments is the subjects used to acquire the EEG signals in Dataset II, III-A, and III-B. We call the subjects in Dataset II, III-A, and III-B, Subject II, Subject III-A, and Subject III-B, respectively. We apply our SLES method (proposed and presented in Chapter 5) to select different appropriate sensor subsets from an initial large set of 64 sensors for Subject II, Subject III-A, and Subject III-B. Therefore, we can use the training dataset of Dataset II, III-A, and III-B to apply our SLES sensor selection method. More specifically, for our SLES method, these training datasets are used to train the $OSLN_{(S)}$ and calculate

$score_j$ in each iteration of our SLES method (see Algorithm 1 in Chapter 5). For the details of the setup for our SLES please refer to the last paragraph in Section 5.3.1.

Step 2. We build two P300 speller implementations, namely the PEoCNN-based P300 speller and the EoCNN-based P300 speller. For the PEoCNN-based P300 speller, depending on given subset of m sensors selected to acquire EEG signals in **Step 1**, we select values for p_1, p_2, and p_3, which are used to configure PEoCNN. As described in Section 6.2.2, we use SMAC to select appropriate values for p_1, p_2, and p_3. When using SMAC, we need to calculate $Q(\mathbf{p})$ which is the cost function minimized by SMAC (see our defined optimization problem in Equation (6.4)). We use the training dataset in Dataset II, III-A, and III-B to calculate $Q(\mathbf{p})$. We divide each training dataset in Dataset II, III-A, and III-B into two parts: the first sub-dataset containing 60% of a given training dataset and the second sub-dataset containing the left 40% of the given training dataset (for the reason of using 60% and 40% to split a dataset please refer to Section 5.4.1.). The first sub-dataset is used to train PEoCNN configured with $\mathbf{p} = [p_1, p_2, p_3]$, selected by SMAC, in the PEoCNN-based P300 speller with the selected m sensors to acquire EEG signals. We run the trained PEoCNN on the second sub-dataset to calculate $Q(\mathbf{p})$ using Equation (6.5), (6.6), (6.7), and (6.8).

Step 3. After selecting $\mathbf{p}$ and configuring PEoCNN, we calculate the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art character spelling accuracy and max-ITR. Firstly, we calculate the spelling accuracy and the max-ITR of the PEoCNN-based P300 speller and EoCNN-based P300 speller with the different selected sensor subsets from **Step 1** using Dataset II, III-A, and III-B. The spelling accuracy $acc^{m}_{char(k)}$ is calculated using Equation (5.3) in Section 5.3.1, where $acc^{m}_{char(k)}$ denotes the spelling accuracy achieved when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m sensors. The ITR ITR^{m}_{k} is calculated using Equation (6.1) and (2.35), where ITR^{m}_{k} denotes the ITR achieved when using the first k epochs for each character and using the EEG signals acquired with the selected sensor subset containing m sensors. After the calculation of ITR^{m}_{k}, we calculated the max-ITR $maxITR^{m}$ using Equation (6.2), where $maxITR^{m}$ denotes the max-ITR achieved when using the EEG signals acquired with the selected sensor subset containing m sensors. Secondly, after the calculation of $acc^{m}_{char(k)}$ and $maxITR^{m}$, we calculate the minimal number of sensors needed to acquire EEG signals without losing the state-of-the-art spelling accuracy and max-ITR of the P300 speller. We use m^{acc}_{min} to denote the minimal number of sensors needed to acquire EEG signals without losing the state-of-the-art spelling accuracy. m^{acc}_{min} is calculated as the minimal $m \in [1, 64]$ which makes $acc^{m}_{char(k)} >= acc^{soa}_{char(k)}$. Here, $acc^{soa}_{char(k)}$ denotes the state-of-the-art spelling accuracy of the P300 speller when using k epochs, i.e., the spelling accuracy achieved by EoCNN using k epochs when 64 sensors are used to acquire EEG signals in the P300 speller. We use m_{min}^{itr} to denote the minimal number of sensors needed to acquire EEG signals without losing the max-ITR of the P300 speller. m_{min}^{itr} is calculated as the minimal $m \in [1, 64]$ which makes $maxITR^{m} >= maxITR^{soa}$. Here, $maxITR^{soa}$ denotes the state-of-the-art max-ITR of the P300 speller, i.e., the max-ITR achieved by EoCNN when 64 sensors are used to acquire EEG signals in the P300 speller.

6.3.2 Experimental Results

In this section, we present the experimental results, we have obtained, for the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy (see Section 6.3.2.1) as well as the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art max-ITR (see Section 6.3.2.2).

6.3.2.1 Minimal Number of Sensors Without Losing State-of-the-art Spelling Accuracy

Table 6.1, 6.2 and 6.3 show the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy for epoch $k \in [1, 15]$. The first column in the tables lists the different CNNs used in a P300 speller for the inference of the characters. Each row provides the minimal number of sensors needed to acquire EEG signals used to acquire EEG signals in a P300 speller for different epoch numbers $k \in [1, 15]$. A number in bold indicates that the minimal number of sensors needed to acquire EEG signals in the P300 speller based on the corresponding CNN is lower than or equal to the minimal number of sensors needed to acquire EEG signals in the P300 speller based on the other CNN.

Table 6.1: Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset II.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PEoCNN	**10**	**13**	**20**	**15**	**17**	**8**	**8**	**3**	**3**	**3**	**3**	**4**	**3**	**3**	**3**
EoCNN	**10**	16	28	19	22	9	**8**	**3**	**3**	**3**	**3**	**4**	**3**	**3**	**3**

Table 6.2: Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset III-A.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PEoCNN	**12**	**14**	**14**	**12**	**19**	**17**	**11**	**24**	**29**	**16**	**30**	**13**	**13**	**15**	**18**
EoCNN	18	39	37	21	41	38	20	39	44	23	46	33	21	16	31

Table 6.3: Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art spelling accuracy of the P300 speller on Dataset III-B.

