Master's thesis

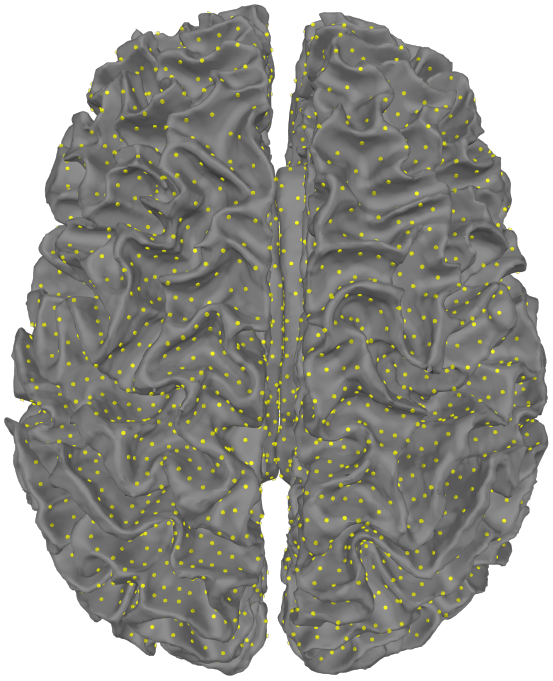
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Simen Piene Fløtaker

Classification of RGB Color Stimuli in Source Space

Exploring methods for decoding EEG responses to red, green, and blue color stimuli by the use of source reconstruction, deep neural networks, and channel reduction with genetic algorithm

Master's thesis in Cybernetics and Robotics
Supervisor: Marta Molinas
Co-supervisor: Andres Soler
June 2023



NTNU
Norwegian University of
Science and Technology

Simen Piene Fløtaker

Classification of RGB Color Stimuli in Source Space

Exploring methods for decoding EEG responses to red, green, and blue color stimuli by the use of source reconstruction, deep neural networks, and channel reduction with genetic algorithm

Master's thesis in Cybernetics and Robotics
Supervisor: Marta Molinas
Co-supervisor: Andres Soler
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis concludes a Master of Technology at the Department of Engineering Cybernetics of the Norwegian University of Science and Technology (NTNU). The work on this thesis was conducted throughout the spring of 2023. The work is a continuation of a Semester project conducted by the author in the autumn of 2022: *Deep learning for classification of primary color responses in EEG signals using source reconstruction* [27]. The project and master thesis was proposed by Professor Marta Molinas and Andres Soler. Andres Soler also provided the dataset used in this thesis, which was recorded by him as part of a previous study [52].

Section 3.2 is a revised version of a similar section in the Semester project report [27]. The method described in section 3.3 is directly inspired by the work of Luis Alfredo Moctezuma [54], both in approach and implementation. Chapter 4 presents the results of the work conducted in this thesis. These results are the original work of the author, unless explicitly stated otherwise.

Trondheim, June 1, 2023

Simen Fløtaker

Acknowledgement

I am deeply grateful to Marta Molinas, for her support and motivation throughout my work on this thesis. Above all, her sincere interest in the research has sparked interesting discussions, that have motivated and inspired me. Knowing that there is genuine interest in my work has made it all the more exciting. I would also like to thank her for encouraging me to write and submit a paper on the preliminary study for this work, which has been an interesting and educational experience.

I also extend my utmost appreciation to Andres Soler. He has been an immense resource of technical knowledge and insight. His help has been available at all times, for which I am very thankful. Andres has always helped me to explore my ideas, whilst making sure to guide them in the right direction.

Abstract

This thesis explores the feasibility of using electroencephalogram (EEG) signals to determine whether a person has been exposed to red, green, or blue (RGB) visual stimuli. If a classifier can be developed to distinguish such stimuli, it could serve as the basis for a brain-computer interface (BCI), a system for controlling computers by explicit manipulation of brain activity. In this work, deep neural networks (DNNs), specialized for EEG signals, have been employed to serve as classifiers. In addition to the classifiers, efforts were focused on methods for manipulating the data to make it more suitable for classification. Source reconstruction (SR), a method for estimating brain activity at a set of positions in the brain based on EEG recordings, was used to produce a data representation with higher spatial resolution than the raw electrode data. Both the raw electrode data and the source reconstructed data contain channels, representing signals from scalp locations (for electrodes) or within the brain (for source reconstruction). A channel selection method has been developed, using a genetic algorithm (GA), to search for optimal subsets of channels to use when classifying brain activity elicited by RGB stimuli. A dataset comprised of EEG recordings from 31 subjects was used to evaluate the performance of the different methods and classifiers.

All the classifiers developed in this work have been intra-subject classifiers, meaning they are trained on data from only one subject, and tested on data from the same subject. Four tests, with a total of fourteen subtests, were performed to evaluate different combinations of the developed methods. The results of these tests did not show an improvement in classification accuracy when using reconstructed sources as opposed to raw electrode data. The best results obtained using electrode data had an accuracy of 80%, averaged across subjects. With reconstructed sources, the best average accuracy was 73%. The results proved that channel selection by use of GA is a suitable method for reducing the number of channels without loss of classification accuracy. For reconstructed sources, the channel selection reduced the number of channels by 59%, with the average accuracy remaining the same. When using channel selection on electrodes, the number of channels was reduced by 83%, with an increase in average accuracy of 3%. The dataset used in this thesis has been subject to study in previous research. The best result from that research was an average accuracy of 74%, for a subset of subjects. With the best classifier from this work, the same subset had an average accuracy of 88%.

Sammendrag

Denne oppgaven utforsker muligheten for å bruke elektroencefalografi (EEG) signaler for å avgjøre hvorvidt en person undergår rød, grønn, eller blå (RGB) visuelle stimuli. Dersom en algoritme kan utvikles, som kan klassifisere slike stimuli, kan den utgjøre grunnlaget for et hjerne-datamaskin grensesnitt (BCI), et system for å kontrollere datamaskiner ved eksplisitt manipulasjon av hjernesignaler. Dype nevrale nettverk (DNN), spesialisert for EEG signaler, har blitt anvendt for å klassifisere data i dette arbeidet. Utover dette har arbeidet vært rettet mot metoder for å manipulere data, slik at den er mer tilrettelagt for klassifisering. En rekonstruksjonsmetode for å estimere kildene til signalene målt med EEG ble benyttet for å lage en representasjon av EEG data med høyere romlig oppløsning enn de direkte elektrodemålingene. Både elektrodemålingene og de rekonstruerte kildene har flere kanaler med data, som representerer signaler fra posisjoner på hodebunnen (for elektroder) og i hjernen (for rekonstruerte kilder). En metode har blitt utviklet, ved bruk av en genetisk algoritme (GA), for å lete etter optimale undergrupper av kanaler til bruk i klassifisering av hjerneaktivitet forårsaket av RGB stimuli. Et datasett bestående av EEG målinger fra 31 testpersoner ble brukt til å evaluere ytelsen til de forskjellige metodene og klassifiseringsalgoritmene.

Alle klassifiseringsalgoritmene utviklet i dette arbeidet har blitt trent på data fra kun én person, og testet på data fra den samme personen. Fire tester, med til sammen fjorten undertester ble utført, for å evaluere forskjellige kombinasjoner av de utviklede metodene. Resultatene fra disse testene viste ingen forbedring i klassifisering ved bruk av rekonstruerte kilder sammenliknet med direkte elektrodedata. De beste resultatene ved bruk av elektrodedata hadde en gjennomsnittlig nøyaktighet på 80%. Med rekonstruerte kilder var den beste gjennomsnittlige nøyaktigheten 73%. Resultatene viste at metoden for å lete etter undergrupper av kanaler med en GA var en egnet metode for å redusere antallet kanaler, uten tap av nøyaktighet. For rekonstruerte kilder reduserte denne metoden antallet kanaler med 59%, uten å påvirke nøyaktigheten av klassifiseringen. Med elektrodedata reduserte metoden antallet kanaler med 83%, og økte nøyaktigheten med 3%. Datasettet som ble brukt i denne oppgaven har også blitt brukt i tidligere en tidligere studie. De beste resultatet fra det studiet var en gjennomsnittlig klassifiseringsnøyaktighet på 74%, for et utvalg av testpersonene. Med den beste klassifiseringsalgoritmen i dette arbeidet, har det samme utvalget en nøyaktighet på 88%.

Contents

Preface . . . i
Acknowledgement . . . ii
Abstract . . . iii
Sammendrag . . . iv
List of Figures . . . ix
List of Tables . . . x

1 Introduction **1**
1.1 Background . . . 1
1.2 Related Work . . . 2
1.2.1 Classification of Color Stimuli . . . 2
1.2.2 Classification of Reconstructed Sources . . . 3
1.3 Objectives . . . 4
1.4 Approach . . . 4
1.5 Limitations . . . 4
1.6 Outline . . . 5
1.7 Contributions . . . 5

2 Theory **6**
2.1 Brain Signals . . . 6
2.1.1 Generation of Signals in the Brain . . . 6
2.1.2 Color Perception in the Brain . . . 7
2.1.3 EEG Recording . . . 9
2.1.4 EEG Noise and Artifacts . . . 10
2.2 Genetic Algorithms . . . 10
2.2.1 Basic Structure of a Genetic Algorithm . . . 11
2.2.2 NSGA-III . . . 12
2.3 Deep Learning . . . 13
2.3.1 Training and Testing of ML Classifiers . . . 13
2.3.2 Convolutional Neural Networks . . . 14
2.3.3 Regularization . . . 16
2.3.4 Layers of the Keras Framework . . . 17
2.3.5 Hyperparameter Tuning . . . 18

3 Materials and Methods **19**

3.1 Dataset . . . 19
3.1.1 Recording and Collection . . . 19
3.1.2 Preprocessing . . . 21
3.1.3 Structure of the Dataset . . . 22
3.2 Source Reconstruction . . . 25
3.2.1 Forward Model . . . 25
3.2.2 Inverse Problem . . . 28
3.2.3 Template Forward Models . . . 29
3.3 Channel Selection with Genetic Algorithm . . . 30
3.3.1 Implementation . . . 30
3.3.2 Electrode Selection . . . 32
3.3.3 Source Selection . . . 33
3.4 Classification . . . 40
3.4.1 Structured Spatial Source Representation . . . 40
3.4.2 3M3DCNN . . . 43
3.4.3 2DCNN . . . 44
3.4.4 ST3DCNN . . . 45
3.4.5 Time of Interest Selection . . . 46
3.4.6 Data Augmentation . . . 48
3.4.7 Training and Testing Routine . . . 49
3.5 Software and Hardware . . . 51
3.5.1 Software . . . 51
3.5.2 Hardware . . . 51

4 Results **53**
4.1 Tests Performed . . . 53
4.1.1 Test 1: Full vs Minimal Preprocessing . . . 53
4.1.2 Test 2: All vs Selected Channels . . . 53
4.1.3 Test 3: Data Augmentation . . . 53
4.1.4 Test 4: 3M3DCNN, 2DCNN, and ST3DCNN . . . 54
4.2 Test Results . . . 54
4.2.1 Test 1: Full vs Minimal Preprocessing . . . 54
4.2.2 Test 2: All vs Selected Channels . . . 55
4.2.3 Test 3: Data Augmentation . . . 56
4.2.4 Test 4: 3M3DCNN, 2DCNN, and ST3DCNN . . . 57
4.2.5 Comparison with Previous Studies . . . 58
4.2.6 Confusion Matrices . . . 59

5 Discussion, Conclusion, and Further Work **62**
5.1 Summary of Findings . . . 62
5.2 Discussion of Findings . . . 63
5.2.1 Choice of Preprocessing . . . 63
5.2.2 Source Space for Improved Classification . . . 63
5.2.3 A Review of Channel Selection . . . 64

5.2.4 Structured Spatial Source Representation 65
5.2.5 Data Augmentation 67
5.2.6 A Review of BCI Applicability 67
5.3 Suggestions for Further Work 68
5.3.1 Inter-Subject Classifiers 68
5.3.2 Investigate Different Source Reconstruction Methods 68
5.3.3 4D Spatiotemporal Convolutions 69
5.3.4 More Extensive Source Selection 69
5.4 Conclusion 70

References **71**

A List of Abbreviations **77**

B Abstract for BCI Society Meeting **79**

C Paper for EMBC Conference **81**

List of Figures

2.1 An action potential. . . . 7
2.2 The wavelength sensitivity of the three types of cone photoreceptors. . . . 8
2.3 The pathway of the neural signals responsible for visual perceptions. . . . 8
2.4 Topographical map of the areas of the brain. . . . 9
2.5 The 10-10 electrode placement system. . . . 10
2.6 The basic structure of genetic algorithms. . . . 11
2.7 Pareto fronts. . . . 12
2.8 Non-dominated sorting in the NSGA-III algorithm. . . . 12
2.9 An illustration of over- and underfitting in machine learning. . . . 14
2.10 Basic structure of a neural network. . . . 15
2.11 An example of a convolution in neural networks. . . . 16
2.12 The typical segmentation of the dataset in a machine learning project. . . . 18

3.1 The layout of the 60-electrode EEG-cap used to record the dataset. . . . 20
3.2 The stimuli protocol. . . . 20
3.3 The epoching process. . . . 21
3.4 The structure of the dataset. . . . 23
3.5 The relationship between source space and electrode space. . . . 25
3.6 The structure of the forward model. . . . 26
3.7 Example of three-layer head segmentation. . . . 26
3.8 Example of the white and gray matter segmentation. . . . 27
3.9 The distribution of sources on the white matter surface. . . . 28
3.10 The channel selection chromosome representation. . . . 31
3.11 The channel selection fitness function. . . . 32
3.12 The solutions of channel selection in electrode space. . . . 33
3.13 Two electrode configurations found by channel selection. . . . 33
3.14 The source selection based on selected electrodes. . . . 35
3.15 The solutions of channel selection in source space. . . . 36
3.16 The distribution of selected sources, in a selected area of the brain. . . . 37
3.17 The distribution of selected sources, in the whole brain, in each region of interest. . . . 39
3.18 An illustration of structured and unstructured dimensions. . . . 41
3.19 Voxelization of point cloud for source space data. . . . 41
3.20 Voxel and pixel representations of source space data. . . . 42
3.21 Feature extraction for 2DCNN. . . . 46

3.22 Time of interest selection results. 47
3.23 An example of data augmentation. 48
3.24 The cross validation method. 50

4.1 Confusion matrices from Test 1. 60
4.2 Confusion matrices from Test 2. 60
4.3 Confusion matrices from Test 3. 61
4.4 Confusion matrices from Test 4. 61

5.1 The various data pipelines that have been explored. 62
5.2 The effects of pixelization by projection from behind. 66

List of Tables

3.1 The dataset used for classification. 24
3.2 Results from channel selection with genetic algorithm in electrode space. 32
3.3 Results from channel selection with genetic algorithm in source space. 36
3.4 Results from subject-specific source selection. 37
3.5 The distribution of sources in ROIs, for the 192-source selection. 38
3.6 The 3M3DCNN architecture. 43
3.7 The 2DCNN architecture. 45
3.8 The ST3DCNN architecture. 47

4.1 Overview of the different test configurations. 54
4.2 The results from test 1, comparing full and minimal preprocessing. 55
4.3 The results from test 2, comparing channel selection vs. all channels. 56
4.4 The results from test 3, comparing data augmentation methods. 57
4.5 The results from test 4, comparing 3M3DCNN, 2DCNN and ST3DCNN. 58
4.6 Comparison with results from previous studies. 59

5.1 Difference in data examples and accuracies between minimal and full preprocessing. . 64

Chapter 1

Introduction

1.1 Background

Brain-computer interfaces (BCIs) are systems built to allow control of computer systems, through explicit manipulation of brain activity [66]. Since BCIs do not rely on the physical use of muscles, as keyboards and touch screens do, they can serve as alternative interfaces for users with physical disabilities. People suffering from locked-in syndrome, who cannot use either traditional interfaces or speech-controlled devices, may still use BCI systems. People that previously could not manipulate their environment, might be able to interact with their surroundings through such systems, giving them more freedom, control, and quality of life. Further into the future, with sufficient advances in the technology, BCIs might also serve as a more efficient and seamless interface for any user.

The control tasks, the set of mental efforts performed by the user as input to the BCI [66], is an important design choice. These mental efforts can be directly produced by the user (endogenous) or they can be induced automatically by some external stimuli (exogenous). Control tasks such as motor imagery (MI) (where the user thinks of performing certain muscle movements) and imagined speech (where the user thinks of saying certain words) are examples of previously studied endogenous tasks [24, 15]. For exogenous tasks, sounds and visual cues can be used as part of the control task paradigm. The user can be presented with several stimuli, and manipulate their brain activity by focusing on one of them. Several considerations have to be made when designing the control tasks; There should be an intuitive connection between the mental effort and the desired output, the brain activity from different tasks should be different enough to distinguish reliably, the tasks should be easy to perform for anyone, etc. Exogenous paradigms typically have the advantage of being more robust and require less user training than systems based on endogenous tasks [67]. On the other hand, the stimuli in exogenous paradigms often cause fatigue and discomfort over time [48, 67].

Different colors could be used as external visual stimuli in a control task paradigm. By presenting the BCI user with different colors, they could focus on a specific color to perform a specific task such as opening and closing doors. Compared to common exogenous tasks, based on flashing lights or letters [66], focusing on a static color may cause less fatigue, thus overcoming an often-mentioned disadvantage of exogenous paradigms. Another advantage is that colors do not require electricity or a display of some sort, making them more practical for integration in an environment than flashing

lights and letters. Previous research has shown through behavioral and physiological studies, the ability of prelingual infants to distinguish and categorize colors [45, 7]. Such studies indicate an innate ability of the human brain to separate colors. It is thus natural to assume that the brain activity induced by different colors is separable. With a background in all these aspects, color stimuli could be a promising part of a BCI paradigm.

To the knowledge of the author, the brain's response to color stimuli has not yet been used as the control task in any BCI, however, some research has been conducted on offline classification of brain activity elicited by color stimuli [52, 4, 61, 60]. The methods of recording brain activity used in these studies were electroencephalography (EEG) and magnetoencephalography (MEG). Of these methods, only EEG has properties suitable for integration in a BCI [66]. Although [52] and [4] demonstrate that color stimuli can be decoded above chance level with EEG data, the accuracies were too low compared to what should be expected of a robust BCI. In recent years, source reconstruction (SR) has been demonstrated to improve the decoding accuracy of certain brain activities [50, 24, 9]. SR is a method for estimating the brain activity at a set of positions in the brain, from EEG or MEG measurements. It is thought that the improvement in classification achieved with SR can be attributed to source-reconstructed data being more highly correlated to the control tasks, due to its high spatial resolution compared with raw EEG data [22]. Preliminary studies, conducted by the author, explored the use of SR for decoding color stimuli, however, these studies did not demonstrate an increase in performance when using source-reconstructed data instead of raw electrode data [27]. Hence, the questions remain open as to whether color stimuli can be decoded with high enough accuracy for use in a BCI, and if SR can be used to improve performance in color stimuli classification.

This thesis explores methods for classifying stimuli from the primary colors red, green, and blue (RGB) based on EEG recordings, and assesses the possibility of using such stimuli as part of a BCI paradigm. The main focus has been the use of three distinct methods. Firstly, SR was used to transform the data to a format that might improve classification. Secondly, a genetic algorithm (GA) was used to search for a subset of data channels to use in the classification, to reduce the amount of data, and improve the signal-to-noise ratio (SNR). Lastly, deep neural networks (DNNs) have been used to perform the classification itself. These three methods are the focus of sections 3.2, 3.3, and 3.4 respectively.

1.2 Related Work

The design and implementation of the methods used in this work have been guided by previous research in the field. Specifically, previous studies of decoding EEG signals using SR and decoding of color stimuli have been important, as they give motivation for exploring SR and give an indication of the state of the art in color decoding. This section provides a short description of some studies which has been important for this work, for motivating certain methods, and for serving as benchmarks by which the results could be evaluated.

1.2.1 Classification of Color Stimuli

[60] Rasheed et al., "EEG Spectral Analysis of Visual Evoked Potential Produced by RGB Color Stimuli"

An early study into color-induced brain activity, by means of EEG, was conducted by Rasheed et al..

The aim of the study was to verify whether the response to the primary colors red, green, and blue could be detected in EEG data. Time-frequency spectra of responses to the different colors revealed some distinct differences between the colors. This result indicated that future BCI applications could be feasible, and the paper encouraged further work on the subject.

[4] Åsly, "Supervised learning for classification of EEG signals evoked by visual exposure to RGB colors"
The focus of this work was classifying responses to primary color stimuli in EEG signals. Results were presented for traditional machine learning (ML) classifiers and DNN classifiers. The classifiers based on traditional ML were trained and tested on both a subject-specific basis and across all subjects. The DNN classifiers were tested only across all subjects. When trained across all subjects, the DNN was reported to achieve 46% accuracy while the traditional ML classifier achieved 37%. For the subject-specific ML classifiers, an average accuracy of 45% was attained.

[52] Ludvigsen et al., "The Augmented Human: Development of BCI for RGB colour-based automation"
In this study, traditional ML classifiers were developed to classify primary color responses in EEG recordings. The classifiers were trained and tested using the same dataset as in this study. Hence, their results served as good benchmarks for comparison with the results of this study. Their best results were obtained with a minimum distance to mean with geodesic filtering (FgMDM) Riemannian classifier. The result was an average accuracy of 74.48% per subject, with intra-subject classifiers. Classification of source-reconstructed data was also performed, with the best classifier having an average accuracy of 49.73%.

[61] Rosenthal et al., "Color space geometry uncovered with magnetoencephalography"
This study explored the brain's response to visual stimuli of different colors and color words. The data used in this research was recorded using MEG. Several properties of the responses were investigated, among them the delay between stimulus onset and maximum decoding accuracy and the separability of different colors. It was reported that the maximum decoding accuracy was at 115ms after stimuli onset. However, this varied substantially from subject to subject. The research also found that a higher accuracy was achieved for decoding warm colors (such as pink) than cool colors (such as green).

1.2.2 Classification of Reconstructed Sources

[9] Cincotti et al., "High-resolution EEG techniques for brain–computer interface applications"
This study aimed to investigate if using reconstructed source data could yield better classification than when using raw EEG data. Brain activity was recorded with EEG while subjects were prompted to perform four types of imagined limb movements. Both statistical analysis and online application in a BCI paradigm demonstrated that using reconstructed sources could improve the performance of a classifier.

[50] Li and Ruan, "A novel decoding method for motor imagery tasks with 4D data representation and 3D convolutional neural networks"
This paper introduced a convolutional neural network (CNN) for decoding source-reconstructed data. The data to decode was of different imagined movements. The developed CNN used three-dimensional

(3D) spatial convolutions to take advantage of the spatial properties of the reconstructed sources. Results from testing the developed classifier on public datasets showed that it outperformed previously proposed classifiers operating on raw EEG data.

[24] Fang et al., "Decoding motor imagery tasks using ESI and hybrid feature CNN"
This paper also introduced a custom CNN for classifying reconstructed sources. However, the data input to this classifier had a two-dimensional (2D) spatial representation of the brain activity, as opposed to the 3D representation used in [50]. In addition, frequency features were extracted before the data was fed into the classifier. Experiments showed that the proposed classification procedure outperformed almost all other state-of-the-art methods.

1.3 Objectives

The overall aim of this thesis has been to explore methods for classifying color responses in EEG signals. This exploration has been conducted with a focus on three distinct methods, which serve as the objectives for the work:

O1: Explore the use of source reconstruction (SR) on EEG signals to create data with high spatial resolution.

O2: Implement a genetic algorithm (GA) paradigm for selecting subsets of channels in EEG data and source-reconstructed data, that are beneficial for classification accuracy.

O3: Explore and test deep neural network (DNN) architectures for classifying primary color responses in source-reconstructed data.

1.4 Approach

This work builds upon the preliminary studies conducted in [27]. A dataset containing EEG recordings of a set of subjects receiving color stimuli has been the foundation of the research. This dataset was divided into two parts: one used during the development and exploration of the different methods, and one used to test the fully developed methods. SR was then employed to create an alternative representation of the dataset. Both the source-reconstructed and raw datasets were then used by a GA to search for optimal subsets of sources and electrodes. The GA used the best classifier from the preliminary studies conducted, to evaluate the performance of different possible subsets. Three DNN architectures designed specifically for source-reconstructed data were implemented in an effort to take advantage of the high spatial resolution of such data. Finally, a set of tests were defined to systematically evaluate the performance of different combinations of the developed methods.

1.5 Limitations

All classifiers developed in this project are intra-subject, they are trained and tested on data from only one subject. This means that the results presented in this work reflect how well the proposed classifiers can classify on subjects it is trained on. How well the classifiers would perform if presented with data from other subjects than it is trained on has not been explored.

The EEG data used in this work are recorded in a controlled environment with optimal conditions. During recording, subjects were sitting down in a magnetically shielded room, and instructed to refrain from moving and blinking, thus ensuring a minimal amount of artifacts and noise in the recorded data. The color stimuli received was also occupying a substantial portion of their view, and only one color was visible at a time. These conditions can not be expected in a BCI used in everyday life, thus the results obtained in this research cannot be considered realistic estimates for the obtainable accuracy in a BCI using the same methods.