	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PEoCNN	**14**	**13**	**15**	**23**	**12**	**10**	**13**	**19**	**22**	**21**	**24**	**15**	**18**	**16**	**11**
EoCNN	25	26	28	27	23	14	16	25	30	29	33	22	30	29	23

Table 6.1, 6.2, and 6.3 show that in all 45 cases (i.e., all epoch columns in the three tables), the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller is lower than or equal to the minimal number of sensors needed to acquire EEG signals in the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy. This demonstrates that our solution approach, described in Section 6.2, is effective, i.e., the PEoCNN-based P300 speller needs less number of sensors to acquire EEG signals than the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy.

In addition, Table 6.1, 6.2 and 6.3 show that for 31 different epoch numbers out of all 45 epoch numbers, the PEoCNN-based P300 speller can achieve the state-of-the-art spelling accuracy with less than or equal to 16 sensors to acquire EEG signals. In contrast, only for 15 different epoch numbers out of 45 epoch numbers, the EoCNN-based P300 speller can achieve the state-of-the-art spelling accuracy with less than or equal to 16 sensors to acquire EEG signals. When a P300 speller is configured with different epoch numbers, the P300 speller has different spelling accuracy and communication speed: typically, a P300 speller, configured with a large epoch number, has a high spelling accuracy but a low communication speed while a P300 speller, configured with a small epoch number, has a low spelling accuracy but a high communication speed. The PEoCNN-based P300 speller has more configurations, in terms of the epoch numbers, than the EoCNN-based P300 speller when used in a low-complexity BCI systems with less than or equal to 16 sensors to acquire EEG signals. Thus, the PEoCNN-based P300 speller has more options to trade off the spelling accuracy for the communication speed and vice versa than the EoCNN-based P300 speller when used in such low-complexity BCI systems.

6.3.2.2 Minimal Number of Sensors Without Losing State-of-the-art max-ITR

Table 6.4 shows the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller and in the EoCNN-based P300 speller without losing the state-of-the-art max-ITR for Dataset II, III-A, and III-B. In this table, the first column lists the different CNNs used in a P300 speller for the inference of the characters. Each row provides the minimal number of sensors needed to acquire EEG signals in a P300 speller without losing the state-of-the-art max-ITR for the different datasets. A number in bold indicates that the minimal number of sensors needed to acquire EEG signals in the P300 speller based on the corresponding CNN is lower than or equal to the minimal number of sensors needed to acquire EEG signals in the P300 speller based on the other CNN.

Table 6.4: Minimal number of sensors needed to acquire EEG signals in the P300 speller based on different CNNs without losing the state-of-the-art max-ITR on Dataset II, III-A, and III-B.

	Dataset II	Dataset III-A	Dataset III-B
PEoCNN	**10**	**14**	**13**
EoCNN	**10**	37	26

Table 6.4 shows that for all three datasets, the minimal number of sensors needed to acquire EEG signals in the PEoCNN-based P300 speller is less than the minimal number of sensors needed to acquire EEG signals in the EoCNN-based P300 speller without losing the state-of-the-art max-ITR. This demonstrates again that our solution approach, described in Section 6.2, is effective, i.e., the PEoCNN-based P300 speller needs less number of sensors to acquire EEG signals than the EoCNN-based P300 speller without losing the state-of-the-art max-ITR.

Moreover, Table 6.4 shows that for all three datasets, the PEoCNN-based P300 speller achieves the state-of-the-art max-ITR when using less than 16 sensors to acquire EEG signals. In contrast, for only one dataset, the EoCNN-based P300 speller achieves the state-of-the-art max-ITR when using less than 16 sensors to acquire EEG signals. This demonstrates that by using our solution approach, described in Section 6.2, we enhance the usability of a P300 speller, having the state-of-the-art max-ITR, on low-complexity BCI systems across different subjects.

6.4 Conclusions

In this chapter, we present our research on how to achieve the state-of-the-art character spelling accuracy and max-ITR of the P300 speller with popular low-complexity and relatively cheap BCI systems that use less than or equal to 16 sensors to acquire EEG signals. We perform a study on the EoCNN-based P300 speller with different number of sensors to show that EoCNN has the problem of putting equal importance on using OSLN, OTLN, and OCLNN for the P300 speller irrespective of the number of sensors used to acquire EEG signals. In order to solve this problem, we propose an improved EoCNN called PEoCNN. In PEoCNN, we parameterize the ensemble processing of the outputs from OSLN, OTLN, and OCLNN. Then, we use SMAC to select appropriate values for the parameters depending on the number of sensors utilized in the P300 speller. Experimental results on three benchmark datasets show that by using our PEoCNN, we are able to achieve the state-of-the-art performance, in terms of the character spelling accuracy and the max-ITR, of the P300 speller when using less than or equal to 16 sensors to acquire EEG signals. Moreover, our proposed PEoCNN enhances the usability of a P300 speller, having the state-of-the-art performance, on low-complexity BCI systems across different subjects.

Chapter 7

Summary and Conclusions

A P300-based Brain Computer Interface (BCI) character speller, also known as P300 speller, has been an important communication pathway, under extensive research, for people who lose motor ability, such as patients with Amyotrophic Lateral Sclerosis (ALS) or spinal-cord injury because a P300 speller allows human-beings to directly spell characters using eye-gazes, thereby building communication between the human brain and a computer. Unfortunately, P300 spellers are still not used in human's daily life and remain in an experimental stage at research labs. The reason for this situation is that the performance and the efficiency of current P300 spellers are unacceptably low for BCI users in their daily life. Therefore, in this thesis, we have focused our attention on developing high performance and efficient P300 spellers in order to bring P300 spellers into practical use. More specifically, in order to increase the performance of a P300 speller, we have developed methods to increase the character spelling accuracy and the Information Transfer Rate (ITR). In order to improve the efficiency of a P300 speller, we have developed methods to reduce the number of sensors needed to acquire EEG signals as well as to reduce the complexity of the classifier used in a P300 speller without losing the performance.