1.6 Outline

This introduction is the first of five chapters in this thesis. Chapter 2 covers some theory on brain signals, genetic algorithms, and machine learning, to provide sufficient background for the subsequent chapters. Chapter 3 describes the materials and methods used in this work. The chapter covers four main parts of this work: the dataset, channel selection with genetic algorithms, source reconstruction, and classification using deep learning. Chapter 4 presents the results obtained from a set of tests performed to evaluate the various methods introduced in Chapter 3. Finally, Chapter 5 provides a discussion of the results and experiences gained throughout the work. This last chapter also suggests topics for further work and closes with a conclusion summarizing the work. In addition to these chapters, there are three appendices in this thesis. The first, a list explaining all abbreviations used in this thesis can be found in Appendix A. The second, Appendix B, is an abstract of the preliminary work, accepted for the 2023 BCI Society Meeting. The final appendix, Appendix C, is a paper about the preliminary work, accepted for the 2023 IEEE EMBC conference.

1.7 Contributions

This work contributes with both insight and methods for the decoding of EEG signals, with an emphasis on the decoding of RGB color stimuli. The major contributions are:

- An approach for reducing the number of channels in a source space representation of EEG data without loss of decoding accuracy, using a GA.
- An evaluation of different DNN architectures, for decoding RGB color stimuli from source-reconstructed data.
- A demonstration of deep learning's ability to classify with high accuracy on EEG data with minimal preprocessing, and no artifact suppression.

The preliminary work also contributed to the evaluation of different DNN architectures. An abstract of that work has been accepted for the 2023 BCI Society Meeting. This is a meeting for persons involved in BCI research and clinical use. More information about this event can be found at bcisociety.org/bci-meeting/. The preliminary work is also presented in a paper, which has been accepted for the 2023 IEEE EMBC conference. This is a conference bringing together companies, institutes, universities, and researchers in the field of biomedical science. More information on this conference can be found at embc.embs.org/2023/

Chapter 2

Theory

2.1 Brain Signals

The brain is the most complex organ in the human body, and is a part of the central nervous system (CNS) [5]. It is responsible for interpreting many different sensory signals, and for producing the signals that control muscle movement. In this section, some theory will be presented on how these functions are achieved. The section will also cover the basics of recording brain signals using EEG. The intention is to provide sufficient background theory for interpreting and discussing the data and methods used in this work.

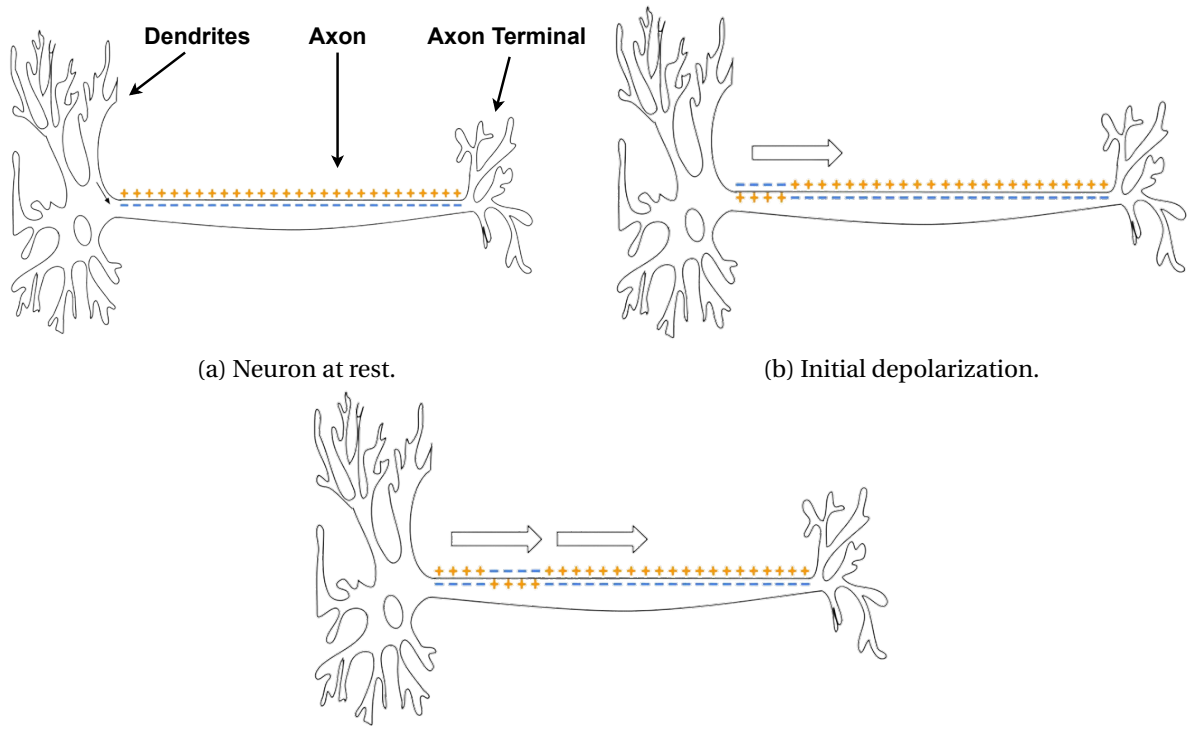
2.1.1 Generation of Signals in the Brain

In order to understand the signals recorded by an EEG, we must first have an idea of the function of a single neuron. The adult human brain contains roughly 85 billion neurons, these are the cells that communicate incoming sensory signals, and produce appropriate muscle signals for the body to respond to its environment [5]. The function of a neuron can be attributed to three of its main parts:

- **Dendrites:** Tree-like structures on one end of the neuron, receiving inputs from other neurons (or cells). Depicted on the left of the neuron in Figure 2.1.
- **Axon:** The long wire-like part of the neuron that transfers signals over a distance. The middle part of the neuron in Figure 2.1.
- **Axon terminal:** The plates at the end of the axon. This is where the signal of one neuron is transferred to the dendrites of another neuron (or to another type of cell). Seen on the right side of the neuron in Figure 2.1

Signals travel through a neuron by a phenomenon called the action potential (AP) [34]. In its resting state, the inside of the neuronal membrane has a negative electric charge in relation to the outside; it is polarized. The AP is triggered by a depolarization of the membrane at the input side of the neuron. When the membrane is depolarized above a certain threshold, ion channels in the membrane are opened, causing a rapid flux of positively charged ions in through the membrane. This way, the charge on the inside of the membrane becomes positive in relation to the outside. This change in charge

causes ion channels to open in the neighboring part of the axon. This continues in a chain reaction, creating a signal traveling along the axon, as illustrated in Figure 2.1.



(a) Neuron at rest.

(b) Initial depolarization.

(c) Propagation of the AP.

Figure 2.1: An action potential. The action potential travels along the axon, by depolarization of the neuronal membrane. The figure is adapted from [12].

2.1.2 Color Perception in the Brain

Different areas of the brain are responsible for different functions in the human body. Efforts in neuroscience have led to a topographic map associating different parts of the brain with different cognitive functions [66]. One such function is sight, and having some insight into this function is helpful when developing a system for decoding colors, and for interpreting EEG data from visual stimuli. This section provides some theory on the overall structure of color perception in humans, based on theory from [5].

Visual perception starts when light (electromagnetic waves in the visual spectrum) from the environment enters the eye. Inside the eye, there is a set of photoreceptors, and when light hits such a receptor, it depolarizes nearby cells, causing an AP. This AP travels along the optic nerve and enters the brain for further processing. The path of the optic nerve is illustrated in Figure 2.3. There are two types of photoreceptors: rods and cones. The cones are responsible for our ability to distinguish color. There are three types of cones, each with maximum sensitivity to a certain wavelength of light, as shown in Figure 2.2. Often these three types of cones are referred to as red, green, and blue cones, as their sensitivity peaks roughly correspond to these colors. This means that the property we perceive as color is in reality differences in wavelength. Light with different wavelengths will cause different amounts of excitation in the different cones, and the brain can interpret colors by comparing these different signals.

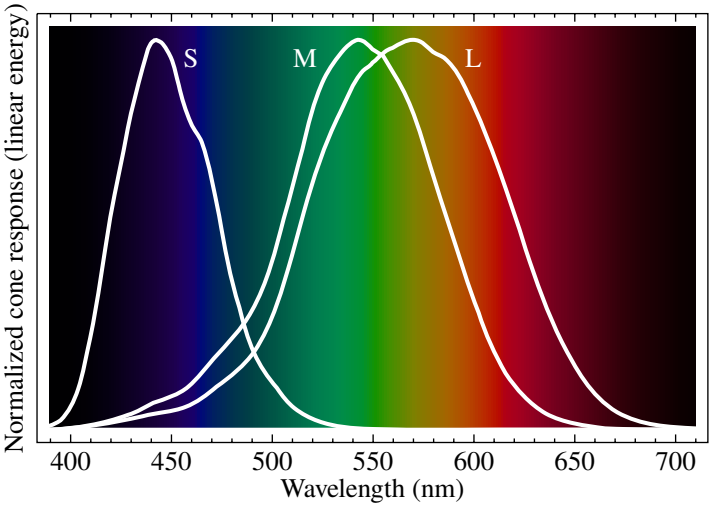


Figure 2.2: The wavelength sensitivity of the three types of cone photoreceptors. Image source: [11].

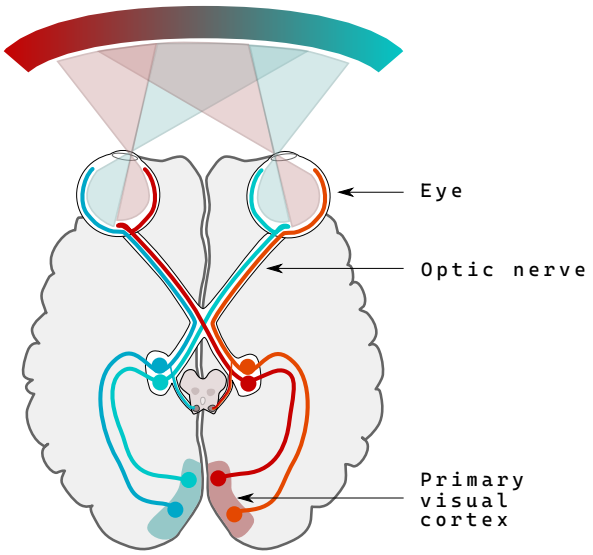


Figure 2.3: The pathway of the neural signals responsible for visual perceptions, starting in the eye and ending in the primary visual cortex. Adapted from [13].

As mentioned, topographical maps relating areas of the brain to their function have been developed, and Figure 2.4 illustrates such a mapping introduced by [51]. Visual area one (V1) in this figure indicates the primary visual cortex, which is the area of the brain that receives visual stimuli from the optic nerve. V1 and the other visual areas (V2, V3, etc.) are located in the section of the brain suitably named the occipital lobe. This is the area colored pink in Figure 2.5. Figure 2.3 shows the pathway of the signal from the eye to the visual cortex. The brain starts processing information such as motion, shape, and color in the visual cortex. However, the signals also propagate from V1 to other areas for specific processing. Especially relevant for this work is the area indicated V8 in Figure 2.4, which has been proven to be important for the perception of color. Evidently, the brain's processing of visual stimuli is distributed, and no exact explanation of every aspect of visual perception has been formulated. Naturally, visual stimuli will also cause less automatic responses such as feelings and memories, activating other areas of the brain than just the visual areas in the occipital lobe.

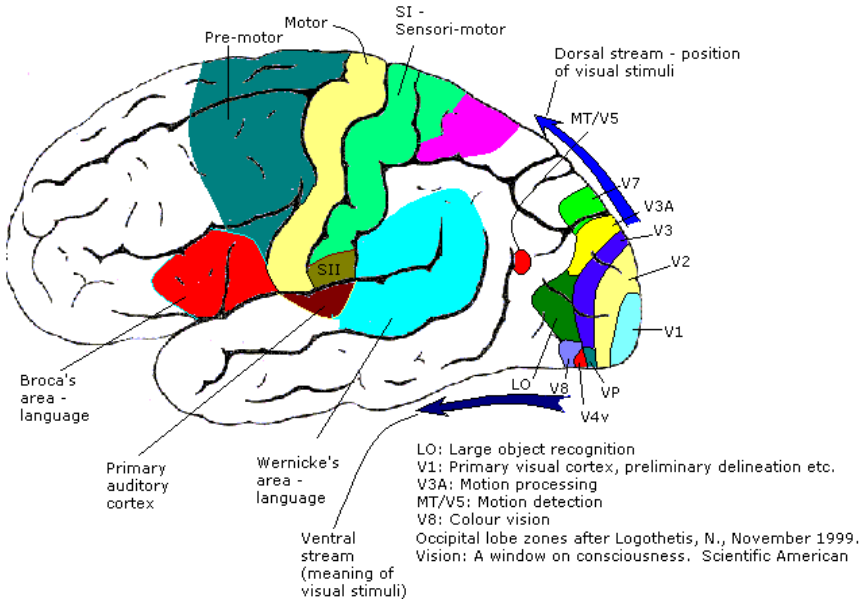


Figure 2.4: Topographical map of the areas of the brain, and their respective functions. The area V8 is involved in color perception. The figure is adapted from [10].

2.1.3 EEG Recording

EEG is a method for recording the electrical activity in the brain. As pointed out in the previous section, the electrical activity in the brain is in large part caused by neuronal activity. Thus, EEG recordings can give information about the different actions performed by the brain. The recording is performed by placing electrodes on the scalp. These electrodes measure the electric potential on the scalp, with reference to some other potential. The reference is usually produced by placing a reference electrode on some other part of the body, such as the ear lobe. Since each neuron behaves as a tiny voltage source when producing an AP, they will cause small electric currents in the brain. In the context of EEG, the head can be considered a volume conductor. The different tissues have different conductance and hence attenuate the signal differently [65]. Therefore, to get the strongest signal possible from neurons in a specific region of the brain, the electrode should be placed at the part of the scalp closest to this region. Typically, EEG-caps will be used when recording brain activity. These caps have many electrodes, spaced such that they cover a large area of the scalp, thus better capturing the electrical activity of the whole brain. Data collected with such caps can also be used to compute an estimate for the position of the activity in the brain that caused the signal measured. This process is called SR and will be further described in section 3.2.

EEG-caps typically follow a standardized electrode layout such as the 10-20 system [41]. This is a system where each electrode is placed based on landmarks on the skull. Each electrode is labeled based on what area of the brain it is in the vicinity of. These regions are the Frontal pole (Fp), Frontal (F), Central (C), Parietal (P), Temporal (T), and Occipital (O). The labels of the electrodes also contain numbers representing the position relative to the mid-line of the scull (going from nasion to inion). Electrodes in the right hemisphere are labeled with even numbers, and odd numbers are used for electrodes in the left hemisphere. More recent systems use more electrodes and have thus expanded on the original 10-20 system. [55] proposed a system adding intermediate regions. Electrodes in

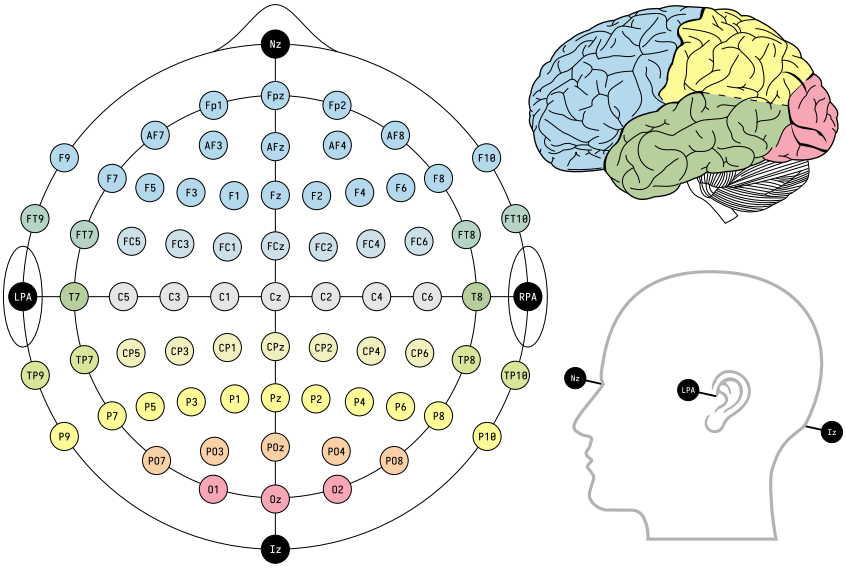


Figure 2.5: The 10-10 electrode placement system. The top right figure illustrates the regions of the brain responsible for the naming of the electrodes. The bottom right figure indicates the landmarks on the skull used to define the positions of the electrodes. Image source: [14].

these intermediate regions would be named based on the regions to whom they were intermediate. For instance, electrodes in the region between the occipital and the parietal region would have the label PO. An illustration of this modified 10-20 system, commonly referred to as the 10-10 system [56], is provided in Figure 2.5.

2.1.4 EEG Noise and Artifacts

Artifacts of biological origin and noise from the environment often contaminate EEG recordings [26]. Among sources of environmental noise, interference from grid power lines may be the most common [49]. However, the electric fields of other electric devices in the vicinity of the EEG device also influence the noise in the recording [39]. This external interference can be reduced by using shielded cables or by performing the recording in a magnetically shielded room. Biological artifacts include blinking and movement of eyes, use of muscles in the head and neck, and cardiac pulse [26]. Unlike external sources of noise, this internal interference cannot be shielded against. A common way to minimize the effect of these is to instruct subjects to refrain from blinking or moving. Obviously, shielded rooms and avoiding blinking are impractical remedies for EEG in everyday use, such as in BCIs. Several algorithms and methods have been proposed for removing both external and internal interference automatically through software [26, 29, 68]. Such methods may help make EEG more applicable for uses outside controlled environments and laboratories.

2.2 Genetic Algorithms

GAs are a class of metaheuristic algorithms inspired by the Darwinian theory of natural selection [42]. Metaheuristic algorithms are stochastic, and are not guaranteed to return an optimal solution, but are meant to serve as an efficient and practical approach to complex optimization problems [70]. GAs

mimic biological evolution, where individuals with advantageous properties are more likely to produce offspring, and thus pass on their properties to following generations. The NSGA-III algorithm, described in subsection 2.2.2, was employed in this work for selecting subsets of data channels. This was a complex problem, where a GA offered a relatively simple solution. Section 3.3 covers the usage of NSGA-III in this work.

2.2.1 Basic Structure of a Genetic Algorithm

GAs use chromosome representation to define a point in solution space. A chromosome is a set of genes, where each gene represents a variable in the optimization problem. The GA requires a fitness function that takes as input a chromosome and returns the value(s) of the objective(s) for the problem at hand. The algorithm is initialized by randomly sampling a set of N chromosomes, this is the initial population. All N individuals are evaluated with the fitness function, and a subset of the population with the best fitness is chosen. This chosen set is used as parents to produce a new population of N chromosomes, this is the second generation. Then, a similar selection is performed on the second generation, and a third generation is produced. This process is repeated until a predetermined number of generations is reached, or some other defined criteria is met.

The production of new generations is based on two operations, inspired by nature: crossover and mutation. Crossover is an operation that takes as input two chromosomes (parents) and returns new chromosomes (offspring) with genes that are some random combination of the original two. Mutation is an operation that has a chance to randomly alter the value of some genes in a chromosome. Crossover is used on the parents of one generation to produce N offspring. Mutation is added to the offspring and the result is the next generation. The whole process is illustrated in Figure 2.6

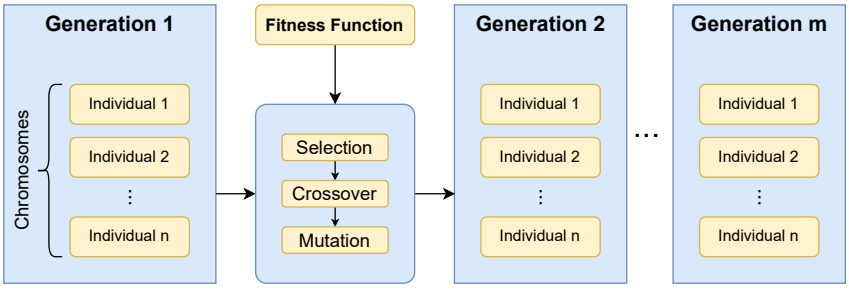


Figure 2.6: The basic structure of genetic algorithms.

GAs can be applied to different problems by suitably defining the chromosome representation and the fitness function. Multiobjective genetic algorithms (MOGAs) can also be implemented to find solutions to multiobjective optimization problems. The goal of a MOGA is to generate a Pareto front [42]. A Pareto optimal solution is a solution where no improvement can be made in any of the objectives without harming one or more of the others. The Pareto front (or non-dominated front) is the set of all Pareto optimal solutions. These solutions are also referred to as non-dominated solutions. When working with a multiobjective optimization problem we seek the Pareto optimal solutions. Other solutions are less interesting, as there exist solutions that improve on some of the objectives, without having to compromise on others. When the Pareto front (or an approximation of it) is found, some compromise has to be made to select one specific solution in this set. An example of

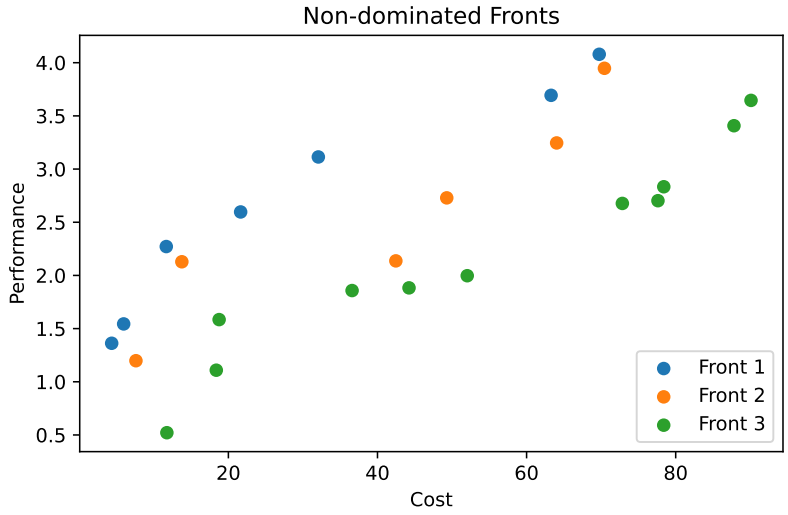


Figure 2.7: An example of Pareto fronts. If one seeks to maximize performance and minimize cost, the blue points are the Pareto optimal solutions.

such Pareto fronts is given in Figure 2.7. The figure demonstrates the concept with an optimization problem where one seeks to maximize performance, whilst minimizing cost. Each point is a candidate solution. Only the blue points are interesting solutions, since for every green or orange solution, there is a blue solution with better performance but similar or lower cost. Choosing among the blue solutions on the other hand, requires a compromise between low cost and high performance.

2.2.2 NSGA-III

NSGA-III is a MOGA based on non-dominated sorting (NS) following the NSGA-II framework [19, 40]. NSGA-III has a similar structure to that illustrated in Figure 2.6, but it also implements an extra step before the selection of parents in a generation. Each time a new generation G_i is created, a set $R_i = G_i \cup G_{i-1}$ is defined. Non-dominated sorting is applied to R_i. This sorting works by partitioning R_i into a set of groups F_1, F_2, etc. These groups are such that F_1 is the Pareto front of R_i, F_2 is the Pareto front of R_i if all elements of F_1 are removed, and so on. Each of these sets can be referred to as a non-dominated front. A new set of N individuals is selected by adding the individuals of each non-dominated front F_1, F_2, and so on in increasing order, until N or more individuals are selected. If the last added front is of such a size that more than N individuals have been added, only an appropriate subset of this front will be selected such that the final population is N. Selection, crossover, and mutation are then performed on this population as usual. Figure 2.8 shows how the NSGA-III algorithm expands on the basic GA.

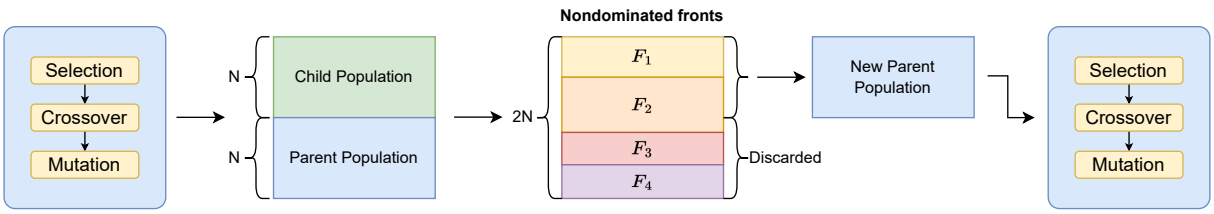


Figure 2.8: The role of non-dominated sorting in the NSGA-III algorithm.

Creating each new population by non-dominated search on the combined set of the previous population and its offspring has the advantage of preserving elitism [20]. This means that the best individuals of the parent generation have the opportunity of being kept in the new population, such that good solutions will not be lost in the crossover and mutation. Before the execution of the NSGA-III algorithm, a set of reference points has to be defined. In the cases where only a subset of the last non-dominated front is added, these reference points will be used to determine which solutions are added to the next population. Solutions close to the reference points will be selected. Defining reference points in a structured manner in the solution space can help ensure diversity in the solutions found by the algorithm.

2.3 Deep Learning

ML algorithms are computer programs that learn from data [30]. ML can be used to perform several tasks such as classification, regression, anomaly detection, and more. When referring to methods and theory of ML in this work, it is in the context of classification. Conventional ML methods typically do not operate on raw data, but require a set of manually designed features extracted from the raw data [47]. The design of such features often requires expertise in the domain of the problem at hand. In contrast to conventional ML, representation learning is a class of methods that learns such features as part of the training process. When an ML algorithm uses several consecutive layers of representation learning it is referred to as deep learning [47]. The structure of deep learning methods facilitates the learning of complex features with a high level of abstraction. This section will introduce some of the main concepts and methods of developing an ML classifier. The section also covers some theory on CNNs, a specific type of DNN especially relevant for decoding EEG signals.

2.3.1 Training and Testing of ML Classifiers

A classifier is a program meant to specify what class of n classes an input belongs to. One instance of an input is often referred to as an example. The program "learns" how to do this by being presented with a large set of training examples, each labeled with what class it belongs to. It will then use these examples to update its internal parameters such that the program classifies the training examples as correctly as possible. This is done by classifying the training examples, comparing the results with the actual classes, and computing a training error. This process is iterated for a predetermined number of times, or until some other condition is met. The goal of the training is to minimize the training error. When developing any ML classifier it is crucial to test how well the algorithm performs. There is no way to guarantee the performance of any ML classifier, unless it has been trained on all possible inputs, which for most real-world scenarios is infeasible. Instead, the conventional approach is to present the fully developed classifier with new labeled data, that has not been used during the training. In the same way, the training error is computed, this new data can be used to compute a test error. This error will then serve as an estimate of how well the classifier would perform in a real-world scenario. If the classifier performs well on this new data, it is said to generalize well, since it has found class-descriptive features that hold not only for the training data, but also in general. Depending on the application, it might be difficult and expensive to gather new data just for testing an algorithm. Therefore, it is normal to collect a large dataset and split it up into two partitions: one for training (training set) and one for testing (test set). When working with a dataset in this manner it is crucial

that the test set is not involved in any part of the development of the classifier. If the test set affects the classifier development in any way, one cannot get a true estimate of the generalization of the classifier during testing.

An ML model will typically be a function class with a set of parameters, and the model is trained by searching for optimal parameter values, such that we obtain an approximation of an optimal function. The capacity of the model is an informal measure of the range of functions that can be described by the function class [30]. Complex function classes with many parameters have a high capacity and can describe many different functions. If the capacity is too low, the function we seek to approximate might not be in the range of the function class. This is problematic, as no matter the amount of training data and training, the model will never serve as a good approximation. This situation is known as underfitting. Inversely, overfitting can happen when the capacity is too large. If this is the case, we might find a function that is more complex than the one we seek to approximate. This function might describe the training data very well, but in doing so it might also describe properties such as noise, which are specific to that data. This will lower the generalization of the model. Figure 2.9 illustrates the concept of over- and underfitting. It shows a regression task, where the function generating the data is a quadratic function affected by noise. In the case of low capacity, only linear functions can be described, hence no solution can be found that fits the data very well. For the case of high capacity, the ML algorithm finds a function that fits the data almost perfectly, however, it is quite different from the actual quadratic function. Thus, if the high-capacity model was used to predict data outside the training set, it would probably perform worse than the model of appropriate capacity. Regression was used as an example here, as it clearly conveys the problem of over and underfitting, but the concept is the same for classification tasks. Choosing a function class with appropriate capacity is a great remedy for over- and underfitting, however, for real ML applications it might be very difficult. Therefore, other methods such as regularization, a topic covered in subsection 2.3.3, are also essential for tackling these problems.

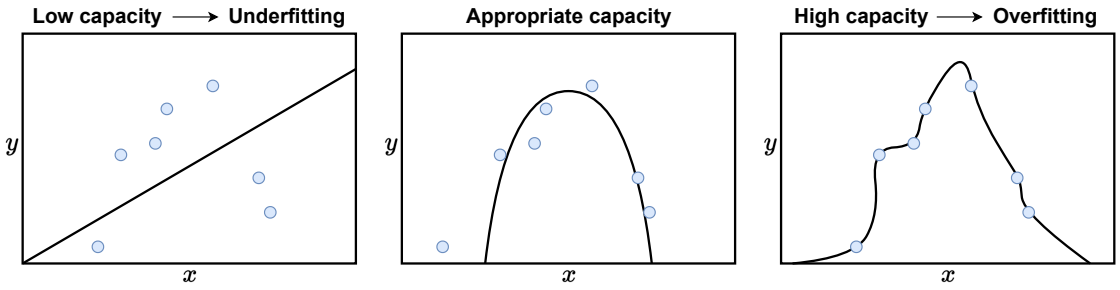


Figure 2.9: An illustration of over- and underfitting in ML (specifically for a regression task). The blue points indicate the data available for training, and the black curves are the imagined output function of three ML models with different capacities.

2.3.2 Convolutional Neural Networks

In this section, the concept of DNNs and CNNs will be introduced. However, some details of their theory will be left out, as the section is only intended to cover the basic concepts needed for discussing CNN designs and results in this work. DNN classifiers are ML models that try to approximate a function $f^*(\boldsymbol{x}) = y^*$, which takes the input example $\boldsymbol{x}$ and returns its correct class y^*. The conventional

DNN is a function $f(\boldsymbol{x};\boldsymbol{\theta}) = y$, where $\boldsymbol{\theta}$ are a set of parameters (weights) which can be adjusted. The training of the model is the process of adjusting these parameters to achieve the best possible approximation. The function is built up of a set of nodes, arranged in layers. The first layer is the input layer, the last layer is the output layer, and all layers in between are hidden layers. Each node will compute a linear combination of the outputs of the previous layer, and the parameters $\boldsymbol{\theta}$ act as weights in these linear combinations. The linear combination is then passed through an activation function. A typical activation function is the rectified linear unit (ReLU) [3]: $g(x) = \max\{0, x\}$. Another important activation function is the SoftMax function [44]. This function takes a set of real inputs and returns a set of values in the range 0-1, and their sum will be 1. Therefore these can be interpreted as probabilities. In classifiers, the last layer can have as many units as classes and use the SoftMax activation to produce a probability distribution for the classes. The layers of a conventional DNN are often referred to as fully-connected layers, as each node in one layer has a separate parameter for each node in the next layer, "connecting" them. An example of such a conventional DNN is shown in Figure 2.10. In this example, the node $h_{1,1}$ would have the output as defined in Equation 2.1.

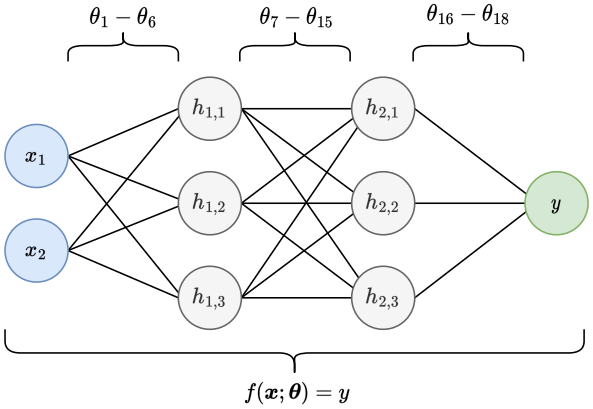


Figure 2.10: Basic structure of a DNN. x_1 and x_2 are the inputs to the network, $h_{1,1} - h_{2,3}$ are the nodes of the hidden layers, and y is the output. $\theta_1 - \theta_{18}$ are the parameters of the model.

$$h_{1,1} = g\left(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}\right) \tag{2.1}$$

Since a DNN is a type of representation learning, it can take raw data as its input, and learn features from this data. However, when working with raw data, such as images or time series, the dimensionality of an example can be large, for instance, an image of 256×256 pixels. In a fully-connected neural network with for instance 256 nodes in the first layer, this would yield $256 * 256 * 256 \approx 16.7 * 10^6$ parameters in just the first layer of the network. CNNs are proven to be an effective alternative to using fully-connected layers, able to work with the same raw data, but using far fewer parameters [30].

In CNNs, the linear combinations of the conventional DNNs are replaced with convolution operations. The discrete convolution is an operation between two discrete functions defined as:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[n-m]g[m] \tag{2.2}$$

In CNNs, one of the functions will be the input to the layer, and the other will be the parameters of the

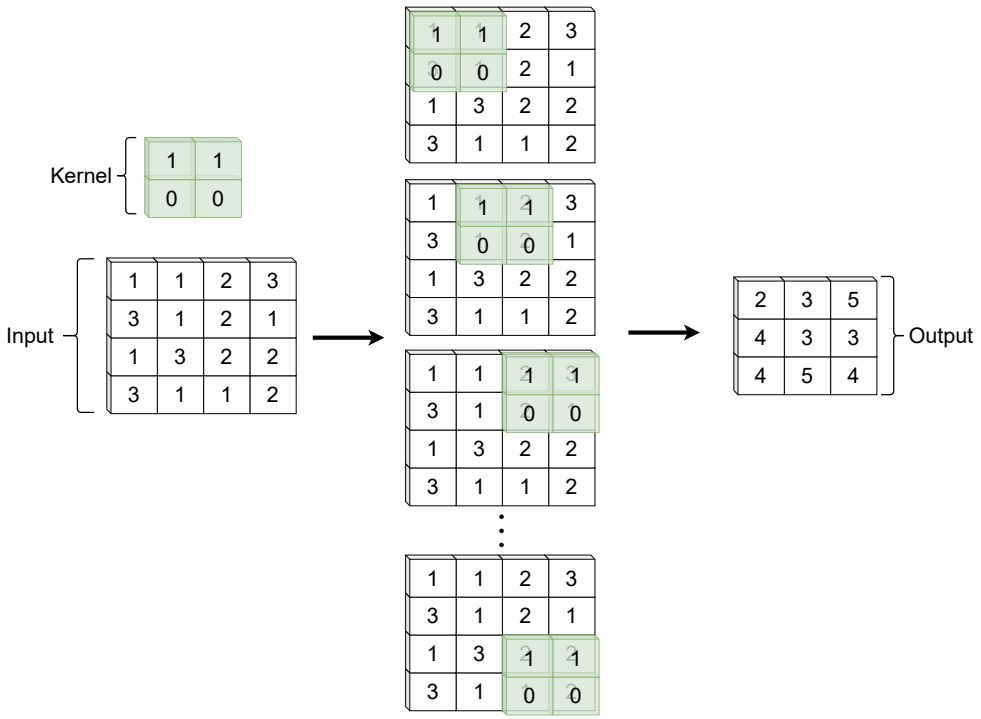


Figure 2.11: An example of a 2D convolution operation.

layer (referred to as the kernel of the operation). Naturally, neither of these will be functions defined for the whole set of integers, rather they are sequences of a finite length. Hence, the definition in Equation 2.2 will not be valid for all values of m or n. The operation performed in CNNs is restricted to the valid values, based on the length of the sequences. This modified convolution can be described as:

$$(f * g)[n] = \sum_{m=0}^{m=M} f[n+m]g[m], \text{ for } n \in [0..N-(M-1)] \tag{2.3}$$

In Equation 2.3, N is the length of the input sequence and M is the length of the kernel. A more visual way of interpreting this operation is as described in Figure 2.11. This figure illustrates a two-dimensional convolution, but the idea is the same as for the one-dimensional convolution. A result of this operation is that the size of the output will be smaller than that of the input.

Since the same kernel is used across the whole input, the number of parameters will not be dependent on the size of the input, just the size of the kernel. Similarly to conventional DNNs, each layer can have several outputs, by having several kernels each producing its own output. These kernels are often referred to as filters. In Figure 2.11 the kernel is shifted one index at a time, however, most CNN frameworks allow for the shift to be defined with a parameter. This parameter is called stride and will have the same number of dimensions as the kernel. To complete a CNN classifier, the features learned in the convolutional layers can be used as input in a fully-connected layer, to produce a one-dimensional output vector with the class probabilities.

2.3.3 Regularization

Although the aim of the training phase is to minimize the training error, fitting the algorithm to the training set, the final goal of the ML algorithm is to minimize the test error, fitting the algorithm to any

new data. Regularization in ML is a wide expression, covering strategies that are designed to lower the test error, but potentially increase the training error [30]. The following list gives a brief introduction to the regularization techniques employed in this project:

- **Dataset augmentation:** These are methods for creating fake training data, such that the training set increases, yielding better generalization. There are many ways of creating such data, often the idea is to apply some transformation to the training set to create another training set with the same labels.
- **Max-norm constraints:** Constraints can be applied to the parameters of the model, such that their norm cannot exceed a certain value. This effectively reduces the capacity of the model, as it has a more limited range of functions it can approximate.
- **Dropout:** This is a method whereby random parts of a DNN are dropped during training [64]. The result is a reduction in overfitting, as the algorithm cannot depend on any one part of the net, instead, each part of the net has to learn useful representations of the data.

2.3.4 Layers of the Keras Framework

Keras [8] is a Python framework for developing neural networks (NNs). In Keras, NNs are defined as a sequence of layers. These layers can be dense or convolutional, as those described in subsection 2.3.2, but other layers implement dropout and other techniques as well. All DNNs created and used in this project has been developed in the Keras framework. This section will provide some theory of the layers used in the architectures described in sections 3.4.2, 3.4.3, and 3.4.4.

- **Dense:** Fully-connected layer. It takes as a parameter the number of units in the layer.
- **Conv2D and Conv3D:** Convolutional layers, for either two- or three-dimensional input data. They take as parameters the number of filters in the layer, kernel size, and stride.
- **Dropout:** This layer will set some of the input units to zero, dropping them, at random. It takes as a parameter the rate of units to drop. This layer will only be active during training, when using the model for prediction this layer is inactive.
- **BatchNormalization:** This layer outputs a normalization of its inputs, by moving the mean towards 0 and the standard deviation to 1. During training the layer computes a mean m and a standard deviation σ of the inputs. When the model is used for prediction, these values are used to normalize inputs.
- **Activation:** Implements standard activation functions such as the ReLU and SoftMax described in subsection 2.3.2.
- **Flatten:** This layer takes inputs of any dimensions and reshapes them into a one-dimensional vector. In CNNs, this layer is typically used as the connection between the multi-dimensional convolutional part and the fully-connected part of a CNN classifier.
- **MaxPooling2D and MaxPooling3D:** Max pooling is used to downsample along two or three dimensions in the data. The layer will be configured with kernel size and stride in the same way as Conv2D and Conv3D. The kernel will then scan over the input similar to the convolution

operation in Figure 2.11, but instead of producing features, the maximal input value for each region will be outputted.

2.3.5 Hyperparameter Tuning

For most ML algorithms (especially DNNs) there will be a number of parameters that govern how the algorithm behaves, but that cannot be included in the set of trainable parameters. Such parameters are called hyperparameters [30]. Examples of such parameters are the number of layers and neurons in a DNN, the probability of dropout, and the type of activation function used. The choice of hyperparameters is important. For instance, the number of layers in a DNN will directly affect the capacity of the model, which, as Figure 2.9 demonstrated, can be crucial for the performance of the algorithm. Although some hyperparameters can be selected with a background in theory and reasoning, it is useful to explore different configurations and select the best one, a process known as hyperparameter tuning. A hyperparameter configuration can be evaluated by training the algorithm with the configuration, and then testing the resulting model on a new set of data, called a validation set. As touched upon in subsection 2.3.1 it is important that this validation set is not the same set used for the final testing, as this introduces the risk of hyperparameters being overfitted to the data. Therefore, to allow for hyperparameter tuning, the dataset in an ML project is often divided into three sets: training, validation, and test, as illustrated in Figure 2.12. The proportion of data used in each of these sets is of course a free choice, however, it is desired to keep as much data as possible for training.

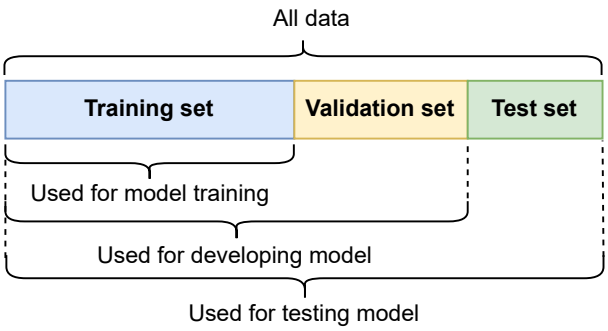


Figure 2.12: The typical segmentation of the dataset in an ML project.

Chapter 3

Materials and Methods

3.1 Dataset

When working with ML, the data used to train and test models play a crucial role in what results are obtained, and how one should interpret them. Most often, ML research has the intention of discovering approaches and solutions for real-world problems and tasks. Therefore, understanding the data used in the research is vital for assessing the possibility of applying the research to these real-world scenarios. This section provides a description of all the data used in this project, and how it was collected, structured, and preprocessed. This data includes EEG recordings, magnetic resonance imaging (MRI) recordings, and digitized electrode positions. The EEG data is the data that is directly used in the training and testing of the ML classifiers. Therefore, it is this data that is referred to when the "dataset" is mentioned throughout this paper. The data used in this work has been subject to study in previous research [52], and in the preliminary work for this research [27].

3.1.1 Recording and Collection

The data was recorded at the Aalto Neuroimaging facility, Aalto University, with the intention of studying the human brain's response to visual primary color stimuli. It includes the recorded brain activity (EEG measurements) of 31 subjects, both male and female, during their exposure to the primary colors red, green, and blue (RGB).

EEG Recording

The EEG recording was performed with a 60-electrode EEG-cap. Two of these channels were used to record ocular activity, electrooculography (EOG). The remaining 58 channels were EEG channels. Figure 3.1 shows the layout of the EEG electrodes, which is similar to the 10-10 standard, but with some electrodes left out. The recording was performed inside a magnetically shielded room, and MEG recording was performed simultaneously. MEG data has not been used in this research. A stimuli signal was also produced, and synchronized with the EEG recording. This signal indicates what stimuli were being applied (red, green, blue, or gray) at any given time.

During the recording, the subjects were placed in front of a display, which alternated between showing gray and a random primary color. In total, each subject was stimulated with each color at least

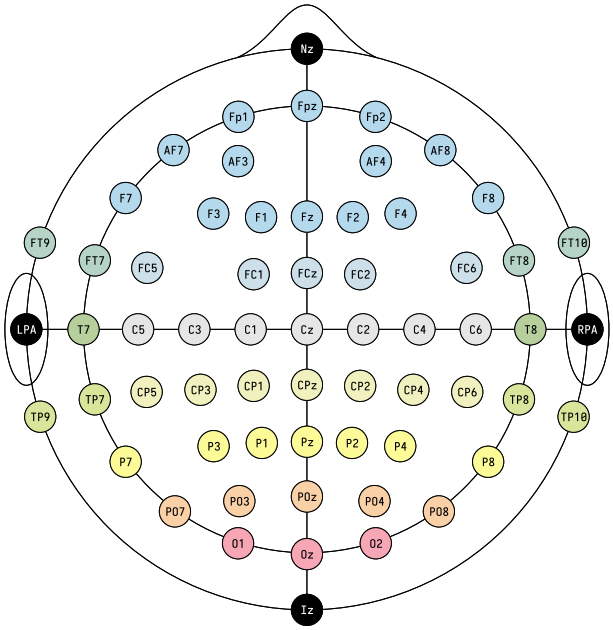


Figure 3.1: The layout of the 60-electrode EEG-cap used to record the dataset. The figure is adapted from [14].

140 times. Figure 3.2 describes the protocol for applying these stimuli. Since undesired behavior of a subject, such as lack of focus or excessive movement, can affect the EEG data, the subjects were observed during the recording, and notes were taken of any undesirable behavior. These notes will be provided later, in Table 3.1. After the recording, it was discovered that some electrodes were faulty for some of the subjects. Another session of recording was performed for most of these subjects, with all electrodes working.

MRI and Digitized Electrode Positions

MRI scans were performed on the head of each subject, and the positions of the electrodes relative to each other in 3D space were measured. This data is needed to construct a mathematical model of the transformation between electrical signals in the neuron of a subject, and the electrodes on their

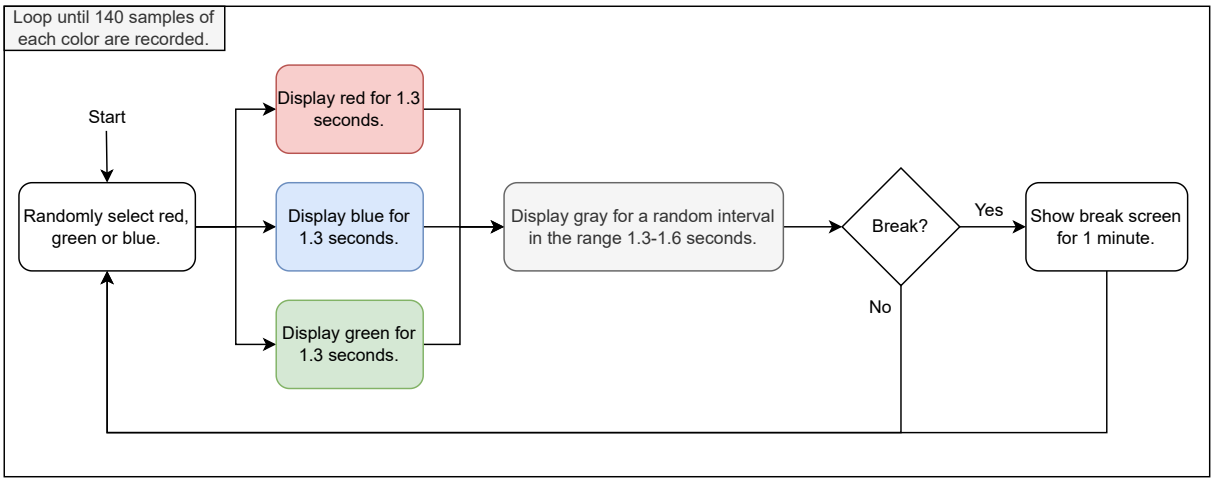


Figure 3.2: The stimuli protocol used for applying color stimuli during the recording of the EEG data.

scalp. Such a model is used in SR, which will be described in section 3.2.

3.1.2 Preprocessing

The raw EEG data is one continuous time series for the whole recording session. Before the data can be used to train and test ML classifiers, the sections of the time series corresponding to stimuli responses have to be extracted and labeled. This process is called epoching. The basic principle of this epoching is demonstrated in Figure 3.3. The first plot shows a section of raw data from one electrode. The second plot shows the data of the stimuli channel for the same period of time. Every time the color of the display changes, the stimuli channel will receive a pulse with a certain value corresponding to the type of stimuli. For this particular example, there are three stimuli events, one of each color, separated by the gray screen. During epoching, the peaks of the stimuli channel are detected and used to extract data and label it, such as the last plot of Figure 3.3 shows.