We summarize the contributions of each chapter of this thesis in Figure 7.1 in order to show how the proposed methods in each chapter improve the performance and/or the efficiency of a P300 speller. In this figure, BA denotes our baseline, i.e., the CNN, called BN3 [LWG$^+$18], used for a P300 speller. We select this baseline because BN3 achieves better spelling accuracy and ITR than other state-of-the-art methods (excluding our proposed methods) for the P300 speller. CH3, CH4, CH5, and CH6 denote our proposed methods in Chapter 3, Chapter 4, Chapter 5, and Chapter 6, respectively. The "Performance" axis in Figure 7.1 shows the max-ITR[1] of a P300

---

[1] The notion of max-ITR is introduced in Section 3.3.5

speller. In this axis, TA shows the theoretically achievable maximum ITR[2] of a P300 speller. The "Cost" axis shows the number of sensors used to acquire EEG signals in a P300 speller. The "Complexity" axis shows the number of parameters (i.e., weights and biases) of a CNN used as the classifier in a P300 speller. Based on Figure 7.1, we summarize and draw the following conclusions for each chater's contributions:

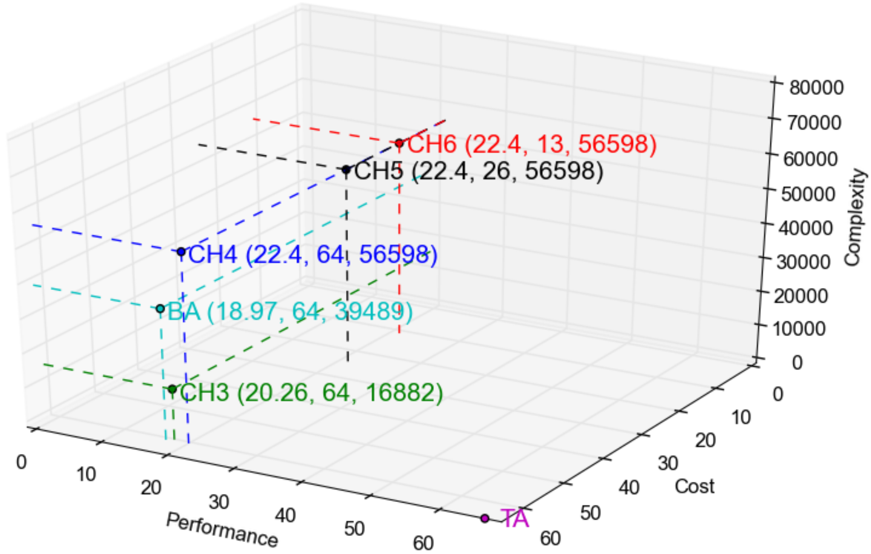


Figure 7.1: Overview of how each chapter's contributions improve the performance and/or the efficiency of a P300 speller.

- **Chapter 3 (CH3):** In order to improve the performance and the efficiency (i.e., to reduce the complexity) of a P300 speller with respect to BA, in Chapter 3, we have proposed a simple, yet effective CNN architecture, called One Convolution Layer Neural Network (OCLNN), for the P300 speller. This CNN has only one convolution layer which is the first layer of the network. This layer performs both a spatial convolution and a temporal convolution at the same time, thereby learning very useful P300-related features from both raw temporal information and raw spatial information. Our OCLNN exhibits very low network complexity because it uses only one

---

[2] The theoretically achievable maximum ITR is discussed in Section 2.4.3

convolution layer and does not use fully-connected layers before the output layer. Figure 7.1 shows that, compared to the baseline BA, by using our OCLNN (see CH3 in Figure 7.1), we have improved the performance, in terms of the spelling accuracy and the ITR, of the P300 speller, as well as we have improved significantly the efficiency, i.e., the complexity of the CNN used in the P300 speller is reduced.

- **Chapter 4 (CH4):** The ITR achieved by our OCLNN (see CH3 in Figure 7.1) still cannot reach the theoretically achievable maximum ITR (see TA in Figure 7.1). Therefore, to increase the ITR of a P300 speller in order to bring it closer to the theoretically achievable maximum ITR, in Chapter 4, we have proposed an ensemble of CNNs for the P300 speller. Our proposed ensemble of CNNs is called Ensemble of Convolutional Neural Networks (EoCNN). EoCNN uses two novel CNNs, we have devised, called One Spatial Layer Network (OSLN) and One Temporal Layer Network (OTLN), respectively. OSLN and OTLN both have only one convolution layer. OTLN performs a temporal convolution in the first layer to learn P300-related separate temporal features. OSLN performs a spatial convolution in the first layer to learn P300-related separate spatial features. Our EoCNN uses the ensemble of OSLN and OTLN together with OCLNN (proposed in Chapter 3), thereby extracting more useful P300-related features than OCLNN alone. As a result, see CH4 in Figure 7.1, our EoCNN achieves higher character spelling accuracy and ITR than OCLNN (see CH3 in Figure 7.1) and other state-of-art methods (see BA in Figure 7.1) for the P300 speller. However, the complexity of our EoCNN is higher than the complexity of OCLNN. Thus, compared to OCLNN, by using our EoCNN, we have improved the performance, in terms of the spelling accuracy and the ITR, of the P300 speller but we have impaired the efficiency, i.e., the complexity of the CNN used in our EoCNN-based P300 speller has been increased.

- **Chapter 5 (CH5):** In order to improve the efficiency of our EoCNN-based P300 speller, in Chapter 5, we have proposed a sensor reduction method, called Spatial Learning based Elimination Selection (SLES), to reduce the number of sensors used to acquire EEG signals in the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy and ITR. Here, the state-of-the-art spelling accuracy and ITR denote the accuracy and ITR achieved by EoCNN when a large number of sensors (e.g., 64 sensors) is used to acquire EEG signals (see CH4 in Figure 7.1). Our SLES uses a novel parametrized CNN, we have devised, to evaluate and rank the sensors during the sensor selection process. This method features an iterative, parametrized, backward elimination algorithm to eliminate and select sensors. The parameter configured in this algorithm controls the training frequency of the CNN and the number of sensors to eliminate in every iteration. Our SLES method significantly reduces the number of sensors used in the EoCNN-based P300 speller without losing the state-of-the-art spelling accuracy and ITR (see CH5 in Figure 7.1). Thus, by using our SLES method, we have improved the efficiency, i.e., we have reduced significantly the number of sensors needed to acquire EEG signals in the EoCNN-based P300 speller without losing the state-of-the-art performance in terms of the spelling accuracy and ITR.