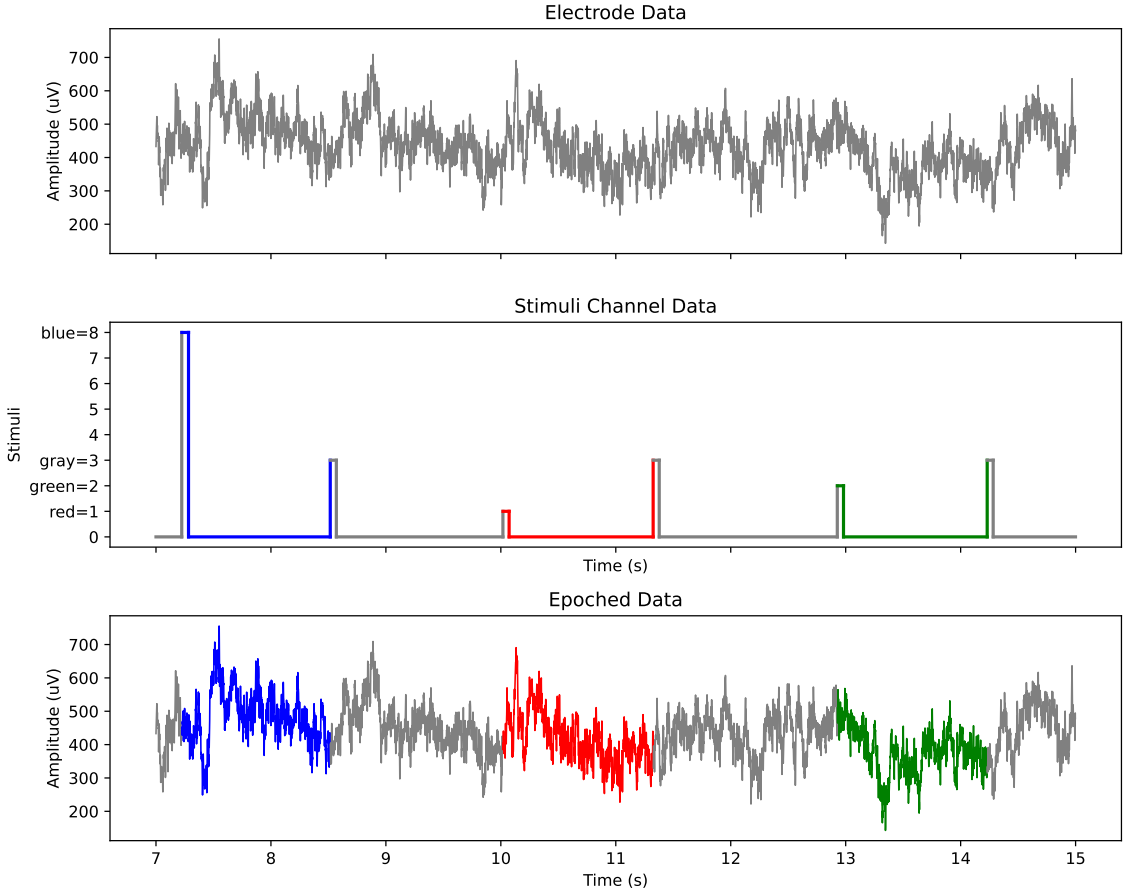


Figure 3.3: The epoching process. The raw electrode data is cropped and labeled based on the values of the stimuli channel.

Some preprocessing can also be performed to increase the SNR of the data, such as low-pass filtering and artifact removal. Two different preprocessing pipelines have been developed and explored in this work. The first, which will be referred to as "full preprocessing", is the same preprocessing as the one developed in the primary work [27]. The second, which will be referred to as "minimal preprocessing", applies almost no preprocessing.

Full Preprocessing

The full preprocessing paradigm applies a number of methods with the intention of increasing the SNR of the data. A summary of the applied methods is provided in the list below:

- **Notch filtering**: To remove powerline noise.
- **Downsampling**: Applied to reduce the amount of data.
- **Filtering**: Filters out frequencies in the data that are not of interest.
- **Signal-space projection (SSP)**: Eye blinks in the data are detected on EOG-channels, and used to apply a projection for removing these artifacts in the EEG data.
- **Epoching**: Epochs are extracted. Epochs with too large peak-to-peak (PTP) amplitude are discarded. Epochs with blinks occurring too close to stimuli onset are discarded.
- **Baseline correction**: Data of each epoch is shifted based on the average value of the data some time before stimuli onset. This is done to correct for any slow-moving shift in electrode values.
- **Standard Scaler**: The mean and standard deviation of data used for training are calculated. Then each sample is scaled by subtracting the mean and dividing by the standard deviation. This is done on a subject-specific basis.

A more thorough description of this preprocessing pipeline is provided in [27].

Minimal Preprocessing

Since exhaustive preprocessing significantly increases the computation, it is undesirable for online applications. Moreover, some of the methods in the full preprocessing results in discarded epochs, thus less data is available for training and testing classifiers. With a background in this, a pipeline with minimal preprocessing was also explored. Only baseline correction and standard scaling are applied to the minimally preprocessed data.

3.1.3 Structure of the Dataset

After epoching, the dataset has its final shape: $n_{\text{subjects}} \times n_{\text{epochs}} \times n_{\text{electrodes}} \times n_{\text{samples}}$. Since one epoch serves as an input to the classifier, the input shape is $n_{\text{electrodes}} \times n_{\text{samples}}$. The goal of this project, differs somewhat from most conventional ML tasks, as the aim is not to find a specific DNN model with trained weights, but rather to find a DNN architecture that generalizes well across subjects. To facilitate this, the data was structured as illustrated in Figure 3.4. Firstly, the subjects are divided into two groups: validation subjects and test subjects. The purpose of this division is to use data from the validation subjects to explore different DNN architectures and configurations, to find an approach that works well, independently of the individual it is trained on. The test subjects can then be used to verify how well the DNNs perform when trained and tested on data from new individuals. The goal is to create subject-specific classifiers, hence the only parameters that are "learned" and transferred from the validation subjects to the test subjects are hyperparameters and model architecture. As Figure 3.4 indicates, the examples (epochs) of each subject will also be divided into two sets: a training set and a test set. These are the conventional datasets for training and testing classifiers. In this way,

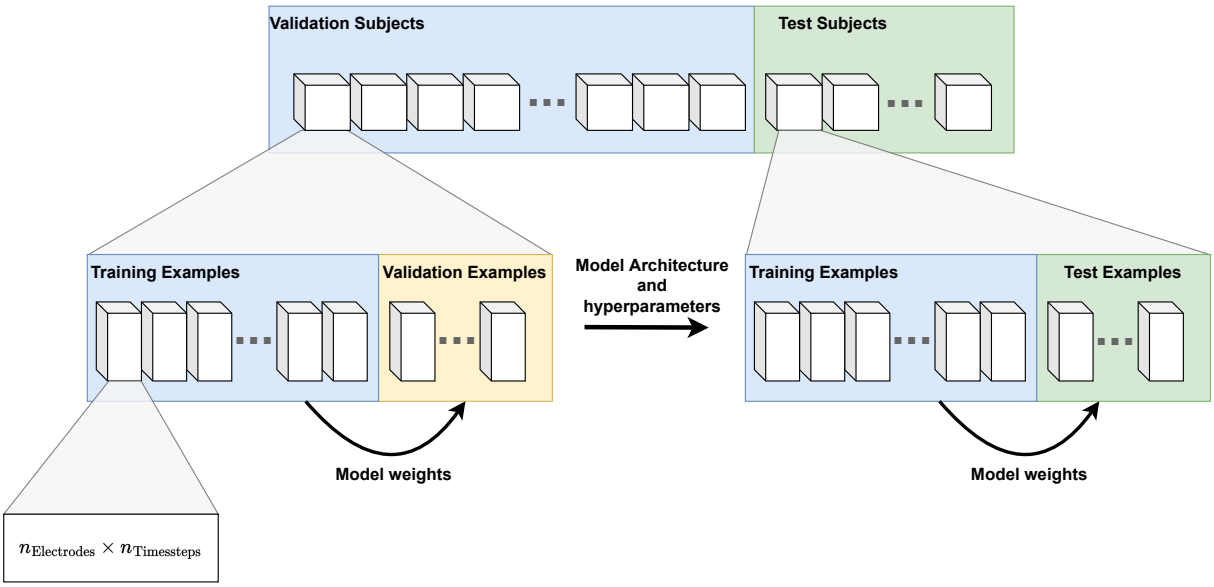


Figure 3.4: The structure of the dataset. The figure is adapted from the preliminary work [27].

each subject defines its own ML problem. Figure 3.4 summarizes the dataset used for classification. The selection of validation subjects was done by random selection of eight subjects. Only data from the eight validation subjects (VAL8) were used during the development and exploration of methods. The remaining data was used in the tests introduced in Chapter 4.

Subject	Used for validation	Number of epochs (full preprocessing)			Number of epochs (minimal preprocessing)			Notes
		Red	Green	Blue	Red	Green	Blue	
sub-01	No	133	131	139	140	140	140	Oz and O2 not working
sub-02	No	137	139	137	140	140	140	
sub-03	Yes	140	137	135	140	140	140	
sub-04	No	127	120	116	140	140	140	
sub-05	No	132	134	136	140	140	140	
sub-06	No	138	140	138	140	140	140	
sub-07	Yes	121	120	116	140	140	140	
sub-08	No	131	128	135	140	140	140	
sub-09	No	70	70	76	151	157	152	Oz and O2 not working and subject moved constantly
sub-10	No	126	122	123	140	140	140	Oz and O2 not working
sub-11	No	138	136	136	140	140	140	
sub-12	No	112	101	87	140	140	140	Oz and O2 not working
sub-13	No	139	140	140	140	140	140	
sub-14	No	140	141	140	140	140	140	
sub-15	No	131	131	128	140	140	140	
sub-16	No	135	133	139	140	140	140	CP3, CP5 and TP9 not working
sub-17	No	131	127	128	140	140	140	Oz and O2 not working and subject fell asleep
sub-18	Yes	136	137	138	140	140	140	
sub-19	No	123	118	118	140	140	140	
sub-20	No	139	139	140	140	140	140	
sub-21	No	115	113	105	140	140	140	
sub-22	Yes	111	129	117	140	140	140	Subject may have fallen asleep
sub-23	No	129	130	128	140	140	139	
sub-24	No	139	139	138	140	140	140	
sub-25	Yes	132	128	128	140	140	140	
sub-26	Yes	137	137	139	140	140	140	
sub-27	No	70	73	71	140	140	140	Subject fell asleep
sub-28	No	139	138	136	140	140	140	
sub-29	No	127	128	127	140	140	140	
sub-30	Yes	139	138	138	140	140	140	
sub-31	Yes	140	140	139	140	140	140	

Table 3.1: The dataset used for classification. The notes indicate if certain electrodes were not working properly and if the subject's behavior was not correct.

3.2 Source Reconstruction

EEG signals are not a direct measurement of the neuronal activity in the brain. Each neuron contributes with an electric potential, and this potential travels through the head volume before reaching the EEG electrodes. The conductive properties of the tissues between the neuron and the electrode will result in attenuation, distortion, and blurring of the signal [35]. This is called the head volume conduction effect. The signal measured at each EEG electrode is thus a mixture of the electric activity of all neurons in the brain. SR is a method aimed at unmixing EEG signals, to obtain a more accurate representation of the brain activity, with higher spatial resolution [53]. The method consists of two main steps: creating a forward model and solving the inverse problem. The forward model is a model describing the transformation from the signals in the brain to the signals measured on the electrodes. Since this model describes a transformation of signals from one space to another, the signals measured on electrodes are said to be in electrode space, and the original signals in the brain are said to be in source space. The inverse problem is the problem of estimating the source space signals from EEG signals, using a known forward model. Figure 3.5 illustrates the connection between the source- and electrode space. This section will cover the main steps of implementing SR in general, as well as the specifics of its implementation in this project.

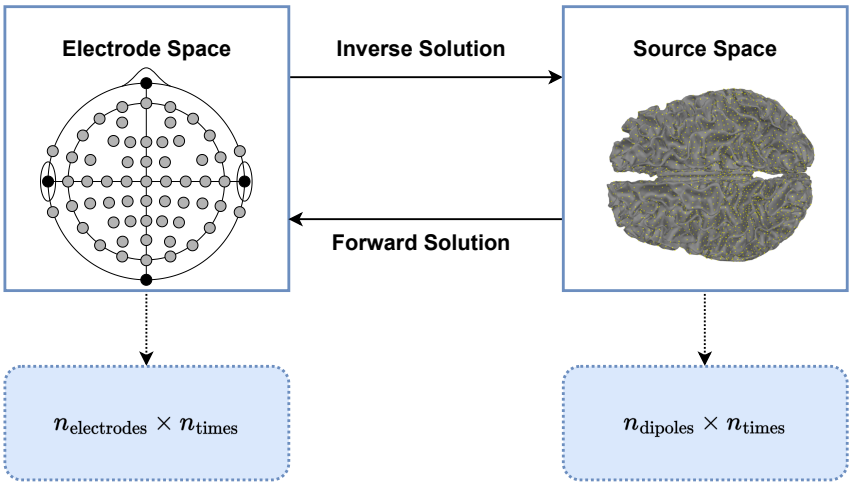


Figure 3.5: The relationship between source space and electrode space. The figure is adapted from the preliminary work [27].

3.2.1 Forward Model

The forward model describes the physical process that transforms brain activity into electric potentials in the EEG electrodes. How accurate this model is, will affect the accuracy of source estimates found by SR. Depending on the situation one might create a forward model that is tailored to a specific individual, for higher accuracy, or create a more generalized but less accurate model. This will depend on the data and computational resources available, as will be further explained below. The forward model can be divided into three main parts: A source space definition, a head model, and an electrode space definition. This structure is illustrated in Figure 3.6.

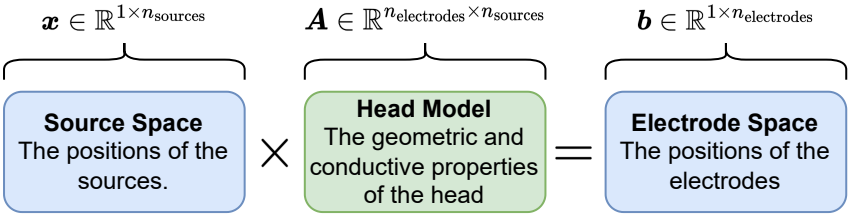


Figure 3.6: The structure of the forward model.

Head Model

The goal of the head model is to describe the volume conduction in the head, by describing its shape, and the conductive property in each point of the volume. Head shape models used in SR vary from simple shapes such as spheres, to complex realistic shapes based on MRI recordings [35]. Similarly, the conductivity can be described simply by one homogeneous volume, or more realistically by segmenting the shape into different areas with different conduction, such as skull, skin, and so on. In this project, the MRI recordings of each subject are available and have been used to create individual forward models. A common way to create a head model based on MRI is the boundary element method (BEM) [28]. With this method, the head volume is segmented into three different compartments: the skin, the skull, and the brain. Each of these compartments is considered to have known homogeneous conductivity. The FreeSurfer [25] software was used to employ BEM in this project. An illustration of the different compartments created during the procedure is provided in Figure 3.7. Values for the conductivity of the different compartments were chosen according to optimal values for BEM found in a previous study [65].

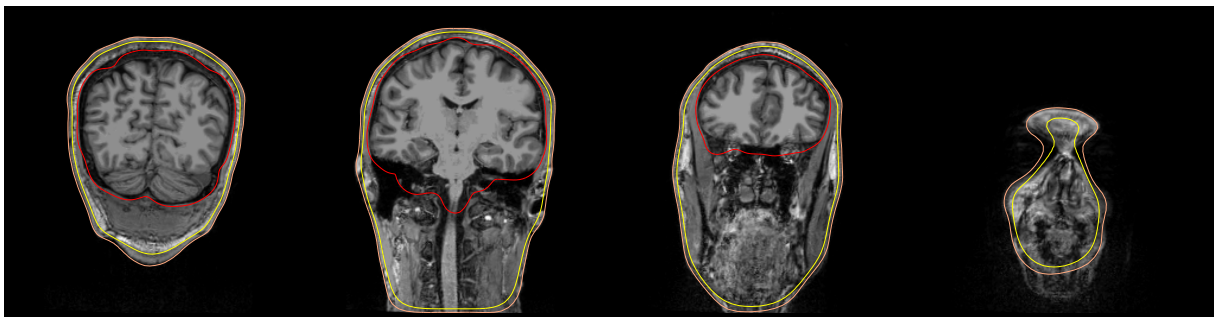


Figure 3.7: Example of three-layer BEM head segmentation done in FreeSurfer. The orange boundary indicates the scalp surface, the yellow boundary indicates the skull surface, and the red boundary the brain surface.

Source Space

The signals measured with EEG originate from the many billion neurons inside the brain, however, it is neither feasible nor necessary to reconstruct the current in every single neuron. Instead, we can model the brain activity as a set of current dipoles (sources) spread out in the brain volume [33]. As with the head model, MRI can be used to create a set of realistic source positions, based on actual brain geometry. Usually, the source space is restricted to the gray matter on the surface of the brain (the cerebral cortex), as this is the main source of the electric potentials measured with EEG [33]. In this project, FreeSurfer was used to create segmentations of white and gray matter from MRI data. An

example of such a segmentation is provided in Figure 3.8. The MNE-Python software was then used to distribute 8196 sources evenly across the border between the white and gray matter. This border is defined by the red shapes in Figure 3.8. Figure 3.9 shows the distributions of the sources for one subject.

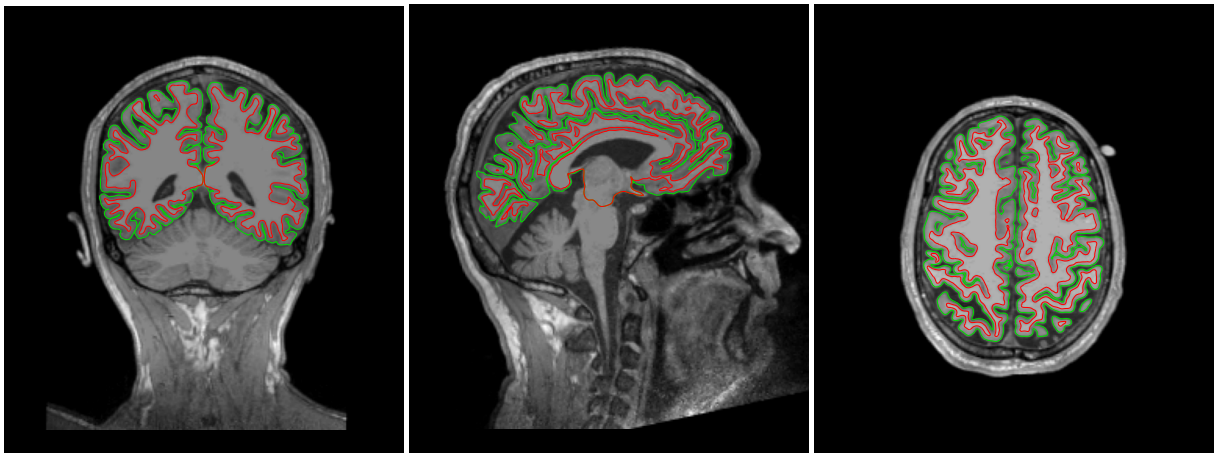


Figure 3.8: Example of the white and gray matter segmentation. The white matter is the area inside the red border. The gray matter is the area between the red and the green borders. Thus the red border forms the gray-white matter boundary.

Electrode Space

Similarly to the sources, the position of each electrode has to be defined in order to compute the forward model. As described in section 3.1, the positions of the electrodes relative to each other were recorded. However, the electrode positions are not defined in the same frame as the MRI data. Hence, a transformation is needed between the frame of the MRI data and the frame of the electrode positions. The process of finding such a transformation is called co-registration, and MNE-Python provides a tool for this purpose. Using this tool involves manual alignment of electrodes to the head shape (provided by MRI), by moving and scaling the frame of the electrodes. The tool then stores the transformation matrix yielding this alignment.

Lead Field

When the source space, head model, and electrode space are properly defined, the lead field can be calculated. This is a matrix $\boldsymbol{A} \in \mathbb{R}^{n_{\text{electrodes}} \times n_{\text{sources}}}$, describing a linear transformation from source space to electrode space [18]. With this model, the potential at an electrode is a linear combination of the current density of each source. The weights of this linear combination, the values of each row in $\boldsymbol{A}$, are calculated based on the known positions of sources and electrodes, and the conductivity of the volume between them. The final forward model can then be described by the linear equation:

$$\boldsymbol{b}(t) = \boldsymbol{A}\boldsymbol{x}(t) \tag{3.1}$$

Where $\boldsymbol{b}$ are the electrode potentials and $\boldsymbol{x}$ are the source currents. Figure 3.6 indicates the dimensions of the elements of Equation 3.1. The use of such a linear model is an approximation, as it will only account for attenuation in signals.

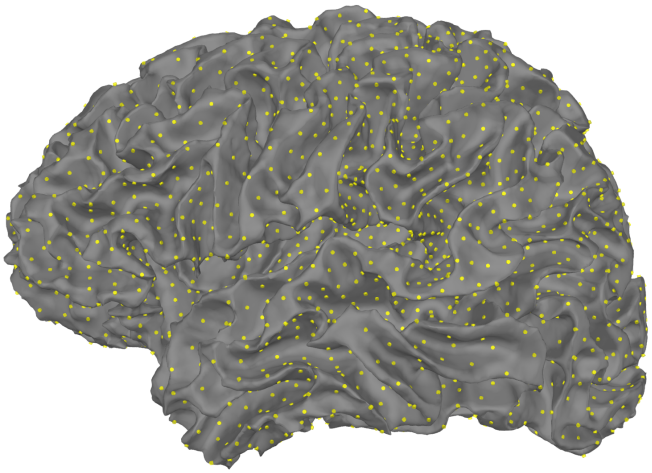


Figure 3.9: The distribution of sources on the white matter surface. This is a left-side view of the white matter of a subject. Each yellow point is a source.

3.2.2 Inverse Problem

The inverse problem is the problem of estimating current distributions in source space by using a forward model and electrode space data. The system described in Equation 3.1 is underdetermined, as there are more unknown variables (sources) than known variables (electrodes) [59]. There are over 8000 sources, and only 58 electrodes, so many different current distributions in source space will yield the same measurement at the electrodes. Presented with only a set of electrode measurements, there is no way of knowing which of these current distributions was the cause. Mathematically, this manifests itself in A not being invertible, thus no unique solution can be found solving Equation 3.1 for source estimates $x(t)$. To obtain a unique solution, a priori knowledge has to be used to form constraints on the problem. A commonly used method for doing this is the minimum norm estimate (MNE) [53]. In this method, two a priori assumptions are made [16]:

- The source estimates $\boldsymbol{x}(t)$ are assumed to be normally distributed with zero mean and covariance matrix $\boldsymbol{R}$.
- The measurements $\boldsymbol{b}(t)$ are affected by additive noise such that $\boldsymbol{b}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{n}$, where the noise $\boldsymbol{n}$ is normally distributed with zero mean and covariance matrix $\boldsymbol{C}$.

As the name suggests, the method aims at minimizing the error of the source estimates. This means it seeks a linear operator $\boldsymbol{W}$ such that:

$$\boldsymbol{W} = \arg\min_{\boldsymbol{W}} ||\boldsymbol{W}\boldsymbol{b} - \boldsymbol{x}||^2 \tag{3.2}$$

This linear operator is time-independent and can be used to find an estimate for the source distribution at any time, given the electrode measurements. The solution to Equation 3.2, considering the aforementioned assumptions is [16]:

$$\boldsymbol{W} = \boldsymbol{R}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top + \boldsymbol{C})^{-1} \tag{3.3}$$

Once the linear operator $\boldsymbol{W}$ is calculated, source estimates $\hat{\boldsymbol{x}}$ are found with:

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{W}\boldsymbol{b}(t) \tag{3.4}$$

The covariances $\boldsymbol{R}$ and $\boldsymbol{C}$ act as parameters in the MNE method. Since $\boldsymbol{R}$ describes the correlation between the different sources it can serve as a parameter for spatially smoothing the source estimates. Since EEG data is available, an estimate of $\boldsymbol{C}$ can be computed using this data. A number of methods expanding on the basic MNE method have been proposed, and several are available in the MNE-Python software. In this work, the dSPM method in MNE-Python was used, which is a method introduced in [17].

3.2.3 Template Forward Models

The SR described in the previous sections is an individual process, it requires MRI and other data recorded of the individual. Hence, if SR is to be used in online applications such as BCIs, they cannot be plug-and-play for the user. They would require an MRI recording session for each new user, and subsequent configuration of the system using that data. Since the source space created for each individual is different, the source space data from different individuals are difficult to compare. In electrode space, electrodes are placed based on standard rules, such that the data recorded on a specific electrode has approximately the same position relative to the brain for different subjects. Thus, it is reasonable to assume that the recordings on for example Fp2 have the same properties across subjects, and that the data can be compared. In the individual source space, there is no such labeling and standard layout.

To overcome the issues caused by the individual source space data, several studies have been conducted to develop template forward models [37]. These templates are created from MRI recordings of a group of subjects and serve as an approximate model of the average human head. Although all heads are different, there are some common factors. Thus, a template model can still provide a priori information in the SR method. Naturally, the individual forward models are better approximations [37], but the templates produce data in a standard source space. In this project, a template forward model was used to search for good subsets of sources to use for classifying RGB stimuli. This procedure is described in section 3.3. The FsAverage template from FreeSurfer was used for this purpose. It is an average head model, based on MRI data from 40 individuals [25].

3.3 Channel Selection with Genetic Algorithm

Throughout this work, the terms "channels," "electrodes," and "sources" will be frequently used. It is important to note that in this work the term "channel" encompasses both electrodes and sources, depending on the specific context. In the preliminary work conducted in [27], the classifiers were tested with all electrodes and with a selected subset of eight electrodes close to the occipital lobe. Similarly, in source space, the classifier was tested with all sources and with a subset of sources located in the occipital lobe. There was a small increase in accuracy when using the subsets, instead of all channels, in both source- and electrode space. These results demonstrated that the choice of channels is important for the task at hand. If some channels do not contribute information relevant to the task, including them in the dataset might lower the SNR. If this is the case, there should exist an optimal subset of channels to include in the dataset, for maximizing the classification accuracy. Using fewer channels is beneficial for online applications, as it requires less computation, and in the case of electrodes: less hardware. Thus, finding a subset of channels to include is desirable in itself. One way to search for an optimal channel configuration is by using a GA. This has already been explored in the context of EEG decoding [54]. The results showed that including only a subset of the electrodes in the dataset could increase the accuracy of decoding epileptic seizures. In light of this, the genetic algorithm NSGA-III [19, 40] was used to search for optimal channel configurations in both electrode space and source space. This section presents the methods used to perform this channel selection, and the results obtained.