- **Chapter 6 (CH6):** Although the number of sensors needed to acquire EEG signals in the EoCNN-based P300 speller is significantly reduced by our SLES method (see CH5 in Figure 7.1), we still need to use more than 16 sensors to acquire EEG signals in the EoCNN-based P300 speller in most cases in order to preserve the state-of-the-art spelling accuracy and ITR. Unfortunately, popular low-complexity and relatively cheap (affordable) BCI systems utilize a small number of sensors for the acquisition of EEG signals. Typically, such small number of sensors is less than or equal to 16 sensors. Therefore, in Chapter 6, we have performed research on how to achieve the state-of-the-art spelling accuracy and ITR of the P300 speller with less than or equal to 16 sensors to acquire EEG signals. We have performed a study on the EoCNN-based P300 speller with different number of sensors, which reveals that EoCNN has the problem of putting equal importance on OSLN, OTLN, and OCLNN when combining the outputs from OSLN, OTLN, and OCLNN irrespective of the number of sensors used to acquire EEG signals. To solve this problem, we have proposed an improved EoCNN for the P300 speller called PEoCNN. In PEoCNN, first, we parameterize the process of combining the outputs from OSLN, OTLN, and OCLNN. Then, we use the Sequential Model-based Algorithm Configuration (SMAC) to automatically find and set values for the parameters depending on the number of sensors used in the P300 speller. In this way, PEoCNN adapts/configures the importance of using the outputs from OSLN, OTLN, and OCLNN for the P300 speller depending on the number of sensors used to acquire EEG signals. As a result, see CH6 in Figure 7.1, the PEoCNN-based P300 speller can be used in popular low-complexity BCI systems with less than 16 sensors to acquire EEG signals without losing the state-of-the-art spelling accuracy and ITR. Thus, compared to EoCNN (see CH5 in Figure 7.1), by using our PEoCNN, we have improved the efficiency, i.e., we have further reduced the number of sensors needed to acquire EEG signals in the P300 speller without losing the state-of-the-art performance in terms of the spelling accuracy and the ITR.

Bibliography

[AA$^+$16] Martín Abadi, Ashish Agarwal, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[Act] Brain Products ActiCHamp Webpage. `https://www.brainproducts.com/productdetails.php?id=42`. Accessed: 2019-02-16.

[B-A] B-Alert X10 Webpage. `https://www.advancedbrainmonitoring.com/xseries/x10/`. Accessed: 2019-05-17.

[BFL13] Luzheng Bi, Xin-An Fan, and Yili Liu. Eeg-based brain-controlled mobile robots: a survey. *IEEE transactions on human-machine systems*, 43(2):161–176, 2013.

[Bio18] Biosemi. Biosemi webpage, 2018. `https://www.biosemi.com/products.htm`, Last accessed on 2018-09-03.

[BKG$^+$00] Niels Birbaumer, Andrea Kubler, Nimr Ghanayim, Thilo Hinterberger, Jouri Perelmouter, Jochen Kaiser, Iver Iversen, Boris Kotchoubey, Nicola Neumann, and Herta Flor. The thought translation device (ttd) for completely paralyzed patients. *IEEE Transactions on rehabilitation Engineering*, 8(2):190–193, 2000.

[Bla03] B Blankertz. BCI competition II webpage. *http://www.bbci.de/competition/ii/*, 2003.

[Bla08] B Blankertz. BCI competition III webpage. *http://www.bbci.de/competition/iii/*, 2008.

[Bos04] Vladimir Bostanov. BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram. *IEEE Transactions on Biomedical Engineering*, 51(6):1057–1061, 2004.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[C$^+$15] François Chollet et al. Keras. *https://github.com/keras-team/keras*, 2015.

[CB64] Robert M Chapman and Henry R Bragdon. Evoked responses to numerical and non-numerical visual stimuli while problem solving. *Nature*, 203(4950):1155, 1964.

[CBJ16] Jinsung Chun, Byeonguk Bae, and Sungho Jo. Bci based hybrid interface for 3d object control in virtual reality. In *2016 4th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–4. IEEE, 2016.

[CCH+10] Andrew Campbell, Tanzeem Choudhury, Shaohan Hu, Hong Lu, Matthew K Mukerjee, Mashfiqui Rabbi, and Rajeev DS Raizada. Neurophone: brain-mobile phone interface using a wireless eeg headset. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, pages 3–8. ACM, 2010.

[CdBV+14] Yun-Hsuan Chen, Maaike de Beeck, Luc Vanderheyden, Evelien Carrette, Vojkan Mihajlović, Kris Vanstreels, Bernard Grundlehner, Stefanie Gadeyne, Paul Boon, and Chris Van Hoof. Soft, comfortable polymer dry electrodes for high quality ecg and eeg recording. *Sensors*, 14(12):23758–23780, 2014.

[CFF16] Tsan-Yu Chen, Chih-Wei Feng, and Wai-Chi Fang. Development of a reliable SSVEP-based BCI mobile dialing system. In *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, pages 269–272. IEEE, 2016.

[CG11] 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, 2011.

[CRC+10] Hubert Cecotti, Bertrand Rivet, Marco Congedo, Christian Jutten, Olivier Bertrand, Emmanuel Maby, and Jérémie Mattout. Suboptimal sensor subset evaluation in a P300 brain-computer interface. In *Signal Processing Conference, 2010 18th European*, pages 924–928. IEEE, 2010.

[CRC+11] Hubert Cecotti, Bertrand Rivet, Marco Congedo, Christian Jutten, Olivier Bertrand, Emmanuel Maby, and Jérémie Mattout. A robust sensor-selection method for P300 brain–computer interfaces. *Journal of neural engineering*, 8(1):016001, 2011.

[CRT+14] KA Colwell, DB Ryan, CS Throckmorton, EW Sellers, and LM Collins. Channel selection methods for the P300 speller. *Journal of neuroscience methods*, 232:6–15, 2014.

[CS16] Emanuele Cannella and Todor P Stefanov. Energy efficient semi-partitioned scheduling for embedded multiprocessor streaming systems. *Design Automation for Embedded Systems*, 20(3):239–266, 2016.

[Els09] JA Elshout. Review of brain-computer interfaces based on the p300 evoked potential. Master's thesis, 2009.