3.3.1 Implementation

The problem of choosing an optimal channel configuration is an optimization problem. However, it cannot be described on the conventional form, as an equation with a set of input variables and output objectives. The process for evaluating a configuration involves creating and training a classifier with the appropriate dimensions and testing it. This process is both complex and non-deterministic, as the training involves stochastic operations. GAs are a group of methods well suited for solving such problems, as they treat the objective function as a black box. Defining a GA paradigm for the channel selection problem requires the formulation of two GA components:

- **Chromosome representation:** A datatype must be defined that can represent any channel configuration.
- **Fitness function:** A function that takes as input a chromosome (channel configuration) and returns an objective value, reflecting how well the classifier performs with that chromosome.

The rest of the GA is independent of the problem, therefore, the GA itself does not need to be implemented for this exact problem. A multi-objective optimization framework for Python, called pymoo, was used to implement the GA [6]. This framework has an implementation of the NSGA-III algorithm, which was used. Whether the channels to be selected are electrodes or sources does not affect the chromosome representation or the fitness function, except for a few parameters, which will be explained below.

Chromosome Representation

Any configuration of n channels can be represented as a list of n binary values, where the i-th element represents whether the i-th channel is to be included in the subset or not. The chromosome representation was chosen to be such a list. Each element of the list, each gene, could take the value 0 or 1, representing whether the corresponding channel should be included or not. Figure 3.10 illustrates the chromosome representation. The representation is valid for both sources and electrodes, the only difference is the length of the list, which has to be either n_{sources} or $n_{\text{electrodes}}$ respectively.

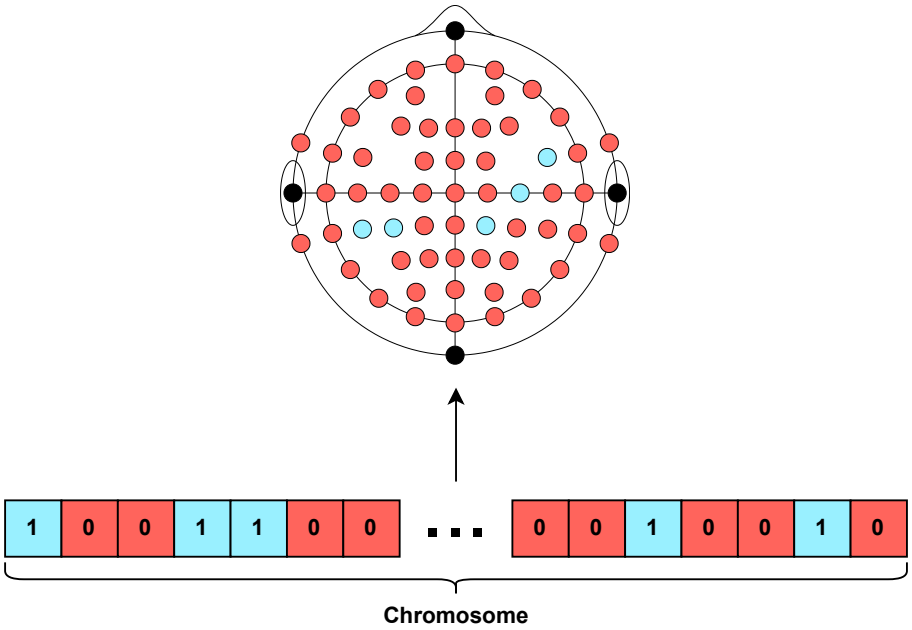


Figure 3.10: The chromosome representation. A binary array of length n_{channels} is used to represent a unique channel configuration. The configuration used is an example of an arbitrary electrode selection.

Fitness Function

The channel configuration serves as a kind of hyperparameter with respect to the DNNs used for the classification. Therefore it has to be set before the training of the DNN. The evaluation of a given channel configuration thus involves: building a DNN with correct dimensions, training it, and then testing it. Finally, the function will return the accuracy achieved during testing. This process is illustrated in Figure 3.11. The best classifier from the preliminary work, deepConvNet [62], was used for classification in the fitness function. Since the channel selection is a form of hyperparameter tuning, it cannot be done using all data available, as no data would be left for the final testing of the classifiers. The VAL8 subjects were used to produce the accuracy returned by the fitness function. This means that the channel configurations returned by the GA will be suitable for these eight subjects. Whether these configurations were suitable also for the other subjects, is discussed in subsection 5.2.3. For the selection of sources, the dataset used to train and test the DNN was source space data, and for electrode selection, data in electrode space was used. Except for this difference, the fitness function is identical for the two scenarios. The fitness function also returns the number of selected channels, such that the GA has two objectives to optimize. Accuracy should be maximized and the number of

channels should be minimized.

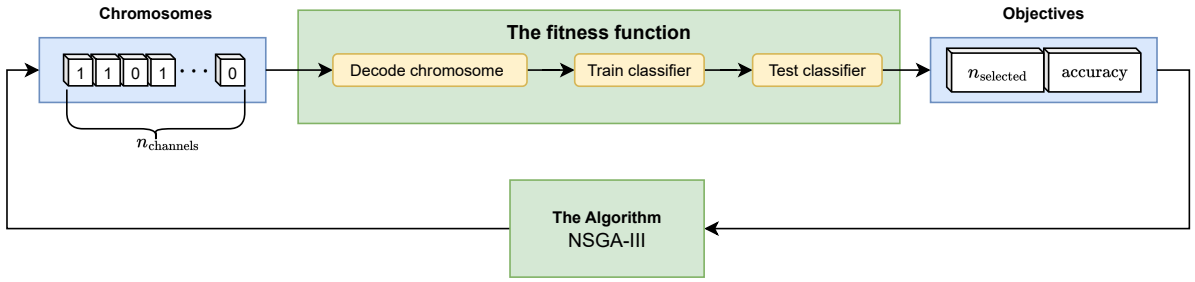


Figure 3.11: The fitness function. For each chromosome, a new DNN is trained and tested. Then, the accuracy and number of selected channels are returned.

3.3.2 Electrode Selection

The electrode selection was run with a population size of 20, for 100 generations. All solutions returned by the GA are presented in Figure 3.12. These are five non-dominated solutions, using 4, 5, 6, 7, and 10 electrodes. Figure 3.13 shows which electrodes were selected for the 4- and 10-electrode solutions. These results indicate that using more than the 10 electrodes will not yield any increase in accuracy. To better compare these solutions to the solutions found using all electrodes in [27], a new training and testing was run using these solutions, to get the accuracy of each individual subject in each configuration. Table 3.2 shows the results of this test compared to the results of [27]. The results in this second test are somewhat different from those returned by the GA. In [27] it was found that the accuracy of classifiers had a standard deviation of 0.04, so such a difference is to be expected. In the second test, all of the configurations had a lower average accuracy than that achieved with all electrodes. However, there is a difference of only 0.01 for the 10-electrode configuration and 0.03 for the 4-electrode configuration (both within the standard deviation found in [27]). As Figure 3.13 shows, the most important electrodes for achieving high accuracy are placed close to the occipital lobe. Although the 10-electrode configuration does include electrodes in the frontal, central, and temporal areas of the brain, it does also include all electrodes used in the 4-electrode configuration.

Subject	All electrodes	10 electrodes	7 electrodes	6 electrodes	5 electrodes	4 electrodes
sub-18	0.92	0.90	0.90	0.92	0.90	0.90
sub-26	0.86	0.77	0.78	0.82	0.87	0.82
sub-07	0.85	0.88	0.82	0.83	0.85	0.82
sub-31	0.74	0.77	0.75	0.73	0.73	0.67
sub-22	0.68	0.67	0.81	0.65	0.74	0.76
sub-03	0.80	0.77	0.82	0.87	0.80	0.80
sub-30	0.87	0.81	0.77	0.81	0.82	0.81
sub-25	0.75	0.78	0.71	0.71	0.65	0.65
average	0.81	0.80	0.79	0.79	0.79	0.78

Table 3.2: Results from testing the channel configurations from channel selection with genetic algorithm in electrode space.

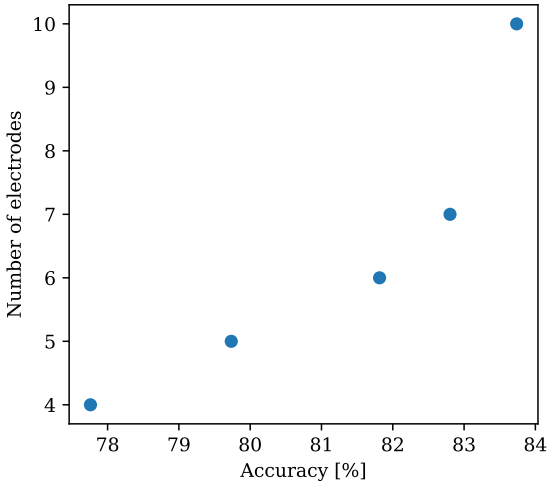
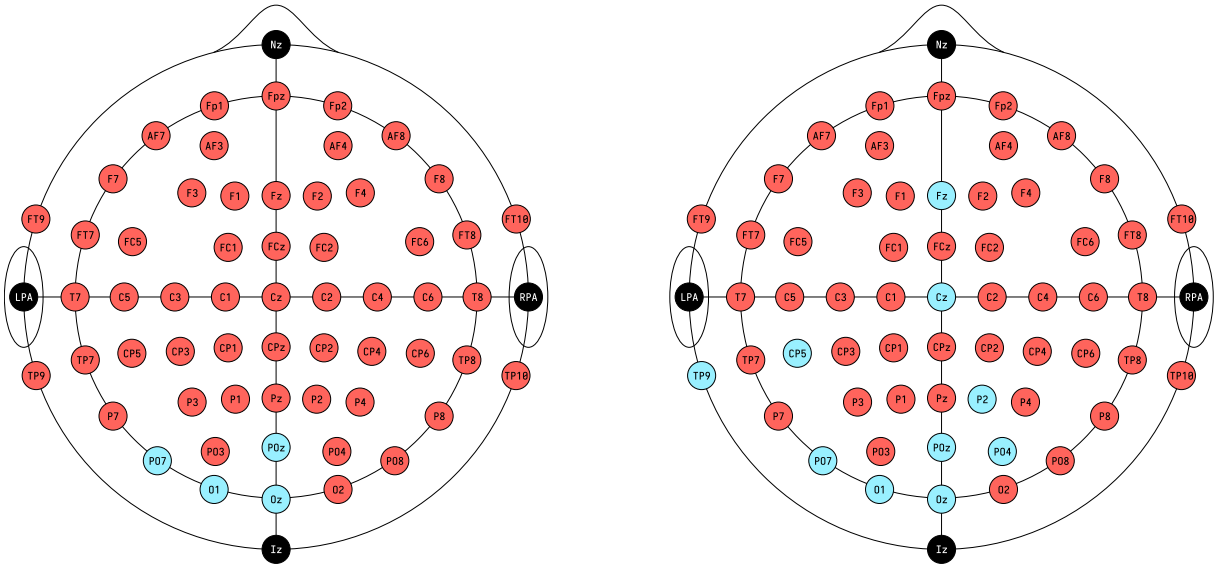


Figure 3.12: The solutions of channel selection in electrode space. Each point indicates the number of electrodes and achieved accuracy of a pareto-optimal solution found by the GA.



(a) The 4-electrode solution configuration.

(b) The 10-electrode solution configuration.

Figure 3.13: Two electrode configurations found by GA channel selection.

3.3.3 Source Selection

Approach

As there are more than 8000 sources and only 58 electrodes, the channel selection problem becomes considerably larger when selecting sources rather than electrodes. The effects on complexity when having more channels are twofold. Firstly, the amount of data increases, thus training and testing of the DNN require more computation. Secondly, the solution space is larger; there are more different configurations to explore, thus the GA will likely need larger populations and more generations than for the electrode problem to converge on good configurations. To limit the number of channels in the source selection problem, a routine was established to select a subset of the sources, based on the

electrodes returned by the electrode selection. The routine is as follows:

1. Project the digitized positions of the selected electrodes onto the MRI scalp surface. Thus obtaining a vector normal to the scalp at the electrode position.
2. For each selected electrode, define a sphere with a center on the line defined by the normals obtained in the previous step. These spheres should be tangent to their respective projected electrode positions, with their centers falling within the head volume.
3. Select all sources that reside within the volume defined by the union of the spheres defined in the previous step. The selected sources are the sources to be used in the GA channel selection.

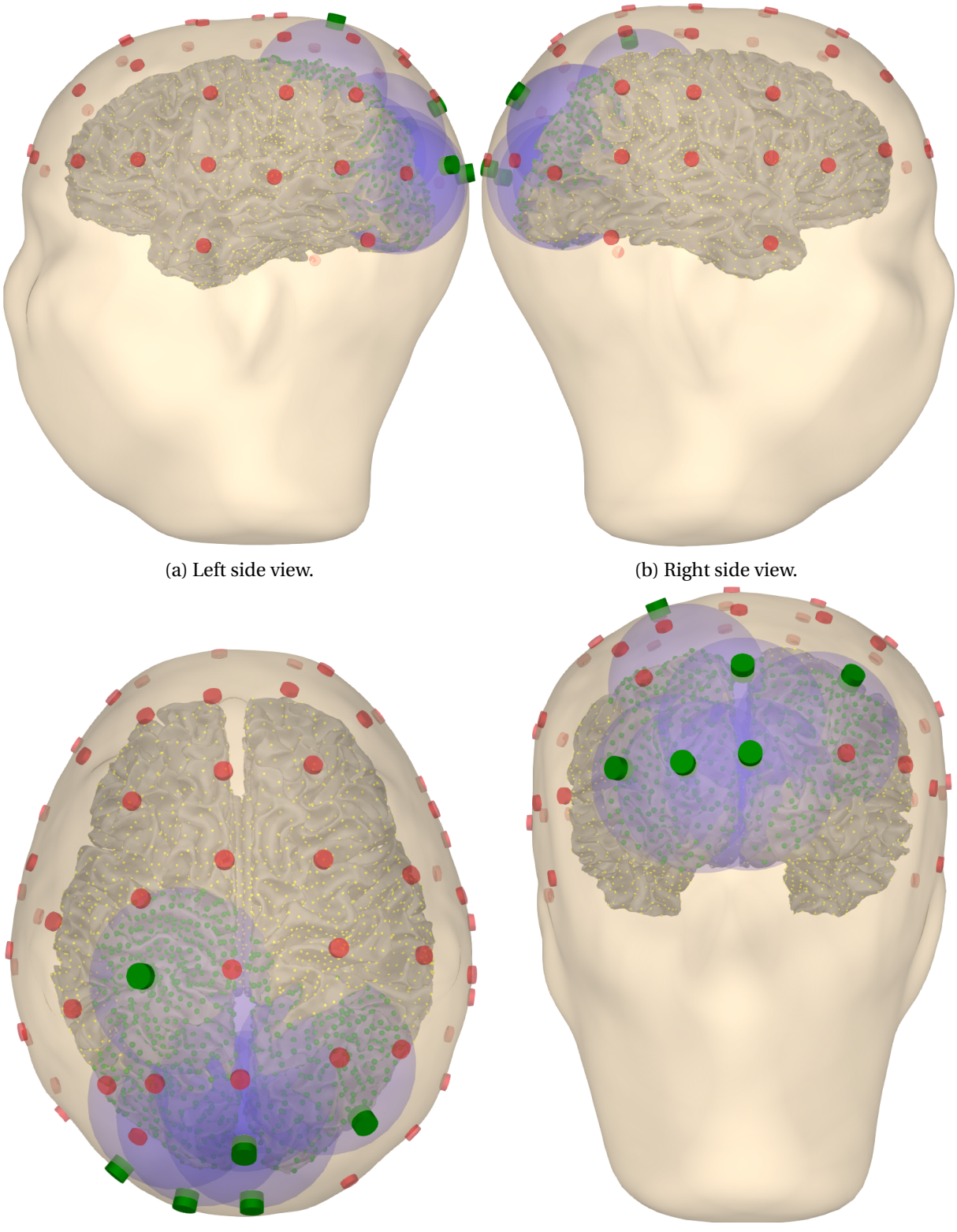
Figure 3.14 illustrates the idea behind this routine.

This method was designed based on the reasoning that the electrodes providing the best accuracy, are likely to be placed close to the areas of the brain where the activity of color responses occur. In this way, the electrodes are used to define regions of interest (ROIs) in the brain. Limiting the set of sources available in the GA to these ROIs might keep most of the sources that make up an optimal set, while significantly reducing the amount of computation needed in the GA. How many sources this routine selects will depend on the number of selected electrodes and the radius of the sphere. The 6-electrode configuration presented in subsection 3.3.2 was used to select sources. After this selection had been made, the radius of the spheres was tuned such that a reasonable number of sources were selected. This yielded 474 sources.

Since the source selection, similar to the electrode selection, is intended to yield a set of sources that are optimal not just for one individual, but that is suitable for any subject, the data used in the fitness function should come from the set of validation subjects. However, the source space used to produce the dataset described in section 3.2 is tailored to each subject, based on their individual MRI data. A general source space needs to be established, such that a configuration of selected sources will be applicable to the reconstructed sources for any subject. This was achieved by using the same forward model for all subjects when doing source reconstruction. This common forward model was computed using the FsAverage head. FsAverage is a template head model developed using MRI data from a set of subjects and is meant to serve as a template for the average human head. It is part of the FreeSurfer software [25].

Results

After running the GA for 100 generations, with a population size of 20, the four configurations shown in Figure 3.15 were obtained. The pareto front estimated by these solutions is very similar in nature to the one obtained in electrode space (Figure 3.12). As for the electrode space results, new tests were run for each configuration to evaluate their performance on each individual validation subject. The results of those tests are provided in Table 3.3. Just as for the results in electrode space, the average accuracy for all configurations is slightly lower, than the accuracies returned by the GA. This trend will be discussed in Chapter 5. Regardless of this discrepancy, all the configurations yield better results than using all 474 sources. Table 3.3 also shows that it is very subject-dependent whether the source selection is beneficial or not. For instance, subject 26 has a substantial increase in performance with all configurations, as opposed to using all sources. Meanwhile, both subject 31 and subject 25 perform



(a) Left side view.

(b) Right side view.

(c) Top view.

(d) Back view.

Figure 3.14: The source selection based on selected electrodes. The green points on the scalp indicate the selected electrodes. The blue spheres constitute the volume within which sources are selected. The green points on the white matter surface indicate which sources would be selected with this configuration.

Subject	All sources	192 sources	161 sources	151 sources	149 sources
sub-18	0.52	0.48	0.49	0.48	0.55
sub-26	0.68	0.87	0.84	0.77	0.81
sub-07	0.48	0.50	0.54	0.54	0.49
sub-31	0.48	0.37	0.35	0.48	0.40
sub-22	0.38	0.53	0.46	0.50	0.56
sub-03	0.55	0.70	0.71	0.72	0.69
sub-30	0.60	0.73	0.80	0.78	0.72
sub-25	0.43	0.40	0.44	0.54	0.40
average	0.52	0.57	0.58	0.60	0.58

Table 3.3: Results from testing the channel configurations from channel selection with genetic algorithm in source space.

worse or similar with selected sources. To investigate this difference further, these three subjects were used in a new test described in the following section.

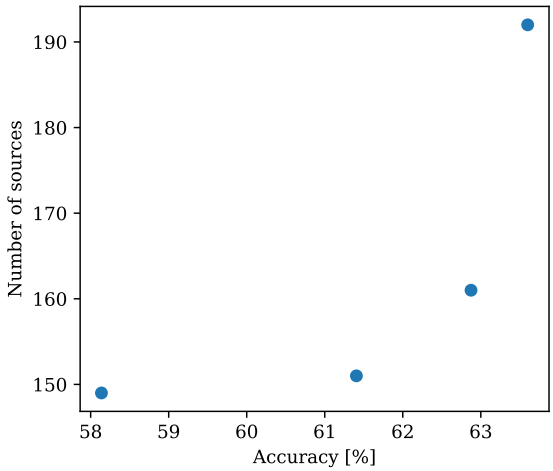


Figure 3.15: The solutions of channel selection in source space. Each point indicates the number of sources and the achieved accuracy of a pareto optimal solution found by the GA.

Subject-specific Selection in Source Space

Since the channel selection in source space required the use of a template head, it is natural to assume that the approach would be better suited for subjects with heads resembling the template. This could explain the substantial difference in performance seen in Table 3.3. To test this, channel selection on an individual basis was performed for three subjects. This was done using the same method as previously, except the forward model used in the SR was based on MRI from the individuals. Since the white-matter surfaces of these individuals are different from that of the template head, sources will not have the same positions as those depicted in Figure 3.14. Therefore, when using the sphere volume selection method on the individual source space, the number of sources selected will not be exactly the same as for the template. It was ensured that the number of sources for the three subjects did not deviate too much from the number of sources selected in the template head. The results of these tests are presented in Table 3.4. It should be noted that the minimal preprocessing

Sub-25		Sub-26		Sub-31	
Sources	**Accuracy**	**Sources**	**Accuracy**	**Sources**	**Accuracy**
149	0.64	149	0.90	149	0.81
151	0.64	151	0.86	151	0.71
161	0.67	161	0.86	161	0.83
192	0.69	192	0.86	192	0.76
all	0.60	all	0.83	all	0.86
144 (subject-specific)	0.62	133 (subject-specific)	0.88	133 (subject-specific)	0.83
145 (subject-specific)	0.62	134 (subject-specific)	0.88	136 (subject-specific)	0.79
147 (subject-specific)	62	137 (subject-specific)	0.88	166 (subject-specific)	0.80
154 (subject-specific)	0.67	159 (subject-specific)	0.88	all (subject-specific)	0.86
all (subject-specific)	0.62	all (subject-specific)	0.88		

Table 3.4: Results from subject-specific source selection. The results from channel selection using individual forward models are indicated with subject-specific. The other results are from the channel selection with the template forward model.

pipeline was used to obtain these results, while the full preprocessing was used to obtain the previous results. As the results suggest, using minimal preprocessing improves the classification accuracy for all the subjects. However, there is still a noticeable difference in accuracy between the three subjects, similar to what was seen in Table 3.3. When comparing the subject-specific results against the results using the template head, there does not seem to be a significant difference. There is also little to no increase in accuracy with the selected sources in the subject-specific results, compared to using all subject-specific sources. In general, there is very little variation in performance within each subject. This was not expected and will be further discussed in Chapter 5.

Distribution of Selected Sources

The position of the sources selected by the GA is an important part of the result. How the selected sources are distributed in the brain could reveal information about the way our brain interprets color. The source distribution could also serve as a guide for optimizing electrode placement for decoding colors. Figure 3.16 illustrates the distribution of sources in the 192-source configuration in the different ROIs defined by the parcellation from [21].

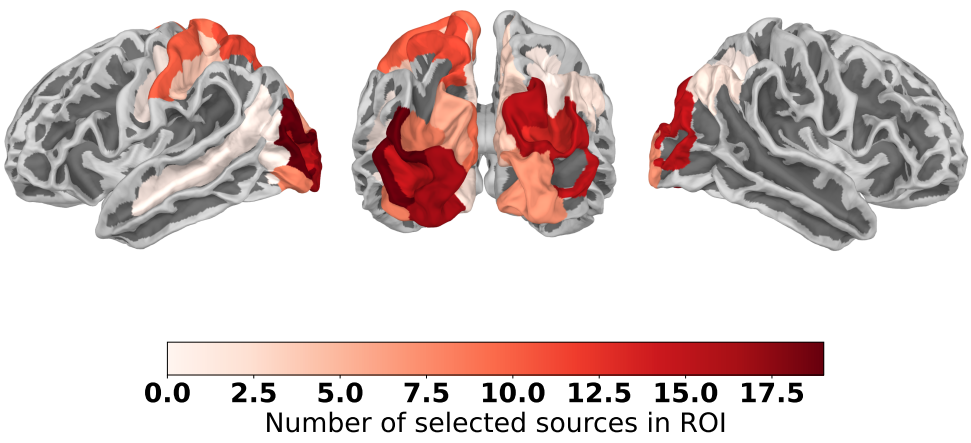


Figure 3.16: The distribution of selected sources in each ROI, for the 192-source configuration.

Region of Interest	Number of selected sources
Inferior occipital gyrus and sulcus (lh)	8 (of 20)
Paracentral lobule and sulcus (lh)	9 (of 14)
Cuneus (lh)	9 (of 26)
Cuneus (rh)	4 (of 15)
Lingual gyrus(lh)	2 (of 2)
Middle occipital gyrus (lh)	19 (of 42)
Middle occipital gyrus (rh)	15 (of 29)
Superior occipital gyrus (lh)	6 (of 28)
Superior occipital gyrus (rh)	16 (of 41)
Angular gyrus (rh)	2 (of 7)
Superior parietal lobule (lh)	11 (of 25)
Postcentral gyrus (lh)	10 (of 28)
Precuneus (lh)	2 (of 10)
Precuneus (rh)	3 (of 4)
Occipital pole (lh)	17 (of 39)
Occipital pole (rh)	7 (of 16)
Calcarine sulcus (lh)	4 (of 9)
Central sulcus (lh)	1 (of 3)
Marginal branch of the cingulate sulcus (lh)	0 (of 1)
Posterior transverse collateral sulcus (lh)	1 (of 3)
Intraparietal sulcus and transverse parietal sulci (rh)	1 (of 3)
Middle occipital sulcus and lunatus sulcus (lh)	15 (of 24)
Superior occipital sulcus and transverse occipital sulcus (lh)	8 (of 20)
Superior occipital sulcus and transverse occipital sulcus (rh)	13 (of 33)
Anterior occipital sulcus and preoccipital notch (lh)	1 (of 7)
Parieto-occipital sulcus (rh)	3 (of 10)
Postcentral sulcus (lh)	4 (of 13)
Superior temporal sulcus (lh)	1 (of 2)
Total	192 (of 474)

Table 3.5: The distribution of sources in ROIs, for the 192-source selection. The ROIs are defined by the parcellation introduced in [21].

Table 3.5 presents the exact distribution of the 192-source configuration. From the figure, it can be observed that especially many sources are located in or close to the primary visual cortex (V1). The figure can be somewhat misleading, as the number of sources available for selection varies significantly from region to region. Also, due to the restriction made by the sphere selection, the channel selection could only select sources in certain regions in the back of the head. To overcome these issues, a source selection was done using the full source space. To limit the complexity of the problem, a lower resolution source space was created, with roughly 2000 sources. Source selection with GA was performed on all these sources, for 100 generations with a population size of 20. Figure 3.17 shows the distribution of the sources when the GA was free to select among all the 2000 sources throughout the whole brain. This distribution is fairly even, and most ROIs have around 40% of their sources selected. This suggests that most areas of the brain are relevant for decoding the color responses. The electrode selection indicates the opposite, as all electrode selections preferred electrodes close to the occipital lobe. It could be that the number of generations and population size was too small for the problem, as choosing a configuration of 2000 sources is substantially larger than choosing among 58 electrodes.

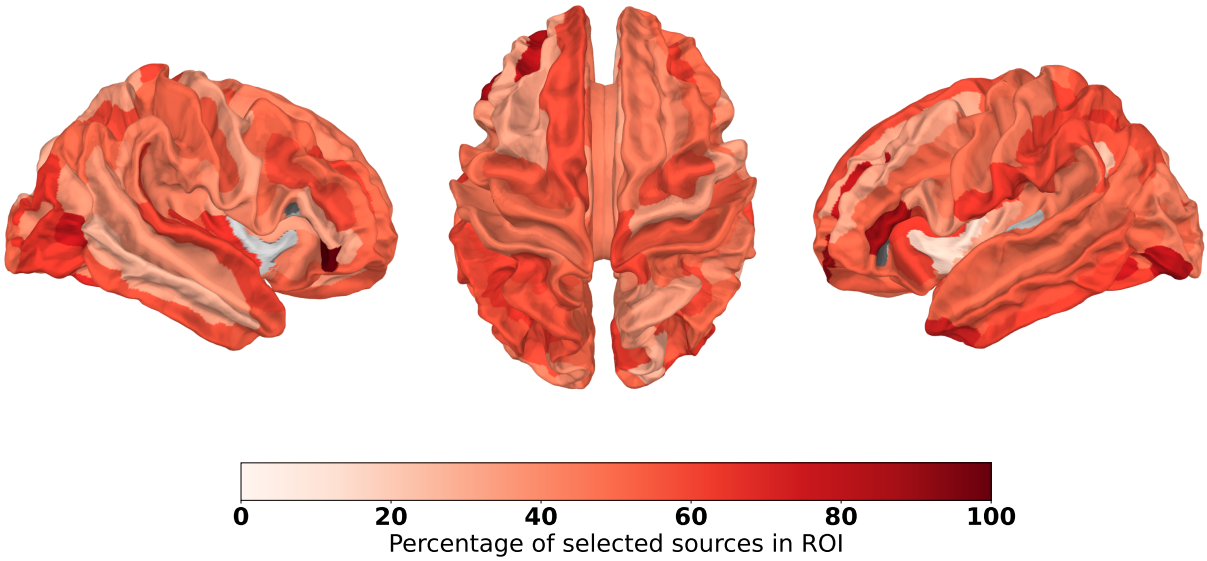


Figure 3.17: The distribution of selected sources in each ROI, from a selection based on a complete source space.

3.4 Classification

In the preliminary work, three DNN architectures were implemented and tested for decoding RGB responses. Of these, the deepConvNet architecture, originally proposed in [62], had the best performance. For this reason, the deepConvNet was used as the classifier in the fitness function for the channel selection described in section 3.3. However, in the preliminary work, it was also pointed out that deepConvNet has a focus on temporal features, as it is a CNN performing convolutions only over the temporal dimension. As described in subsection 2.1.2, the spatial position of neural activity may also be descriptive for color responses, so facilitating learning spatial features could be beneficial. Since the source-reconstructed data has a high spatial resolution it should be well suited for finding spatial features. Due to the nature of the source-reconstructed data, it is not straightforward to apply spatial convolutions. Some additional transformations, which are the topic of subsection 3.4.1, are required for facilitating spatial convolutions. In this work, three models using spatial convolutions were explored: 3M3DCNN [50], 2DCNN [24], and ST3DCNN a novel architecture developed as part of this work.

3.4.1 Structured Spatial Source Representation

CNNs take advantage of information embedded in the structure of the input data. For instance, if the input is an image, the only explicit data is the color values of each pixel. However, the position of the pixel in the 2D grid contributes spatial information. In a picture, pixels close to each other are often part of the same feature, while pixels far away from each other likely have no correlation. CNNs take advantage of this information, as the kernel only produces features from pixels within a certain distance from each other. The data produced by SR is a matrix with shape $(n_{\text{sources}} \times n_{\text{samples}})$. The second dimension, representing time, is structured. So data which is close to each other in this dimension of the matrix were recorded close in time. Thus, convolution can be applied in this dimension. The first dimension on the other hand, is not structured. It is a random ordering of all the sources. Applying a convolution in this dimension would not be meaningful, as there is no reason to assume that a source is more correlated to sources close in the matrix than to sources far away. Figure 3.18 demonstrates the difference between the structured and unstructured dimensions. This section will outline a method for structuring this dimension, and introduce two CNNs specialized for classifying data with this new structure.

Voxelization and Pixelization

The position of each source in 3D space is known. To structure the source data based on these positions, we can voxelize the data. Voxelization is the process of converting any sort of 3D data into a 3D array of voxels (volume pixels) [36]. Source space data is represented as a point cloud, as illustrated in Figure 3.19. To voxelize this data, one can define a box including all the points, and discretize this volume into cubes, these cubes are the voxels. Then each voxel will take the average value of all points residing within it. The algorithm used in this work for creating the map from point cloud to voxels is described in Algorithm 1. It takes the 3D positions of all sources and returns their respective indices in a voxelized 3D array. Figure 3.19 shows an example of a source space point cloud, and the voxelgrid obtained after voxelizing it.

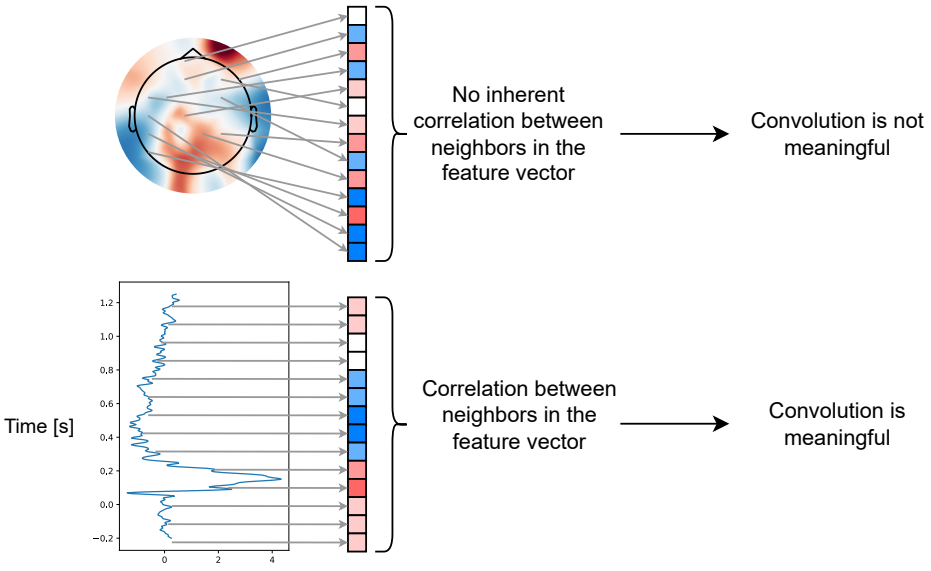


Figure 3.18: An illustration of the structure in the temporal dimension, and the lack of such a structure in the source dimension. This figure is adapted from the preliminary work [27].

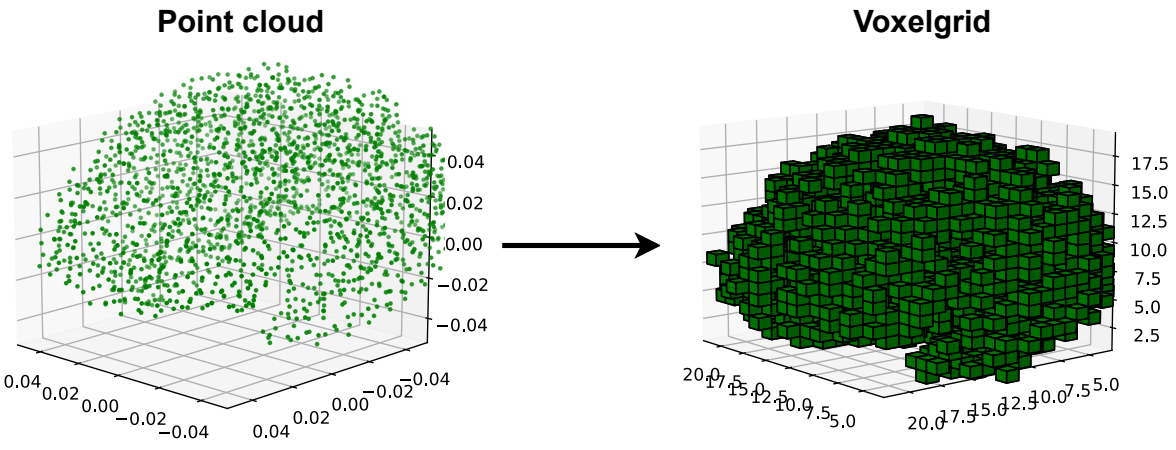
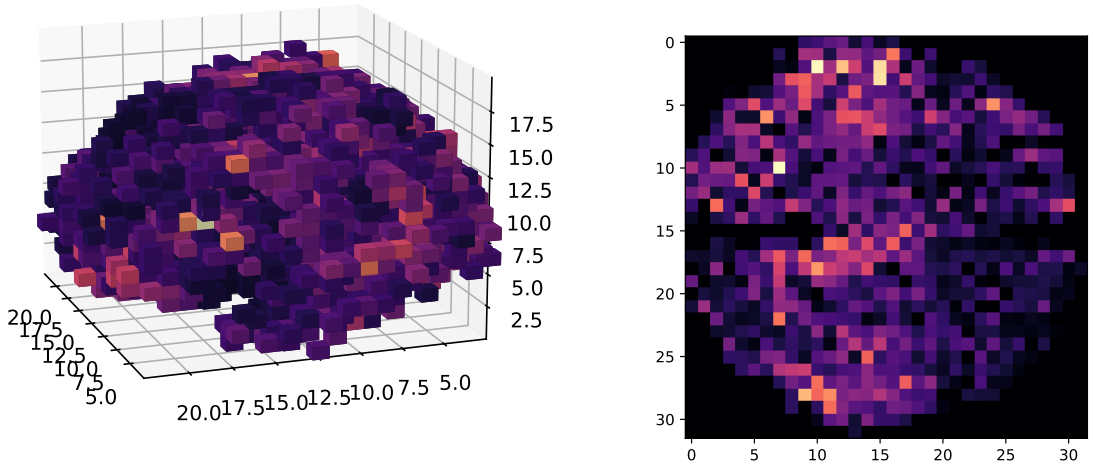


Figure 3.19: Voxelization of point cloud for source space data. In this illustration voxels that do not contain any points are not drawn, however, the voxelization returns a 3D matrix with all voxels, not just those containing points.

An important consideration when doing the voxelization is the size of the voxels, as this effectively sets the resolution of the voxel representation. If the voxels are big, many points will end up inside each voxel, and the resulting spatial resolution will be lower. On the other hand, if the voxels are small, the spatial resolution will be higher, but the data size will also increase significantly. Thus selecting the voxel size becomes a compromise between spatial resolution and data size. As shown in Algorithm 1, the voxel size can be set indirectly by defining the size of each dimension in the 3D voxel grid. One of the DNNs employed in this project, described in subsection 3.4.3, requires source space data with a 2D spatial representation. Such a representation was obtained by performing a projection from above on the voxel grid representation. The result is then a 2D pixel grid (an image). A visualization of what these data representations will look like is provided in Figure 3.20. The temporal dimension is still preserved, so the data from one epoch would have the shape (width × height × depth × n_{samples}) or (width × height × n_{samples}) for voxels and pixels respectively.

Algorithm 1 Voxelization

function VOXELIZEPOINTCLOUD($points, width, height, depth$)
  $max_x \leftarrow \max(points.x), min_x \leftarrow \min(points.x)$
  $max_y \leftarrow \max(points.y), min_y \leftarrow \min(points.y)$
  $max_z \leftarrow \max(points.z), min_z \leftarrow \min(points.z)$
  for all p in $points$ **do**
    $p.x \leftarrow (p.x - min_x) * (width/max_x)$
    $p.y \leftarrow (p.y - min_y) * (height/max_y)$
    $p.z \leftarrow (p.z - min_z) * (depth/max_z)$
  end for
  $points \leftarrow$ FLOOR($points$)
  return $points$
end function



(a) 3D Voxel representation.

(b) 2D Pixel representation.

Figure 3.20: Voxel and pixel representations of source space data. In this example, the pixel representation is a projection from above of the voxel data.

3.4.2 3M3DCNN

3M3DCNN, as named by its developers, is a DNN developed for decoding MI from source-reconstructed EEG signals [50]. The architecture was designed to make full use of the high resolution of the spatial dimensions of source space data. 3M3DCNN takes as input four-dimensional (4D) data, with three spatial dimensions and one temporal dimension. Time of interest (TOI) selection was used to select three samples in time to include in the temporal dimension. The spatial dimensions were defined by a voxelization, with a reasonable size considering the computer memory. The resulting input shape was (width×height×depth×TOIs)=($30 \times 38 \times 28 \times 3$). In [50], this data representation is referred to as 4D dipole feature matrix (4DDFM). Results from tests on public datasets showed that 3M3DCNN could outperform proposed classifiers operating on raw EEG data.

Architecture

Three modules make up the 3M3DCNN architecture. The first two perform 3D convolutions on the spatial dimensions, and the last has dense layers. The architecture with all its layers is described in Table 3.6.

Layer	#Filters/Units	Kernel size	#Parameters	Output	Activation	Options
Input				(30,38,28,T (3))		
Conv3D	32	(3,3,3)	2624	(30,38,28,32)		padding = same, stride = (1,1,1) (3,3,3)
BatchNormalization			128 (120)	(30,38,28,32)		
Activation				(30,38,28,32)	ReLU	
Conv3D	32	(3,3,3)	27680	(30,38,28,32)		padding = same, stride = (1,1,1) (3,3,3)
Activation				(30,38,28,32)	Softmax	
MaxPooling3D		(3,3,3)		(10,13,10,32)		padding = same, stride = (3,3,3)
Dropout				(10,13,10,32)		p = 0.5 (not specified)
Conv3D	64	(3,3,3)	55360	(10,13,10,64)		padding = same, stride = (1,1,1) (3,3,3)
BatchNormalization			128 (120)	(10,13,10,64)		
Activation				(10,13,10,64)	ReLU	
Conv3D	64	(3,3,3)	110656	(10,13,10,64)		padding = same, stride = (1,1,1) (3,3,3)
Activation				(10,13,10,64)	Softmax	
MaxPooling3D		(3,3,3)		(4,5,4,64)		padding = same, stride = (3,3,3)
Dropout				(4,5,4,64)		p = 0.5 (not specified)
Flatten				5120		
Dense	512		2621952	512	Sigmoid	
BatchNormalization			2048	512		
Dropout				512		p = 0.5 (not specified)
Dense	3 (4)		1539 (2052)	3 (4)	SoftMax	

Table 3.6: The 3M3DCNN architecture. There are some deviations from the architecture proposed in [50] and the architecture implemented in this project. For these discrepancies, the original values are provided in red.

Implementation

There is no code available from the original implementation of 3M3DCNN, so not all details of the implementation are known. Keras was used to implement the 3M3DCNN architecture in this project. There are some discrepancies between the 3M3DCNN architecture defined in [50] and the adaption of 3M3DCNN used in this work. These discrepancies are marked in red in Table 3.6:

- The strides of the convolution layers in the original architecture are (3,3,3), while they are (1,1,1) in this paper. In the Keras framework, using a stride of (3,3,3) will reduce the output dimensions. Therefore, the stride was changed to (1,1,1) to ensure the data dimensionality follows that of the original paper.

- In the original paper, the rate of units to drop in dropout layers is not specified. Hence, 0.5 was selected as this has been used in other successful DNNs for decoding EEG [62, 46].
- The number of parameters of the batch normalization layers differs. The original paper has 120 parameters in these layers, while they have 128 parameters in the implementation of this project. This might be due to a difference in the framework used in [50] and Keras.
- The last dense layer is different. Since this layer serves as the output of the classifier, it has to be configured based on the paradigm. In [50], there were four classes, in this project there are three. Thus, the output layer has three instead of four units. As a result, it also has a different number of parameters.
- The temporal dimension of the input had a size of 3 in the original paper. It was left as a parameter in this project, such that different TOIs could be explored.

In [50] only three samples are used as TOI. Different selections of three samples were explored in this work, but with poor results. In the end, a TOI selection was performed, which is described in subsection 3.4.5. Based on this selection all samples from 0-400ms were selected as the TOI.

3.4.3 2DCNN

A DNN for classifying MI tasks, based on a 2D representation of source space data, was proposed in [24]. This architecture is referred to as 2DCNN in this work. Unlike the 3M3DCNN architecture described in subsection 3.4.2, the input to 2DCNN has only two spatial dimensions. The 2D source space representation is achieved by a projection from above, as described in section 3.4.1. Although the 2DCNN classifier uses deep learning, [24] proposes a feature extraction based on continuous wavelet transform (CWT), to be performed on the data before it is sent to the model. The feature extraction produces three features: the average power of three frequency bands. These three bands were selected for their known correlation to MI tasks [24].

Architecture

The 2DCNN architecture uses seven layers of 2D convolution. After the convolution, a small fully-connected network is used to produce the output. Between the convolutional layers, batch normalization and max pooling is used. The detailed description of all the layers is provided in Table 3.7

Implementation

No code was provided in the paper proposing the 2DCNN architecture. In this project, Keras was used for the implementation. There are some known discrepancies between the 2DCNN architecture defined in [50] and the adaption of 2DCNN used in this work. These discrepancies are marked in red in Table 3.7:

- The reported number of parameters in the convolutional layers differ. However, it is likely that the original paper only reported the number of parameters per input. If the number of inputs is taken into account, the number of parameters in the original paper would match those reported in this work.

Layer	#Filters/Units	Kernel size	#Parameters	Output	Activation	Options
Input				(64,64,3)		
Conv2D	32	(3,3)	896 (288)	(64,64,32)		padding = same, stride = (1,1)
Activation				(64,64,32)	Leaky ReLU	
BatchNormalization			128	(64,64,32)		
Conv2D	32	(3,3)	9248 (288)	(64,64,32)		padding = same, stride = (1,1)
Activation				(64,64,32)	Leaky ReLU	
BatchNormalization			128	(64,64,32)		
MaxPooling2D		(2,2)		(64,64,32)		padding = valid, stride = (2,2)
Conv2D	32	(3,3)	9248 (288)	(32,32,32)		padding = same, stride = (1,1)
Activation				(32,32,32)	Leaky ReLU	
BatchNormalization			128	(32,32,32)		
Conv2D	32	(3,3)	9248 (288)	(32,32,32)		padding = same, stride = (1,1)
Activation				(32,32,32)	Leaky ReLU	
BatchNormalization			128	(32,32,32)		
MaxPooling2D		(2,2)		(16,16,32)		padding = valid, stride = (2,2)
Conv2D	64	(3,3)	18496 (576)	(16,16,64)		padding = same, stride = (1,1)
Activation				(16,16,64)	Leaky ReLU	
BatchNormalization			256	(16,16,64)		
Conv2D	64	(3,3)	36928 (576)	(16,16,64)		padding = same, stride = (1,1)
Activation				(16,16,64)	Leaky ReLU	
BatchNormalization			256	(16,16,64)		
MaxPooling2D		(2,2)		(8,8,64)		padding = valid, stride = (2,2)
Conv2D	128	(3,3)	73856 (1152)	(8,8,128)		padding = same, stride = (1,1)
Activation				(8,8,128)	Leaky ReLU	
BatchNormalization			512	(8,8,128)		
MaxPooling2D		(2,2)		(4,4,128)		padding = valid, stride = (2,2)
Flatten				2048		
Dense	512		1049088	512	None	
Dense	3 (2)		15391024	3 (2)	SoftMax	

Table 3.7: The 2DCNN architecture. There are some deviations from the architecture proposed in [24] and the architecture implemented in this project. For these discrepancies, the original values are provided in red.

- The values of the last dense layer differ from the original implementation. This is due to the different number of classes to classify.

As mentioned, the original paper used feature extraction. Three features were extracted from the time series of each pixel in the 2D source representation. Transforming the input data from $(64 \times 64 \times n_{\text{times}})$ to $(64 \times 64 \times 3)$. The three features extracted are the spectral power averaged over a frequency band and over time. The three frequency bands were the α (8-14Hz), β (14-30Hz), and θ (4-8Hz) bands [24]. There exist several definitions of these bands and it is not clear from the paper exactly what frequencies define the bands, so in this work, the definitions from [2] were used. Figure 3.21 illustrates the feature extraction. The three first figures are the maps for the individual frequency bands. The last figure is the combined map and illustrates the complete data sent as input to the 2DCNN architecture. As the figures show, the projection from 3D to 2D was done from behind, not from above. A projection from behind gives better coverage of the data originating from the occipital lobe, and should therefore be beneficial for the task of decoding visual stimuli. The feature extraction was performed using MNE-Python.

3.4.4 ST3DCNN

Three architectures proposed by previous papers have been explored in this work: deepConvNet, 3M3DCNN, and 2DCNN. DeepConvNet relies heavily on temporal convolutions, while the two others

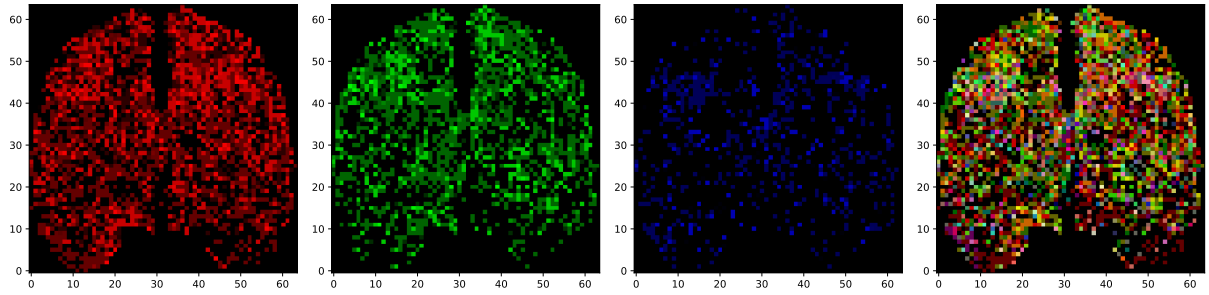


Figure 3.21: Feature extraction for 2DCNN. The three first figures illustrate the average powers of the θ, α, and β bands respectively. The last figure is a composition of these three features into an RGB map.

use spatial convolutions. However, none of these combines both spatial and temporal convolutions. This work proposes an architecture combining both spatial and temporal convolutions, inspired by the three other architectures. This novel architecture is referred to as ST3DCNN (spatiotemporal 3DCNN). The architecture takes as input a 3D representation of source space data, two spatial dimensions, and one temporal. This input is similar in nature to the input used in 2DCNN, except there is no feature extraction.