[EMO] EMOTIV EPOC+ Webpage. `https://www.emotiv.com/epoc/`. Accessed: 2019-02-16.

[FD88] 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.

[FRAG+12] Reza Fazel-Rezai, Brendan Z Allison, Christoph Guger, Eric W Sellers, Sonja C Kleih, and Andrea Kübler. P300 brain computer interface: current challenges and emerging trends. *Frontiers in neuroengineering*, 5:14, 2012.

[FTM+88] Steven F Faux, Michael W Torello, Robert W McCarley, Martha E Shenton, and Frank H Duffy. P300 in schizophrenia: confirmation and statistical validation of temporal region deficit in P300 topography. *Biological psychiatry*, 23(8):776–790, 1988.

[GDS+09] Christoph Guger, Shahab Daban, Eric Sellers, Clemens Holzner, Gunther Krausz, Roberta Carabalona, Furio Gramatica, and Guenter Edlinger. How many people are able to control a p300-based brain–computer interface (bci)? *Neuroscience letters*, 462(1):94–98, 2009.

[GG+10] Joseph T Gwin, Klaus Gramann, et al. Removal of movement artifact from high-density EEG recorded during walking and running. *Journal of Neurophysiology*, 103(6):3526–3534, 2010.

[g.H] g.HIamp Webpage. `http://www.gtec.at/Products/Hardware-and-Accessories/g.HIamp-Specs-Features`. Accessed: 2019-02-16.

[GP+13] Stephen William Gilroy, Julie Porteous, et al. A brain-computer interface to a plan-based narrative. In *International Joint Conference on Artificial Intelligence*, pages 1997–2005, 2013.

[GS06] Alan Gevins and Michael E Smith. Electroencephalography (eeg) in neuroergonomics. *Neuroergonomics: The brain at work*, pages 15–31, 2006.

[GSB+18] Violaine Guy, Marie-Helene Soriani, Mariane Bruno, Theodore Papadopoulo, Claude Desnuelle, and Maureen Clerc. Brain computer interface with the p300 speller: usability for disabled people with amyotrophic lateral sclerosis. *Annals of physical and rehabilitation medicine*, 61(1):5–11, 2018.

[GVF11] Cristian Grozea, Catalin D Voinescu, and Siamac Fazli. Bristle-sensors—low-cost flexible passive dry eeg electrodes for neurofeedback and bci applications. *Journal of neural engineering*, 8(2):025008, 2011.

[GZW10] Junfeng Gao, Chongxun Zheng, and Pei Wang. Online removal of muscle artifact from electroencephalogram signals based on canonical correlation analysis. *Clinical EEG and Neuroscience*, 41(1):53–59, 2010.

[Hay94] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[HCP02] CJ Harland, TD Clark, and RJ Prance. Remote detection of human electroencephalograms using ultrahigh input impedance electric potential sensors. *Applied Physics Letters*, 81(17):3284–3286, 2002.

[Her01] Christoph S Herrmann. Human eeg responses to 1–100 hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena. *Experimental brain research*, 137(3-4):346–353, 2001.

[HHLB11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, page 507–523, 2011.

[HVE06] Ulrich Hoffmann, Jean-Marc Vesin, and Touradj Ebrahimi. Spatial filters for the classification of event-related potentials. Technical report, 2006.

[HZRS16] 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.

[IKV18] Anti Ingel, Ilya Kuzovkin, and Raul Vicente. Direct information transfer rate optimisation for ssvep-based bci. *Journal of neural engineering*, 16(1):016016, 2018.

[Ins] EMOTIV Insight Webpage. `https://www.emotiv.com/insight/`. Accessed: 2019-05-17.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[IS16] Changkyun Im and Jong-Mo Seo. A review of electrodes for the electrical brain signal recording. *Biomedical Engineering Letters*, 6(3):104–112, 2016.

[JAB$^+$10] Jing Jin, Brendan Z Allison, Clemens Brunner, Bei Wang, Xingyu Wang, Jianhua Zhang, Christa Neuper, and Gert Pfurtscheller. P300 chinese input system based on bayesian lda. *Biomedizinische Technik/Biomedical Engineering*, 55(1):5–18, 2010.

[JK09] Garett D Johnson and Dean J Krusienski. Ensemble swlda classifiers for the p300 speller. In *International Conference on Human-Computer Interaction*, pages 551–557. Springer, 2009.

[KB07] Kanthaiah Koka and Walter G Besio. Improvement of spatial selectivity and decrease of mutual information of tri-polar concentric ring electrodes. *Journal of neuroscience methods*, 165(2):216–222, 2007.

[KMG$^+$04] Matthias Kaper, Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, and Helge Ritter. Bci competition 2003-data set iib: support vector machines for the p300 speller paradigm. *IEEE Transactions on biomedical Engineering*, 51(6):1073–1076, 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[KVDS16] JP Kulasingham, V Vibujithan, and AC De Silva. Deep belief networks and stacked autoencoders for the p300 guilty knowledge test. In *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 127–132. IEEE, 2016.

[LL+14] Chin-Teng Lin, Bor-Shyh Lin, et al. Brain computer interface-based smart living environmental auto-adjustment control system in UPnP home networking. *IEEE Systems Journal*, 8(2):363–370, 2014.

[LLJ+18] Lingyu Liang, Luojun Lin, Lianwen Jin, Duorui Xie, and Mengru Li. Scut-fbp5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1598–1603. IEEE, 2018.

[LSCS15] Di Liu, Jelena Spasic, Gang Chen, and Todor Stefanov. Energy-efficient mapping of real-time streaming applications on cluster heterogeneous mpsocs. In *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, pages 1–10. IEEE, 2015.

[LSWS16] Di Liu, Jelena Spasic, Peng Wang, and Todor Stefanov. Energy-efficient scheduling of real-time tasks on heterogeneous multicores using task splitting. In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 149–158. IEEE, 2016.

[LTK13] Chin-Teng Lin, Shu-Fang Tsai, and Li-Wei Ko. EEG-based learning system for online motion sickness level estimation in a dynamic vehicle environment. *IEEE Transactions on Neural Networks and Learning Systems*, 24(10):1689–1700, 2013.

[LWG16] Ke Lin, Yijun Wang, and Xiaorong Gao. Time-frequency joint coding method for boosting information transfer rate in an ssvep based bci system. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5873–5876. IEEE, 2016.

[LWG+18] 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.

[M+97] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[Mar] OPEN BCI Mark IV Webpage. `https://shop.openbci.com/collections/frontpage/products/ultracortex-mark-iv`. Accessed: 2019-05-17.

[MG15] Ran Manor and Amir B Geva. Convolutional neural network for multi-category rapid serial visual presentation BCI. *Frontiers in Computational Neuroscience*, 9, 2015.

[MPP08] Gernot R Muller-Putz and Gert Pfurtscheller. Control of an electrical prosthesis with an ssvep-based bci. *IEEE Transactions on Biomedical Engineering*, 55(1):361–364, 2008.

[MUS] MUSE Webpage. `https://choosemuse.com/muse/`. Accessed: 2019-05-17.

[MWV+10] Maarten Mennes, Heidi Wouters, Bart Vanrumste, Lieven Lagae, and Peter Stiers. Validation of ICA as a tool to remove eye movement artifacts from EEG/ERP. *Psychophysiology*, 47(6):1142–1150, 2010.

[Nie15] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[NRS17] Sebastian Nagel, Wolfgang Rosenstiel, and Martin Spüler. Random visual evoked potentials (rvep) for brain-computer interface (bci) control. *compare*, 250(250ms):250ms, 2017.

[NS17] Sobhan Niknam and Todor Stefanov. Energy-efficient scheduling of throughput-constrained streaming applications by periodic mode switching. In *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 203–212. IEEE, 2017.

[ONB+08] Martin Oehler, Peter Neumann, Matthias Becker, Gabriel Curio, and Meinhard Schilling. Extraction of ssvep signals of a capacitive eeg helmet for human machine interface. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4495–4498. IEEE, 2008.

[PHP10] Konstantinos J Panoulas, Leontios J Hadjileontiadis, and Stavros M Panas. Brain-computer interface (bci): Types, processing perspectives and applications. In *Multimedia Services in Intelligent Environments*, pages 299–321. Springer, 2010.

[Pic92] Terence W Picton. The p300 wave of the human event-related potential. *Journal of clinical neurophysiology*, 9(4):456–479, 1992.

[PN01] Gert Pfurtscheller and Christa Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[PNCB11] Gabriel Pires, Urbano Nunes, and Miguel Castelo-Branco. Statistical spatial filtering for a P300-based BCI: tests in able-bodied, and patients with cerebral palsy and amyotrophic lateral sclerosis. *Journal of neuroscience methods*, 195(2):270–281, 2011.

[Pol07] John Polich. Updating P300: an integrative theory of P3a and P3b. *Clinical Neurophysiology*, 118(10):2128–2148, 2007.

[Qiu] Xipeng Qiu. *Neural networks and deep learning*. `https://nndl.github.io/`. Accessed: 2019-07-29.

[Qui] Quick-8 Webpage. `https://www.cognionics.net/products`. Accessed: 2019-05-17.

[RCMM12] Bertrand Rivet, Hubert Cecotti, Emmanuel Maby, and Jérémie Mattout. Impact of spatial filters during sensor selection in a visual p300 brain-computer interface. *Brain topography*, 25(1):55–63, 2012.

[RCP$^+$10] Bertrand Rivet, Hubert Cecotti, Ronald Phlypo, Olivier Bertrand, Emmanuel Maby, and Jérémie Mattout. Eeg sensor selection by sparse spatial filtering in p300 speller brain-computer interface. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 5379–5382. IEEE, 2010.

[RCS$^+$11] Bertrand Rivet, Hubert Cecotti, Antoine Souloumiac, Emmanuel Maby, and Jérémie Mattout. Theoretical analysis of xdawn algorithm: application to an efficient sensor selection in a p300 bci. In *2011 19th European Signal Processing Conference*, pages 1382–1386. IEEE, 2011.

[RG08] 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.

[RSAG09] Bertrand Rivet, Antoine Souloumiac, Virginie Attina, and Guillaume Gibert. xDAWN algorithm to enhance evoked potentials: application to brain–computer interface. *IEEE Transactions on Biomedical Engineering*, 56(8):2035–2043, 2009.

[RSG$^+$09] Bertrand Rivet, Antoine Souloumiac, Guillaume Gibert, Virginie Attina, and Olivier Bertrand. Sensor selection for P300 speller brain computer interface. In *ESANN*, 2009.

[Sam67] Arthur L Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.

[SD06] Eric W Sellers and Emanuel Donchin. A P300-based brain–computer interface: initial tests by ALS patients. *Clinical Neurophysiology*, 117(3):538–548, 2006.

[SDC07] Thomas J Sullivan, Stephen R Deiss, and Gert Cauwenberghs. A low-noise, non-contact eeg/ecg sensor. In *2007 IEEE Biomedical Circuits and Systems Conference*, pages 154–157. IEEE, 2007.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SL11] Gerwin Schalk and Eric C Leuthardt. Brain-computer interfaces using electrocorticographic signals. *IEEE reviews in biomedical engineering*, 4:140–154, 2011.

[SLS16] Jelena Spasic, Di Liu, and Todor Stefanov. Energy-efficient mapping of real-time applications on heterogeneous mpsocs using task replication. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 1–10. IEEE, 2016.

[SLS18] Hongchang Shan, Yu Liu, and Todor Stefanov. A simple convolutional neural network for accurate P300 detection and character spelling in brain computer interface. In *Proceedings of 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 1604–1610, 2018.

[SMH$^+$04] Gerwin Schalk, Dennis J McFarland, Thilo Hinterberger, Niels Birbaumer, and Jonathan R Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043, 2004.

[Spü15] Martin Spüler. A brain-computer interface (bci) system to use arbitrary windows applications by directly controlling mouse and keyboard. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1087–1090. IEEE, 2015.

[SRC$^+$12] Pietro Salvo, Robrecht Raedt, Evelien Carrette, David Schaubroeck, Jan Vanfleteren, and Ludwig Cardon. A 3d printed dry electrode for ecg/eeg recording. *Sensors and Actuators A: Physical*, 174:96–102, 2012.