Architecture

The architecture is very similar to that of the deepConvNet proposed in [62], consisting of five sections. The first four have the same layer structure: Convolution, batch normalization, activation, maxpooling, and dropout. This is the same structure used in deepConvNet. ST3DCNN employs 3D convolutions, with two spatial dimensions and one temporal. The three first segments perform convolutions on all three dimensions, while the final convolution is only temporal. The sizes of the kernels in the convolution- and maxpooling layers are modeled after deepConvNet and 2DCNN. The strides have been designed to ensure suitable dimensions of the data throughout the DNN. Finally, there is a dense layer with softmax activation, to produce the output. The details of the ST3DCNN architecture are provided in Table 3.8.

3.4.5 Time of Interest Selection

As demonstrated in previous research, using data from only certain time intervals can improve color classification accuracy [61, 50]. Similar to the spatial ROI selection, it can be advantageous to seek out certain TOIs. The advantage of using a set of TOIs, as opposed to the entire time series, is twofold. Firstly, the accuracy can be increased [50]. Secondly, the data size is reduced, meaning less memory and computational resources are needed. To seek out TOIs, the data of the VAL8 subjects was divided into windows of 100 ms. A classifier was trained and tested for each of the subjects on each of the windows. The average accuracy of all subjects for one window served as an indication of the relevance of that window. These results are shown in Figure 3.22.

Layer	#Filters/Units	Kernel size	#Parameters	Output	Activation	Options
Input				(64,64,T,1)		
Conv3D	25	(3,3,8)	1825	(21,21,T_1,25)		padding = valid, stride = (3,3,1)
BatchNormalization			100	(21,21,T_1,25)		
Activation				(21,21,T_1,25)	Leaky ReLU	
MaxPooling3D		(1,1,2)		(21,21,T_2,32)		padding = valid, stride = (1,1,2)
Dropout				(21,21,T_2,32)		p = 0.5
Conv3D	50	(3,3,8)	90050	(19,19,T_3,50)		padding = valid, stride = (1,1,1)
BatchNormalization			200	(19,19,T_3,50)		
Activation				(19,19,T_3,50)	Leaky ReLU	
MaxPooling3D		(3,3,1)		(6,6,T_3,50)		padding = valid, stride = (3,3,1)
Dropout				(6,6,T_3,50)		p = 0.5
Conv3D	100	(3,3,8)	360100	(4,4,T_4,100)		padding = valid, stride = (1,1,1)
BatchNormalization			400	(4,4,T_4,100)		
Activation				(4,4,T_4,100)	Leaky ReLU	
MaxPooling3D		(3,3,1)		(1,1,T_4,100)		padding = valid, stride = (3,3,1)
Dropout				(1,1,T_4,100)		p = 0.5
Conv3D	200	(1,1,8)	160200	(1,1,T_5,200)		padding = valid, stride = (1,1,2)
BatchNormalization			800	(1,1,T_5,200)		
Activation				(1,1,T_5,200)	Leaky ReLU	
Dropout				(1,1,T_5,200)		p = 0.5
Flatten				1600		
Dense	3		4803	3	SoftMax	

Table 3.8: The ST3DCNN architecture. T is the size of the temporal dimension in the input. $T_1 = T-7$, $T_2 = T//2$, $T_3 = T_2-7$, $T_4 = T_3-7$, $T_5 = (T_4-7)//2$

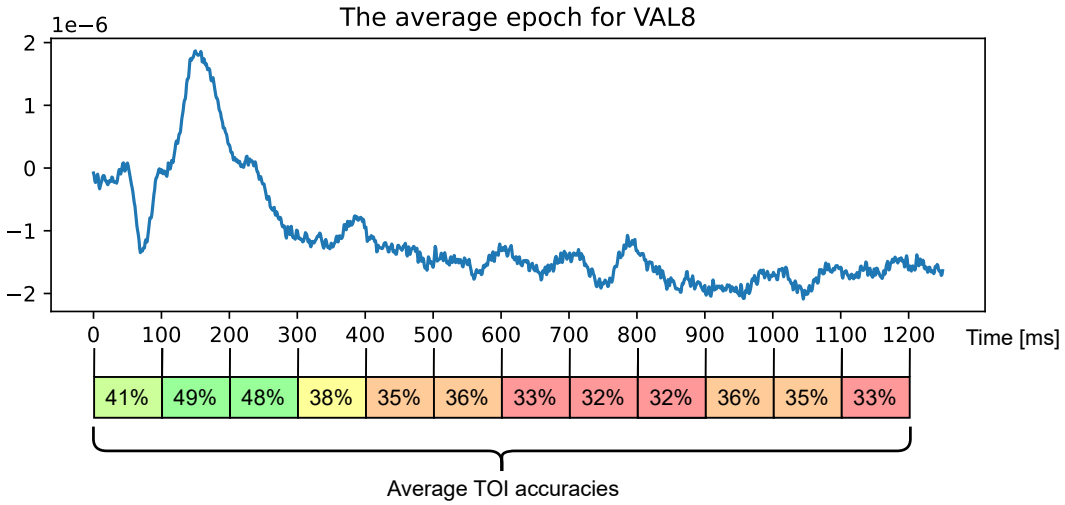


Figure 3.22: TOI selection results. The percentages at the bottom of the figure indicate the average accuracy achieved when using only that TOI for classification on the VAL8 subjects. The plot shows the average epoch of the VAL8 subjects.

Since this is a three-class classification paradigm, the chance level is at 33%, thus the results indicate that the intervals after 400ms do not contribute much information relevant to the classification. These results indicate that the most important TOI is from 100ms-200ms, which coincides well with the results of [61], which reported a peak decoding accuracy at 115ms after stimuli onset. Following these results it seems that 0-400ms would be a suitable TOI for this paradigm. For the tests performed on the three new architectures explored in this work, this interval was used. For the deepConvNet architecture 0-800ms was used.

3.4.6 Data Augmentation

How well an ML classifier generalizes is dependent on the amount of training data available [30]. However, data collection is often expensive and time-consuming. This is certainly the case for acquiring EEG data such as in this project. Data augmentation is a method whereby fake data is added to the training set of an ML algorithm, to increase the generalization of the algorithm [30]. That is, certain transformations can be applied to an original example of a class, to create a new example of the same class. Figure 3.23 demonstrates this principle with a simple example. If we are classifying shapes into squares and circles, transformations in size and color can be applied to original examples. Such transformations would yield new examples that belong to the same class as the original examples. Naturally, the task of finding such transformations is more challenging in the case of EEG

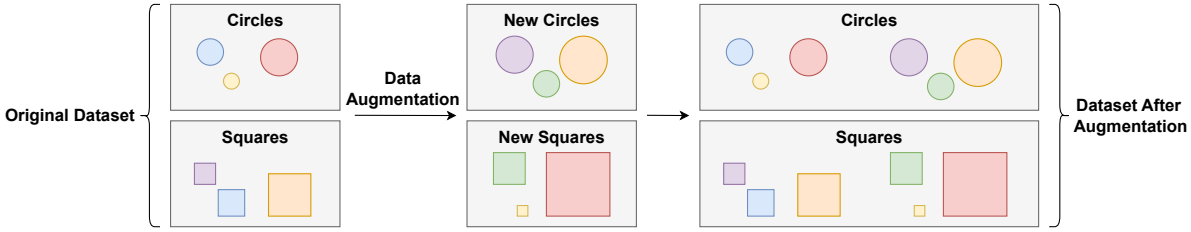


Figure 3.23: An example of data augmentation. By transforming the color and size of the original examples, a new set of examples can be created, which still belong to the classes from which they originated. Thus, the dataset has been doubled in size.

decoding, since it is not known exactly what properties separate the examples of the different classes. Still, data augmentation has been successfully applied to previous EEG decoding tasks [24]. Fang et al. explored four different methods for augmenting EEG data for a motor imagery classifier. Two of these, noise addition and sliding window, yielded an increase in classification accuracy. These two methods were also explored in this project. The basic principle and implementation of these are covered in the following paragraphs. The results obtained when exploring these methods are presented in subsection 4.1.3.

Noise Injection

A simple way to augment training data is by adding noise to it. This method has been successfully employed in several other EEG decoding tasks [24, 69, 38]. Since additive white Gaussian noise (AWGN) will appear naturally in EEG signals [29], creating augmented examples with new AWGN, should preserve the properties making the example distinct to its class, while making the classifier more robust. When generating the AWGN to be injected into the data, the mean and variance of this noise must be chosen. In this work, the mean was set to zero, and the variance was chosen based on the variance of the training data itself. Several values were tested with the validation subjects. Finally, a variance of 10% of the data variance was selected to be used in the test presented in subsection 4.1.3. This means the noise injection doubled the number of epochs available for training, as noise will be added to each original epoch to create an augmented epoch.

Sliding Window

Several crops of the same trial can be made with a sliding window technique, to increase the amount of data in the training set. For EEG decoding this has been applied in a number of previous studies [62, 50, 24, 71]. The implementations of this method vary somewhat, but the basic principle is the same. Suppose the data has a temporal dimension of size T. By creating a window of length $T' < T$, a set of $T - T' + 1$ different crops can be created from one single trial, by sliding the window across the temporal dimension. For instance, if $T = 10$ and $T' = 8$ three different crops could be created: $\text{crop}_1 = \{t_0, t_1 \ldots, t_7\}$, $\text{crop}_2 = \{t_1, t_2 \ldots, t_8\}$, and $\text{crop}_3 = \{t_2, t_3 \ldots, t_9\}$. For this case, the number of examples in the training set has been tripled. Here the window was moved by only one sample for each crop, but this would be a choice in the implementation. In [62, 71], cropped sets were used in the training phase and the testing phase. During testing, each trial would be split into a set of crops, these would be evaluated individually, then the results from all crops would be aggregated into one prediction for the original trial. In [24] the sliding window was only applied to the training data, while the testing data was left unchanged. Naturally, the data in the test set still has to have the same shape as the training data. To achieve this, the test data can use the center of the crops performed on the training data. For the example above, the test set would be cropped to $\text{crop}_2 = \{t_1, t_2 \ldots, t_8\}$. This is how the sliding window method was applied in this project. Different sliding window configurations were tested with the validation subjects. The configuration selected for the final test used three windows, sliding the window by two samples each time. With this configuration, the number of epochs available for training is tripled, as each original example will be transformed into three epochs shifted by two samples in time.

3.4.7 Training and Testing Routine

After exploring and testing different methods such as data augmentation, channel selection, and TOI on the validation subjects, the best configurations can be tested on the test subjects. Chapter 4 will present the various tests performed and the results obtained. All these final tests were performed with cross-validation. This is a method for increasing the statistical certainty of the test results and is often used for small datasets. With cross-validation, the dataset is divided into a set of equal partitions. Then, the classifier is trained on the data from all but one of these partitions. After training, the last partition is used to test the trained classifier, and the test accuracy is stored. Then the process is repeated, leaving out a different partition. Each repetition is called a fold. When the process has been repeated for all possible folds, the standard deviation and average of all accuracies can be calculated. In Figure 3.24 the cross-validation process is illustrated for a five-fold scenario, which was used in this work. As explained in section 3.1, each subject defines its own ML problem, so cross-validation is performed for each subject individually.

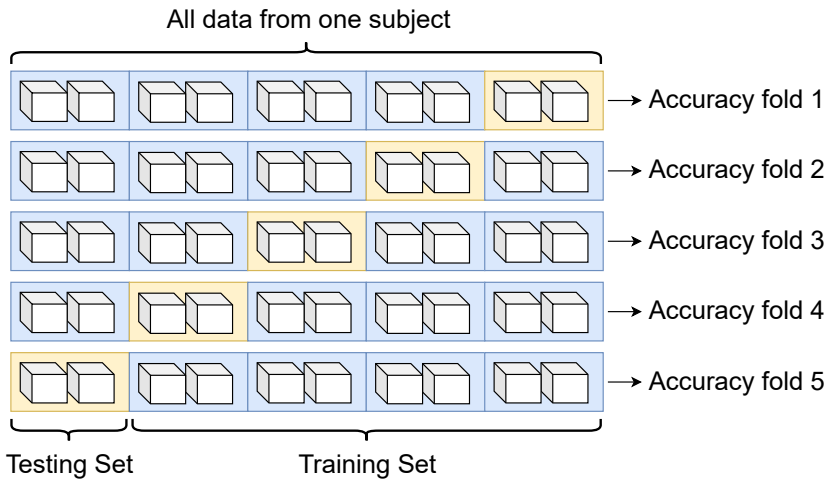


Figure 3.24: The cross validation method. This example illustrates a five-fold cross-validation.

An important hyperparameter for any DNN is the number of iterations used during training. For each iteration the algorithm will update the model parameters to better adapt to the training data, thus a large number of iterations can lead to overfitting [30]. Training is computationally expensive, so training more iterations than needed is also unwanted from a resource perspective. On the other hand, if the model is not trained enough, it will not have time to learn from the data. Choosing a suitable number of iterations can be difficult, as the size of the dataset, the architecture, and the number of parameters in the DNN will affect the training process. A method called early stopping was used to decide the number of training iterations for the tests performed in Chapter 4. In early stopping, the training error of each iteration is monitored. Whenever the training error has not decreased for a certain number of iterations, the training is stopped. Instead of running the early stopping for each subject, it was used on one validation subject, and the resulting number of iterations was rounded to the nearest multiple of 10 and used for all other subjects. As will be explained in Chapter 4, several different configurations were tested. Early stopping was used to select an appropriate number of iterations for each of these configurations. The exact number of iterations is provided in Chapter 4.

3.5 Software and Hardware

The methods described in this chapter, be it source reconstruction, genetic algorithms, or deep neural networks, are all established methods, with many previous implementations. Naturally, no two implementations are identical, and different implementations might lead to somewhat different results. So, for the sake of comparability and reproducibility, this section provides insight into exactly what software and hardware were used for the different approaches.

3.5.1 Software

FreeSurfer [25]

FreeSurfer is an open-source software for visualizing and working with MRI data. This software was used to segment the subject MRIs into layers of skull, skin, and brain, using the BEM method as described in subsection 3.2.1. FreeSurfer was also used to create the parcellation of the brain, which was used to analyze ROIs in the selected sources in section 3.3.

MNE-Python [31]

MNE-Python is an open-source Python library for working with neurophysiological data. This software was used extensively for preprocessing the EEG dataset, and for performing the source reconstruction. The exact use of MNE-Python for the preprocessing has been described in further detail in the preliminary work [27]. All visualizations of the brain, source space, and MRI segmentations presented in this work have been produced using the MNE-Python software.

pymoo [6]

Pymoo is an open-source Python library for performing multi-objective optimization. Pymoo was used to implement the data channel selection paradigm in section 3.3. More specifically, the Pymoo framework for defining optimization problems was used to define the chromosome representation and fitness function for the problem. Then, the Pymoo implementation of the NSGA-III algorithm was used to search for optimal solutions to the problem.

Keras [8]

Keras is an open-source Python library, serving as a deep learning framework using TensorFlow [1]. All the DNN classifiers used in this work were implemented using Keras. The implementation of these models in Keras is described in tables 3.7, 3.6, and 3.8. Such a description of the DeepConvNet implementation can be found in the preliminary work [27].

3.5.2 Hardware

Recording Equipment

The data used in this work was recorded as part of a previous study. Details of the hardware used for this recording can be found in [52].

IDUN [63]

IDUN is a high-performance computer cluster at the Norwegian University of Science and Technology (NTNU). This cluster was used to perform the most demanding computations in this work, including the channel selection, all training and testing of DNNs, and the SR. The computer cluster is equipped with NVIDIA A100, V100, and P100 graphics processing units (GPUs), which were used to accelerate the training and testing of DNNs.

Chapter 4

Results

4.1 Tests Performed

To systematically test the different methods and approaches explored in this work, a final set of tests were defined, to be performed using all subjects. Each test is defined such that it singles out a certain method or parameter to be tested independently. This section provides a description of each test and its purpose. The exact details of the parameters in each test and the results are presented in section 4.2.

4.1.1 Test 1: Full vs Minimal Preprocessing

As covered in subsection 3.1.2, two different preprocessing pipelines were developed in this work. One of them, the full preprocessing, uses a number of methods to increase the SNR of the data, while the other uses a minimal amount of preprocessing. Test 1 is designed to investigate how these two pipelines affect the performance of the classification. The test has four sub-tests: full preprocessing in electrode- and source space, and minimal preprocessing in electrode- and source space.

4.1.2 Test 2: All vs Selected Channels

This test evaluates the channel selection from section 3.3. The channel selection was performed based on the performance of the evaluation subjects, this test will demonstrate whether those channel selections are beneficial for other subjects as well. Although many pareto-optimal channel configurations were found in section 3.3, only the best configurations in terms of classification accuracy will be tested in this test. These were the 10-electrode configuration of Table 3.2 and the 151-source configuration of Table 3.3. Four sub-tests were performed: All channels in electrode- and source space, and selected channels in electrode- and source space.

4.1.3 Test 3: Data Augmentation

The methods for data augmentation introduced in subsection 3.4.6, noise injection and sliding window, are applied and tested in this test. The three sub-tests were: No data augmentation, noise injection, and sliding window.

4.1.4 Test 4: 3M3DCNN, 2DCNN, and ST3DCNN

This test explores the different architectures using a structured spatial source representation. These are the 3M3DCNN, 2DCNN, and ST3DCNN architectures, covered in sections 3.4.2, 3.4.3, and 3.4.4. All three architectures require somewhat different input, so there is some difference in the preprocessing of the data. However, since these are necessary operations for the architecture, the test still serves as a way of comparing the performance of the three architectures.

4.2 Test Results

This section presents and describes the results obtained from the four final tests, using the test subjects. The specification of each test is listed in Table 4.1. Some remarks should be made regarding the results. Firstly, subjects are marked either red, gray, or white in the result tables. In [52], some subjects were excluded from the research, due to requirements of correct behavior, no faulty electrodes, and enough data remaining after preprocessing. The subjects considered insufficient for inclusion in [52] are marked red. The validation subjects are marked gray, and the remaining subjects are marked white. The intention of this coloring is to highlight the performance of these different groups of subjects. The last row in each table provides the average accuracy for the given tests across all test subjects (white and red). Behind these accuracies, in parentheses, are the average accuracy of all subjects. The columns marked σ report the standard deviation from the cross-validation of the tests.

Test	Preprocessing	Channel type	Channel selection	Architecture	Data augmentation	Number of training iterations	TOI
T1.1	Full	Electrodes	Best	DeepConvNet	None	80	0-800ms
T1.2	Full	Sources	Best	DeepConvNet	None	80	0-800ms
T1.3 T2.3	Minimal	Electrodes	Best	DeepConvNet	None	60	0-800ms
T1.4 T2.4 T3.3	Minimal	Sources	Best	DeepConvNet	None	50	0-800ms
T2.1	Minimal	Electrodes	All	DeepConvNet	None	50	0-800ms
T2.2	Minimal	Sources	All	DeepConvNet	None	50	0-800ms
T3.1	Minimal	Sources	Best	DeepConvNet	Noise injection	50	0-800ms
T3.2	Minimal	Sources	Best	DeepConvNet	Windowing	50	0-800ms
T4.1	Full	Sources	All	3M3DCNN	None	70	0-400ms
T4.2	Full	Sources	All	2DCNN	None	90	0-400ms
T4.3	Full	Sources	All	ST3DCNN	None	80	0-400ms

Table 4.1: Overview of the different test configurations. For channel selection, 'Best' means that the configuration with the highest accuracy from the GA channel selection was used.

4.2.1 Test 1: Full vs Minimal Preprocessing

The results are presented in Table 4.2. The aim of the test was to compare the two preprocessing pipelines. A significant increase in classification can be seen in both source space and electrode space when using minimal preprocessing instead of the full preprocessing pipeline. In electrode space minimal preprocessing yields an increase in average accuracy of 8%, and in source space the increase is 16%. This result is counterintuitive, as all the steps in the full preprocessing are designed with the intention of increasing the classification performance. Naturally, this is a promising result for online applications, as minimal preprocessing implies a simpler and less computationally complex pipeline.

Subject	T1.1	σ	T1.2	σ	T1.3	σ	T1.4	σ
sub-13	94	3	59	7	96	2	85	3
sub-21	93	2	75	4	93	3	90	3
sub-02	87	4	65	7	97	2	77	5
sub-07	87	2	59	5	87	3	89	3
sub-24	87	5	50	4	91	3	70	4
sub-14	85	2	70	3	94	2	87	6
sub-20	85	2	59	3	89	2	73	5
sub-26	85	4	67	5	84	4	88	3
sub-29	85	3	80	5	93	4	75	12
sub-06	84	4	64	8	92	3	90	4
sub-18	84	5	54	6	91	4	72	4
sub-19	84	5	60	10	92	3	72	5
sub-23	83	4	51	9	78	3	68	3
sub-05	80	2	46	6	84	3	53	8
sub-15	76	3	48	6	79	4	69	5
sub-30	76	4	65	4	84	4	71	14
sub-08	75	4	41	7	89	3	75	3
sub-31	75	3	40	5	79	5	76	3
sub-11	74	2	59	6	81	3	51	5
sub-03	72	3	61	4	80	3	85	4
sub-16	66	4	49	5	79	4	51	5
sub-25	63	3	45	4	84	1	62	6
sub-22	62	1	44	2	74	5	52	3
sub-27	59	8	59	8	62	5	69	9
sub-04	57	4	41	6	80	6	64	4
sub-28	57	3	41	5	75	5	85	7
sub-17	52	5	38	6	52	4	71	6
sub-01	51	2	43	1	58	5	49	5
sub-10	50	4	54	7	59	7	74	5
sub-09	45	4	50	6	52	3	80	6
sub-12	45	2	47	6	65	4	40	7
average	72(73)	4(3)	54(54)	6(5)	80(80)	4(4)	70(71)	5(5)

Table 4.2: The results from test 1, comparing full and minimal preprocessing. **T1.1:** Full preprocessing in electrode space. **T1.2:** Full preprocessing in source space. **T1.3:** Minimal preprocessing in electrode space. **T1.4:** Minimal preprocessing in source space.

This result will be discussed further in subsection 5.2.1. It can also be observed that the average accuracy is almost identical with or without the validation subjects. This result indicates that there has not been an overfitting to the validation subjects. Lastly, the subjects marked in red tend to perform worse than other subjects, demonstrating that correct behavior and working electrodes are important factors for the performance of the decoding.

4.2.2 Test 2: All vs Selected Channels

The results are presented in Table 4.3. For this test, the expected result was that the use of channel selection would improve the classification in both source space and electrode space. However, the results show no significant difference between the use of all channels and the use of selected channels, neither in electrode space nor source space. The average accuracy with selected channels in source space is 3% higher than with all electrodes, but considering that the standard deviation is 4%,

Subject	T2.1	σ	T2.2	σ	T2.3	σ	T2.4	σ
sub-02	94	3	72	5	97	2	77	5
sub-13	94	3	86	4	96	2	85	3
sub-29	93	2	78	4	93	4	75	12
sub-06	92	3	89	4	92	3	90	4
sub-21	92	3	89	5	93	3	90	3
sub-14	91	4	90	3	94	2	87	6
sub-18	90	4	71	4	91	4	72	4
sub-24	90	2	72	2	91	3	70	4
sub-07	89	3	93	2	87	3	89	3
sub-23	88	3	65	8	78	3	68	3
sub-19	87	4	73	5	92	3	72	5
sub-08	84	2	75	5	89	3	75	3
sub-05	83	6	54	11	84	3	53	8
sub-20	83	6	70	3	89	2	73	5
sub-26	82	4	90	3	84	4	88	3
sub-11	81	2	55	6	81	3	51	5
sub-16	81	4	53	3	79	4	51	5
sub-25	80	5	64	2	84	1	62	6
sub-30	80	5	91	10	84	4	71	14
sub-03	78	5	89	2	80	3	85	4
sub-15	78	6	74	3	79	4	69	5
sub-04	76	4	59	2	80	6	64	4
sub-31	73	8	76	3	79	5	76	3
sub-22	72	4	52	3	74	5	52	3
sub-28	70	2	77	6	75	5	85	7
sub-10	60	7	68	10	59	7	74	5
sub-12	60	2	40	4	65	4	40	7
sub-01	53	4	47	3	58	5	49	5
sub-27	53	2	67	6	62	5	69	9
sub-09	50	4	82	7	52	3	80	6
sub-17	48	3	67	8	52	4	71	6
average	77(78)	4(4)	70(72)	5(5)	80(80)	4(4)	70(71)	5(5)

Table 4.3: The results from test 2, comparing channel selection vs. all channels. **T2.1:** Electrode space all channels. **T2.2:** Source space all channels. **T2.3:** Electrode space selected channels. **T2.4:** Source space selected channels.

this is not a very notable difference. Although the channel selection did not lead to any increase in performance, the result is still very relevant, as the number of channels has been reduced substantially, without loss of performance. The channel selection was one of the classifier configurations most heavily adapted to the validation subjects. Still, the results show that the performance is similar between validation and test subjects.