[SS09] 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.

[SW49] Claude E Shannon and Warren Weaver. The mathematical theory of communication (urbana, il, 1949.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[VELB18] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.

[Wal16] Stephan Waldert. Invasive vs. non-invasive neuronal signals for brain-machine interfaces: will one prevail? *Frontiers in neuroscience*, 10:295, 2016.

[WRMP98] Jonathan R Wolpaw, Herbert Ramoser, Dennis J McFarland, and Gert Pfurtscheller. Eeg-based communication: improved accuracy by response verification. *IEEE transactions on Rehabilitation Engineering*, 6(3):326–333, 1998.

[WW12] Jonathan Wolpaw and Elizabeth Winter Wolpaw. *Brain-computer interfaces: principles and practice*. OUP USA, 2012.

[WWJ11] Yu-Te Wang, Yijun Wang, and Tzyy-Ping Jung. A cell-phone-based brain–computer interface for communication in daily life. *Journal of Neural Engineering*, 8(2):025018, 2011.

[XMYR16] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

List of Publications

1. **Hongchang Shan**, Yu Liu, and Todor Stefanov,
"A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface",
In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, pp. 1604-1610, Stockholm, Sweeden, July 13-19, 2018.

2. **Hongchang Shan**, and Todor Stefanov,
"SLES: A Novel CNN-based Method for Sensor Reduction in P300 Speller,"
In *Proceedings of the 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'19)*, pp. 3026-3031, Berlin, Germany, July 23-27, 2019.

3. **Hongchang Shan**, and Todor Stefanov,
"A Novel Sensor Selection Method based on Convolutional Neural Network for P300 Speller in Brain Computer Interface",
The 56th ACM/IEEE Design Automation Conference (DAC'19) WIP session, Las Vegas, NV, USA, June 2-6, 2019.

4. **Hongchang Shan**, Yu Liu, and Todor Stefanov,
"Ensemble of Convolutional Neural Networks for P300 Speller in Brain Computer Interface",
In *Proceedings of the 28th International Conference on Artificial Neural Networks (ICANN'19)*, pp. 376-394, Munich, Germany, September 17-19, 2019.

5. **Hongchang Shan**, Yu Liu, and Todor Stefanov,
"An Empirical Study on Sensor-aware Design of Convolutional Neural Networks for P300 Speller in Brain Computer Interface,"
In *Proceedings of "12th IEEE International Conference on Human System Interaction (IEEE HSI'19)"*, pp. 5-11, Richmond, Virginia, USA, June 25-27, 2019

Samenvatting

Een P300-gebaseerde Brain Computer Interface (BCI) karakter speller, ook bekend als P300-speller, is een belangrijk communicatiepad, dat uitgebreid onderzocht wordt, voor mensen die motorisch vermogen verliezen, zoals patiënten met Amyotrophic Lateral Sclerosis (ALS) of letsel aan de ruggengraat, omdat een P300-speller mensen in staat stelt om karakters rechtstreeks te spellen met behulp van een blik, waardoor de communicatie tussen het menselijk brein en de computer wordt opgebouwd. Helaas worden P300-spellers nog steeds niet gebruikt in het dagelijks leven van de mens en blijven ze in een experimenteel stadium in onderzoekslaboratoria. De reden voor deze situatie is dat de prestaties en de efficiëntie van huidige P300-spellers onacceptabel laag zijn voor BCI-gebruikers in hun dagelijks leven. Daarom richten we ons in dit proefschrift op het ontwikkelen van krachtige en efficiënte P300-spellers om P300-spellers praktisch in gebruik te nemen. Meer specifiek, om de prestaties van een P300-speller te verbeteren, ontwikkelen we methoden om de spelling van tekens en de Information Transfer Rate (ITR) te verbeteren. Om de efficiëntie van een P300-speller te verbeteren, ontwikkelen we methoden om het aantal sensoren dat nodig is om EEG-signalen te verkrijgen te verminderen en om de complexiteit van de classificator die in een P300-speller wordt gebruikt te verminderen zonder de prestaties te verliezen.

In het eerste deel van dit proefschrift richten we ons op het verbeteren van de prestaties en de efficiëntie (i.a.w. het verminderen van de complexiteit van de classificator) van een P300-speller. We stellen een eenvoudige, maar effectieve CNN-architectuur voor, genaamd One Convolution Layer Neural Network (OCLNN), voor de P300-speller. Onze OCLNN heeft slechts één convolutielaag en kan effectief zeer nuttige P300-gerelateerde gezamenlijke ruimtelijke-temporele kenmerken leren van ruwe EEG-signalen. In vergelijking met de andere geavanceerde methoden voor de P300-speller, verbetert onze OCLNN de prestaties, in termen van de spellingsnauwkeurigheid en de ITR, van de P300-speller, evenals de aanzienlijke verbetering van de efficiëntie, d.w.z. dat de complexiteit van het CNN dat wordt gebruikt in de P300-speller is verminderd.

De door onze OCLNN behaalde ITR kan echter nog steeds niet de theoretisch haalbare maximale ITR bereiken. Om de ITR van een P300-speller te verhogen om deze dichter bij de theoretisch haalbare maximale ITR te brengen, stellen we daarom een ensemble van CNNs voor de P300-speller voor. Ons ensemble van CNNs heet Ensemble of Convolutional Neural Networks (EoCNN). EoCNN gebruikt twee nieuwe CNNs die we hebben bedacht, One Spatial Layer Network (OSLN) en One Temporal Layer Network (OTLN) genoemd. OSLN en OTLN leren respectievelijk P300-gerelateerde afzonderlijke ruimtelijke kenmerken en P300-gerelateerde afzonderlijke temporele kenmerken. Onze EoCNN combineert OSLN en OTLN samen met OCLNN, waardoor meer nuttige P300-gerelateerde functies worden geëxtraheerd dan alleen OCLNN. In vergelijking met OCLNN verbetert onze EoCNN de prestaties, in termen van de spellingsnauwkeurigheid en de ITR, van de P300-speller, maar we beïnvloeden de efficiëntie op een negatieve wijze. Zo is de complexiteit van het CNN dat wordt gebruikt in onze op EoCNN gebaseerde P300-speller toegenomen.

Om de efficiëntie van onze op EoCNN gebaseerde P300-speller te verbeteren, richten we ons in het tweede deel van dit proefschrift op het verminderen van het aantal sensoren dat nodig is om EEG-signalen in onze op EoCNN gebaseerde P300-speller te verkrijgen. We stellen een sensor-reductiemethode voor, genaamd Spatial Learning based Elimination Selection (SLES), om het aantal sensoren te verminderen dat wordt gebruikt in de EoCNN-gebaseerde P300-speller zonder de state-of-the-art prestaties te verliezen. Onze SLES gebruikt een nieuw, door ons bedacht, geparametriseerd CNN, om de sensoren te evalueren en te rangschikken tijdens het sensorselectieproces. Deze methode is voorzien van een iteratief algoritme voor achterwaartse eliminatie om sensoren te elimineren en te selecteren. Door onze SLES-methode te gebruiken, verbeteren we de efficiëntie, d.w.z. we verminderen het aantal sensoren voor de acquisitie van EEG-signalen in de EoCNN-gebaseerde P300-speller zonder de state-of-the-art prestaties te verliezen.

Helaas moeten we, door alleen de SLES-methode te gebruiken, in de meeste gevallen nog steeds meer dan 16 sensoren gebruiken om EEG-signalen te verkrijgen in de EoCNN-gebaseerde P300-speller om de state-of-the-art prestaties te behouden. Populaire lage complexiteit en relatief goedkope (betaalbare) BCI-systemen gebruiken echter minder dan of exact 16 sensoren voor het verkrijgen van EEG-signalen. Daarom voeren we een onderzoek uit naar het bereiken van de state-of-the-art prestaties voor de EoCNN-gebaseerde P300-speller met minder dan of exact 16 sensoren om EEG-signalen te verkrijgen. Deze studie laat zien dat EoCNN het probleem heeft evenveel belang te hechten aan OSLN, OTLN en OCLNN bij het combineren van de uitkomsten van OSLN, OTLN en OCLNN ongeacht het aantal sensoren dat wordt gebruikt om EEG-signalen te verwerven. Om dit probleem op te lossen, stellen we een verbeterde EoCNN voor, genaamd PEoCNN, voor de P300-speller. In PEoCNN parametriseren we eerst het proces voor het combineren van de uitkomsten van OSLN, OTLN en OCLNN. Vervolgens gebruiken we de Sequential Model-based Algorithm Configuration (SMAC) om automatisch waarden voor de parameters te vinden en in te stellen, afhankelijk van het aantal sensoren dat in de P300-speller wordt gebruikt. Op deze manier configureert PEoCNN het belang van de uitkomsten van OSLN, OTLN en OCLNN voor de P300-speller, afhankelijk van het aantal sensoren dat wordt gebruikt om EEG-signalen te verkrijgen. Als gevolg hiervan kan de op PEoCNN gebaseerde P300-speller worden gebruikt in populaire BCI-systemen met een lage complexiteit met minder dan 16 sensoren om EEG-signalen te verkrijgen zonder de state-of-the-art prestaties te verliezen.

Acknowledgements

When I am writing this page, I can shout "Finally, I finish my PhD! Cheers!".

I would like to have a big hug with my parents! I know it is not easy for you to have you son studying abroad. You miss me so much and are often worried about whether I can take good care of myself. Sometimes I am bad-tempered, but you understand me! When I face difficulties, your words are so powerful for me to face my PhD journey. Your supporting makes my heart stronger and stronger. You are always my belief to be a better man! I am proud that you are proud of me!

I am so lucky to have you, my wife **Yuye Que**! I am very grateful that you always accompany me to face and fight against the difficulties. Actually, you are a positive and optimistic person. It is your magic that you are able to get me full of hopes when I feel hard. Your excellent Chinese cuisine always makes me feel warm in my heart. The PhD journey in Leiden is bitter but the life with you in Leiden is sweet! My mom said you are the lucky star to me! Undoubtedly and Absolutely!

I would like to say a big "Thank you" to **Chuan Luo**! Every time I come to you for help, you are always patient and kind no matter how busy you are. I still remember that spring when we walk along the small river behind your house and talk about the recent unhappiness for many times. Your words always calm me down when I feel anxious. I believe our friendship will never fade away.

Many thanks to **Yu Liu**! Thanks so much for your help to let me understand further about the deep learning word. I learn a lot from each discussion with you. You always share time with me to answer my questions very patiently. Your knowledge and critical thinking give me many inspirations in my research field. Having a big brother like you is such a happy thing!

It is a pleasure to work in Leiden Embedded Research Center (LERC)! LERC is a big family. Thanks for meeting the friendly and lovely colleagues in LERC. **Di Liu** often encourages me and gives me suggestions. **Jelena Spasic**'s hard working attitude makes a good example for me. Thanks to **Sobhan Niknam** for organizing activities in LERC. **Erqian Tang**'s optimism is really appreciated by me. Also, thanks for the discussions and help from other colleagues! Working in LERC is an unforgettable memory in my life!

Last but not of less importance, thanks for the friends in Leiden Institute of Advanced Computer Science (LIACS) and in Leiden outside the academic world. There are many memorable gatherings where I can eat the delicious Chinese food cooked by you guys and have talks about our life in the Netherlands. We have basketball matches every Friday night where I can forget research and relax myself. Thank you for organizing activities to experience the Netherlands where I can enjoy the beautiful scenery. Hope you everything goes well in the future!

Again. I owe my sincere gratitude to all of you beyond the words.

Curriculum Vitae

Hongchang Shan was born on October 11, 1989 in Heilongjiang, China. He obtained his B.Eng degree in Automation Science and Technology from Xi'an Jiaotong University, China in 2012. He joined the Leiden Embedded Research Center (LERC), part of the Leiden Institute of Advanced Computer Science (LIACS) at Leiden University, as a Ph.D. candidate in November, 2015. In LERC, he has been working, towards his Ph.D degree, as a research assistant. Besides his work as a researcher, he was involved as a teaching assistant in the Digital Technique, Computer Architecture, and Embedded Systems and Software courses.