4.2.3 Test 3: Data Augmentation

The results are presented in Table 4.4. Data augmentation was only tested with source space data. Both noise injection and sliding window have proved to be useful techniques for improving EEG decoding in previous studies, however, the improvement is minimal in this work. The average accuracy has an increase of 1% and 3% for the noise injection and sliding window respectively. This increase is not very significant, as the results will vary somewhat anyways, as indicated by the standard de-

viation of 5%. Although the increase in performance was not that notable on average, sub-30 had a significant improvement when using data augmentation. In fact, the accuracy reached for sub-30 using sliding window (T3.2) was 99%, the highest accuracy achieved of any subject in any of the tests performed.

Subject	T3.1	σ	T3.2	σ	T3.3	σ
sub-30	91	11	99	1	71	14
sub-06	90	4	91	3	90	4
sub-07	90	2	91	4	89	3
sub-14	90	6	91	4	87	6
sub-26	90	2	88	3	88	3
sub-03	88	3	86	2	85	4
sub-09	87	3	88	2	80	6
sub-21	86	4	89	3	90	3
sub-13	85	2	85	2	85	3
sub-28	81	8	90	2	85	7
sub-02	80	7	78	7	77	5
sub-29	80	8	85	4	75	12
sub-31	79	2	77	5	76	3
sub-08	78	4	76	7	75	3
sub-10	74	9	74	10	74	5
sub-20	74	3	75	4	73	5
sub-15	72	2	75	8	69	5
sub-24	71	5	72	3	70	4
sub-27	71	6	71	6	69	9
sub-17	69	6	71	4	71	6
sub-18	68	2	70	3	72	4
sub-19	68	4	73	5	72	5
sub-23	67	3	70	2	68	3
sub-04	66	5	61	4	64	4
sub-25	64	4	67	3	62	6
sub-11	56	9	61	5	51	5
sub-05	54	6	60	7	53	8
sub-22	53	3	52	4	52	3
sub-16	52	7	52	5	51	5
sub-01	47	3	49	5	49	5
sub-12	44	6	45	4	40	7
average	71(73)	5(5)	73(75)	5(4)	70(71)	5(5)

Table 4.4: The results from test 3, comparing data augmentation methods. **T3.1:** Noise injection. **T3.2:** Sliding window. **T3.3:** No data augmentation.

4.2.4 Test 4: 3M3DCNN, 2DCNN, and ST3DCNN

The results are presented in Table 4.5. All three architectures performed worse on average than the deepConvNet. The novel architecture ST3DCNN performs better than 3M3DCNN and 2DCNN. It is worth noting that the best results obtained in source space with deepConvNet using full preprocessing were an average accuracy of 54% in test T1.2, which is only 6% more than the results obtained with ST3DCNN. Thus, using spatial convolutions for decoding RGB responses demonstrates some potential, which will be further discussed in subsection 5.2.4. Although minimal preprocessing has proved to be favorable for accuracy, it does increase the data size. This is further discussed in sub-

section 5.2.1. Due to this increase, the computational complexity of testing 3M3DCNN, 2DCNN, and STDCNN with minimal preprocessing is substantial. To keep the complexity reasonable, these models were therefore only tested with full preprocessing.

Subject	T4.1	σ	T4.2	σ	T4.3	σ
sub-13	70	13	48	4	71	5
sub-29	65	7	47	5	78	6
sub-26	61	4	39	2	60	6
sub-03	60	7	34	6	56	8
sub-02	57	14	50	7	54	7
sub-14	57	7	41	5	53	5
sub-21	56	7	48	5	78	5
sub-06	51	7	42	2	63	8
sub-23	49	6	41	5	53	5
sub-30	48	3	43	7	56	8
sub-11	47	2	44	3	45	5
sub-24	44	6	37	3	50	2
sub-19	42	5	42	3	47	7
sub-07	41	4	39	3	42	7
sub-18	41	7	39	5	45	5
sub-05	40	7	41	4	39	4
sub-08	39	4	39	6	39	7
sub-15	39	5	37	5	34	5
sub-20	39	5	38	6	39	6
sub-27	38	11	42	10	46	6
sub-31	38	5	44	5	41	5
sub-09	37	7	36	12	43	9
sub-16	37	1	42	4	41	6
sub-25	37	2	40	4	44	7
sub-28	37	3	38	4	40	4
sub-01	36	3	34	4	31	4
sub-22	36	4	39	5	40	2
sub-04	34	4	34	7	38	8
sub-10	33	5	37	3	36	3
sub-17	33	2	35	4	41	3
sub-12	30	8	38	4	39	6
average	44(44)	6(6)	40(40)	5(5)	48(48)	5(6)

Table 4.5: The results from test 4, comparing 3M3DCNN, 2DCNN and ST3DCNN. **T4.1:** 3M3DCNN. **T4.2:** 2DCNN. **T4.3:** ST3DCNN.

4.2.5 Comparison with Previous Studies

The dataset used in this work was also used to test classifiers in [52] and in the preliminary work [27]. Therefore, results from these studies offer the opportunity to directly compare the performance of the approaches taken in this work with those employed in previous research. Table 4.6 compares the best results from these previous studies with the results obtained in this work. Since [52] only included a subset of subjects in their study, the average accuracies achieved among those subjects are provided in the column "Good subjects". In [52], the best classifier in electrode space used a minimum distance to mean with geodesic filtering (FgMDM) Riemannian classifier. Their best results in source space were obtained using shrinkage linear discriminant analysis (sLDA). The best results in

Test/Study	Channel Type	Acc. all subjects	Acc. good subjects
T1.1	electrodes	73	81
T1.2	sources	54	57
T1.3 = T2.3	electrodes	80	88
T1.4 = T2.4 = T3.3	sources	71	73
T2.1	electrodes	78	86
T2.2	sources	72	73
T3.1	sources	73	74
T3.2	sources	75	76
T4.1	sources	44	48
T4.2	sources	40	42
T4.3	sources	48	52
[27]	sources	54	58
[27]	electrodes	77	84
[52]	sources	-	50
[52]	electrodes	-	74

Table 4.6: Comparison with results from a previous study [52] and the preliminary study for this work [27], which both used the same dataset as in this work. The column "Good subjects" are the accuracies averaged across subjects fulfilling the criteria for inclusion in the dataset in [52]. The column "All subjects" are the accuracies averaged across all subjects.

the preliminary studies, for both source- and electrode space, were obtained using the deepConvNet classifier. The best results in electrode space from this study outperform all previous results, while the source space results of this study have an increase in accuracy of over 20% compared to previous results in source space.

4.2.6 Confusion Matrices

Confusion matrices are one way to present the performance of a classifier. These matrices have two dimensions: one represents the actual class of an example, and the other represents the predicted class by the classifier. In a confusion matrix $C_{\text{classes}\times\text{classes}}$, element $c_{i,j}$ represents the number of examples with the actual class i, which are classified as class j. An ideal classifier will only have elements on the diagonal, as all examples are classified into the correct class. If any examples are wrongly classified, the confusion matrix provides information about which classes the classifier confuses for other classes. Confusion matrices have been produced for all the tests, and are presented in figures 4.1, 4.2, 4.3, and 4.4. The matrices include the data from each fold in the cross-validation from each subject, so all possible examples are included. It is important to note that the number of examples from each class is not exactly the same, which means that the coloring in the matrices can be slightly misleading as they represent the number of examples classified to a given class, not the percentage.

When analyzing the confusion matrices, the interesting aspect is the distribution of the wrongly classified examples. For instance, in the T1.1 matrix of Figure 4.1, all the off-diagonal elements are fairly similar, while in T1.2 it is clear that red is classified wrong significantly more often than blue is. When considering all the confusion matrices, a few common trends can be observed. Red is classified as green more often than it is classified as blue, and blue is classified as green more often than it is classified as red. For green, there is a fairly equal distribution of misclassifications. This indicates that red and blue are the easiest colors to separate.

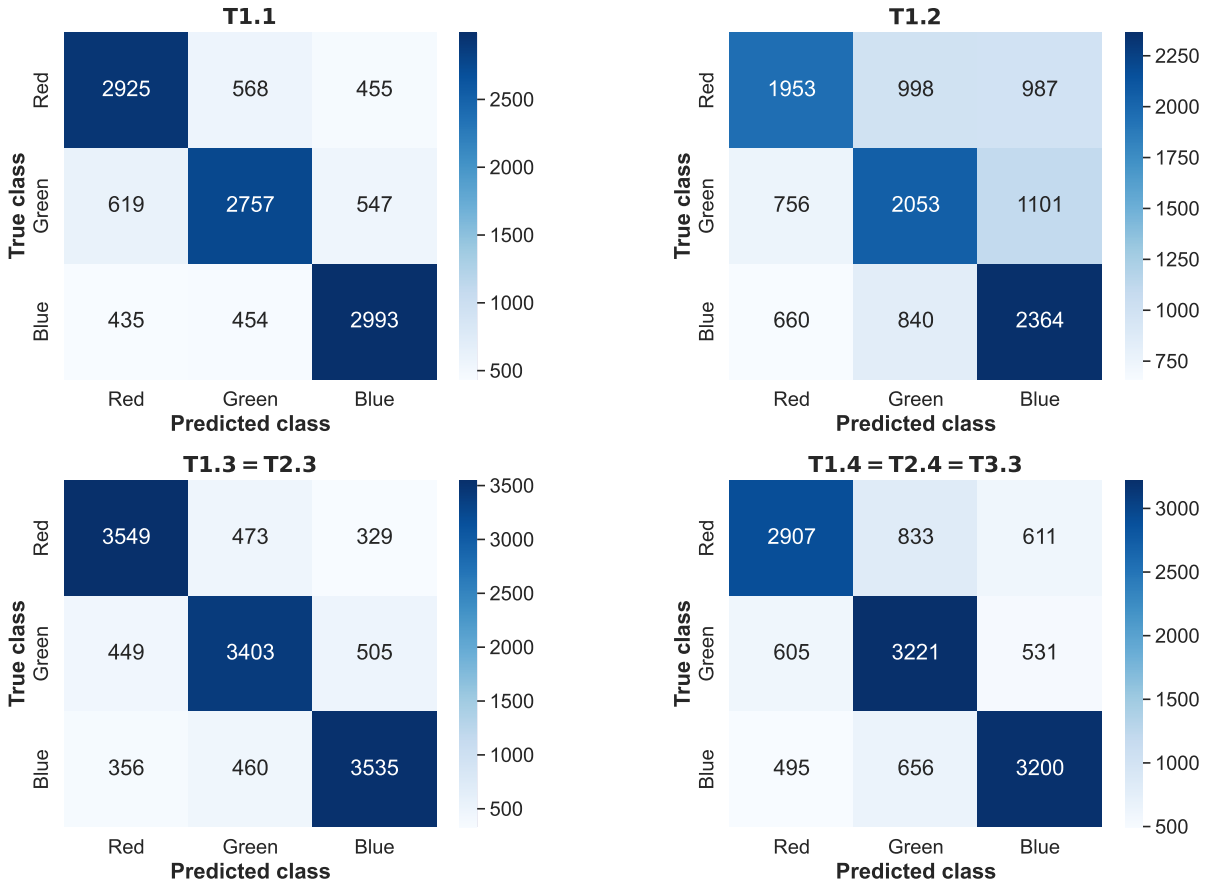


Figure 4.1: Confusion matrices from Test 1

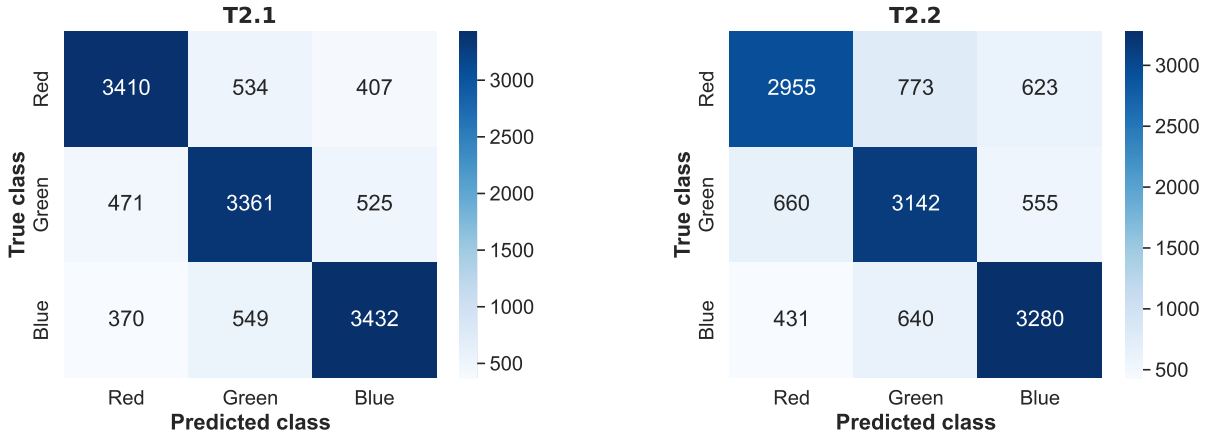


Figure 4.2: Confusion matrices from Test 2

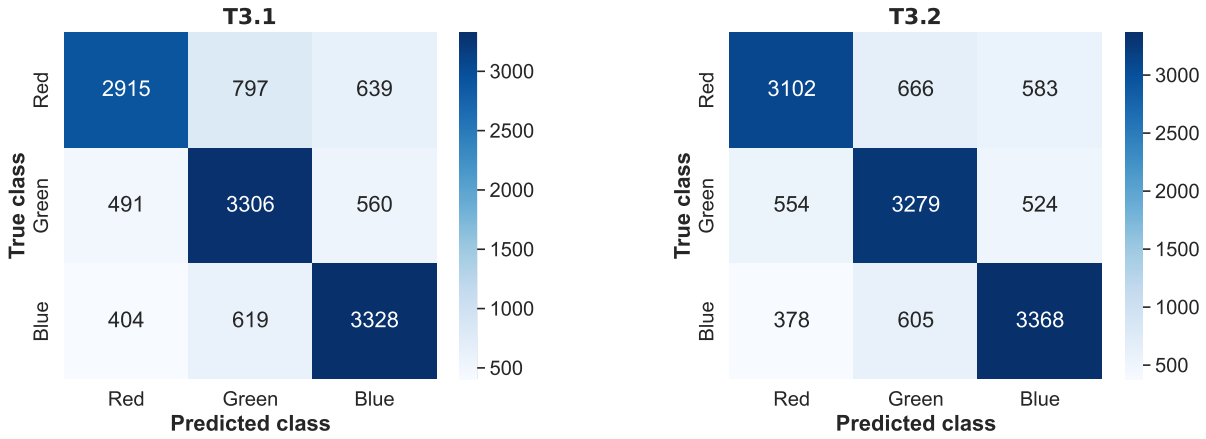


Figure 4.3: Confusion matrices from Test 3

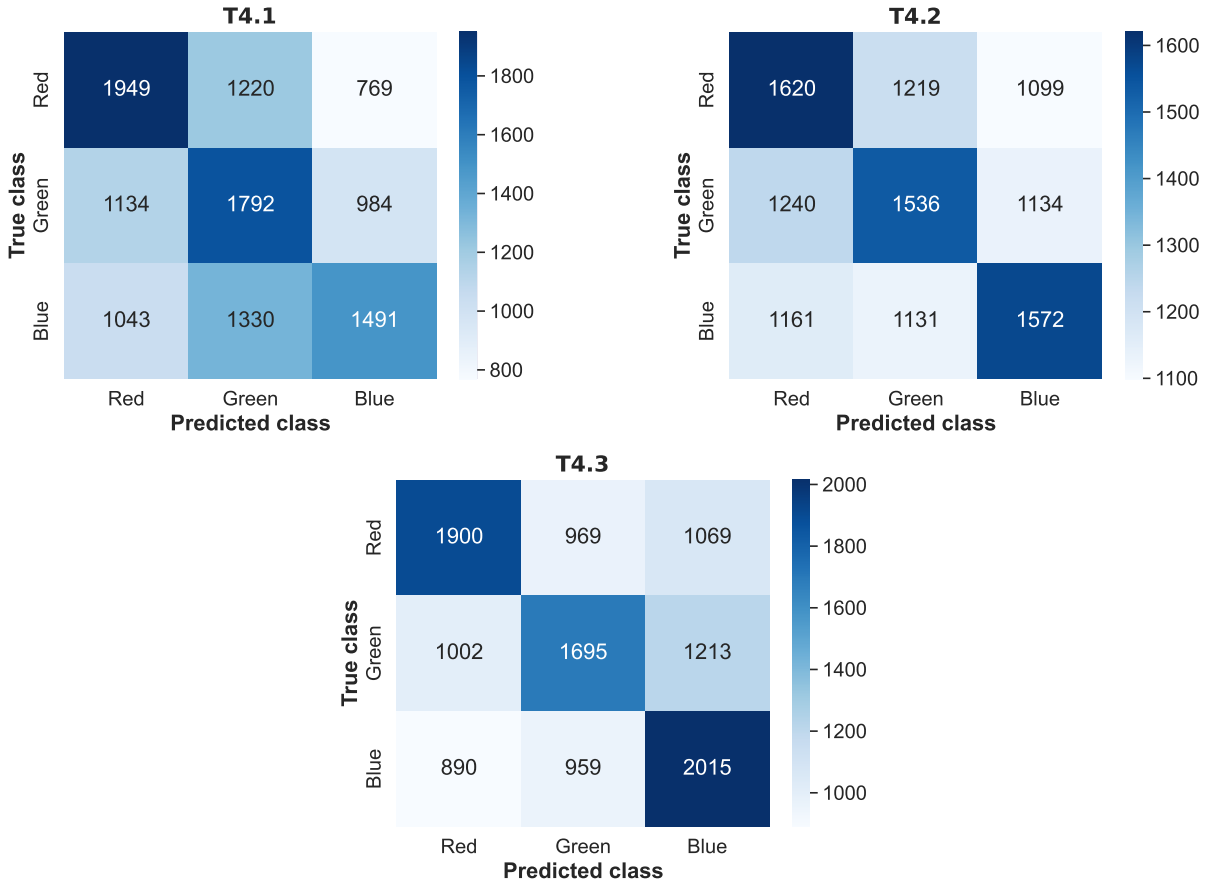


Figure 4.4: Confusion matrices from Test 4

Chapter 5

Discussion, Conclusion, and Further Work

5.1 Summary of Findings

The aim of this work has been to explore different methods for improving the classification of RGB responses in EEG signals. Three main methods were explored: Source reconstruction, channel selection with GA, and classification with DNNs. In addition to these methods, some different preprocessing pipelines and data augmentation has also been implemented. A set of tests were defined to evaluate the performance of each method. Each of these tests can be thought of as a specific pipeline for the data, as shown in Figure 5.1. In general, it was found that pipelines including minimal preprocessing performed better. It was also observed that classification in electrode space outperforms classification in source space. In an effort to take advantage of the high spatial resolution of the source space, three DNN architectures designed specifically for source-reconstructed data were implemented. None of these performed better than any of the pipelines using the deepConvNet architecture. Both methods for data augmentation led to a minor increase in performance. The selection of channels with GA led to a 3% increase in accuracy with electrodes, but no change in electrode space. So the effects on accuracy were relatively insignificant. However, the effect on channel reduction was substantial. In electrode space, the reduction in channels was from 58 to 10 electrodes, a reduction of almost 83%. In electrode space the reduction was from 474 to 192 sources, a reduction

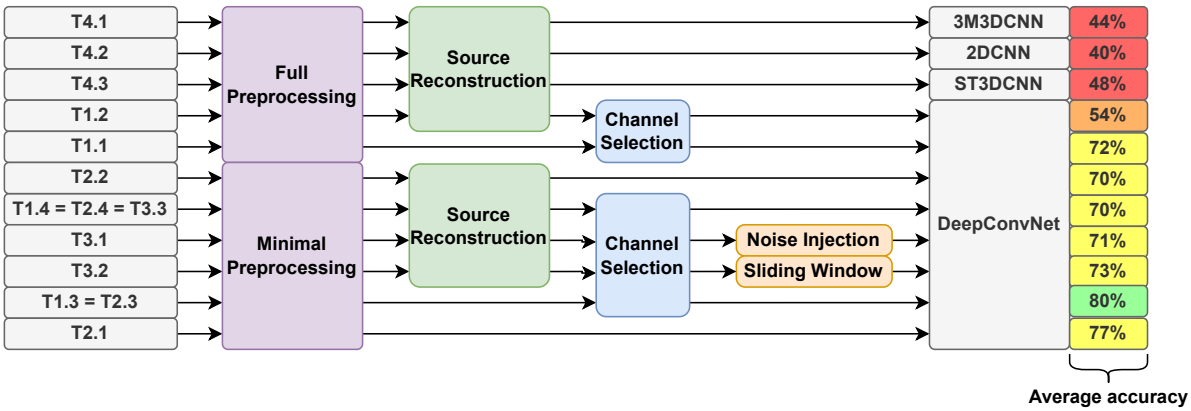


Figure 5.1: The various data pipelines that have been explored and the accuracies achieved with them.

of almost 60%. The significance and implication of all these findings will be discussed in section 5.2. Some suggestions for further work, based on the results and experiences obtained in this work, are provided in section 5.3

5.2 Discussion of Findings

5.2.1 Choice of Preprocessing

Test 4 compared the full and minimal preprocessing pipelines, and yielded a somewhat unexpected result. With minimal preprocessing, the classifiers performed better in both source space and electrode space. This result is favorable for online applications, as it means higher accuracies can be attained, while reducing computational complexity. But the result is unexpected, as the operations in the full preprocessing are designed to reduce SNR and thus facilitate better classification. One possible explanation for this counterintuitive result is that some data is discarded in the full preprocessing. This rejection is based on artifacts and high PTP amplitudes. Since the minimal preprocessing does not discard any data, it has a larger training set. That the minimal preprocessing yields better results could indicate that the data discarded by the preprocessing is still descriptive enough to improve the generalization of the classifiers. Another possible cause is the filtering and downsampling performed in full preprocessing. A low-pass filter of 45Hz was applied and followed by a downsampling from 1000Hz to 200Hz, which reduces the data size by 80%. This process removes information about higher frequencies and reduces data size, which might contribute to the reduction of classification accuracy. To investigate the effect of the increased number of training examples with minimal preprocessing, Table 5.1 was created. The table shows the difference in data size and classification accuracy for each subject when going from full to minimal preprocessing. The data is taken from the T1.2 and T1.4 tests. If minimal preprocessing yields an increase in accuracy due to an increased number of training examples, subjects with a large difference in the amount of training data should also have a large difference in accuracy. So the two columns in Table 5.1 should be positively correlated. However, the table gives no indication of such a correlation. For instance, subject 12 has a data increase of 40%, but the accuracy is reduced by 12%. Thus it seems likely that the downsampling could be the cause of the difference in classification accuracy between full and minimal preprocessing pipelines.

5.2.2 Source Space for Improved Classification

The results indicate that classification is not improved by a source space representation of the data. This was also the result found in the preliminary studies. It was expected that channel selection could be a method for facilitating better classification in source space, but results from T2.2 and T2.4 indicate no increase in accuracy with channel selection. The use of DNN architectures specifically designed to classify source space data was another approach explored for improving source space classification. However, all three architectures performed particularly poorly, compared to the deepConvNet architecture. There are some caveats with the implementation of both channel selection and the three new architectures which might explain their inability to yield better performance. These caveats are discussed in sections 5.2.3 and 5.2.4. Still, the research indicates that source space data overall is less suited for classification than the raw electrode data.

One possible explanation is that the parameters of the source reconstruction are not optimal. The

Subject	T1.2 → T1.4	
	Δ Data (%)	Δ Accuracy (%)
sub-09	+134	+60
sub-27	+96	+17
sub-21	+43	+20
sub-12	+40	-15
sub-07	+18	+51
sub-22	+18	+18
sub-19	+17	+20
sub-04	+16	+56
sub-10	+13	+37
sub-29	+10	-6
sub-17	+9	+87
sub-15	+8	+44
sub-23	+8	+33
sub-25	+8	+38
sub-08	+7	+83
sub-01	+4	+14
sub-05	+4	+15
sub-16	+3	+4
sub-02	+2	+18
sub-03	+2	+39
sub-11	+2	-14
sub-26	+2	+31
sub-28	+2	+107
sub-06	+1	+41
sub-18	+1	+33
sub-24	+1	+40
sub-30	+1	+9
sub-13	+0	+44
sub-14	+0	+24
sub-20	+0	+24
sub-31	+0	+90

Table 5.1: A comparison between the increase in data examples and the increase in accuracy between minimal and full preprocessing. This is based on the source space results from tests T1.2 and T1.4.

results of Table 3.4 could support this. These tables compared source selection on sources reconstructed based on the FsAverage template head and individual MRI data. Since the SR will be more accurate when using MRI data from the individual, it was expected that the results would be better with the individual models. However, there is no noticeable difference in performance between the two. This could be an indication that the SR is not optimal.

5.2.3 A Review of Channel Selection

Channel selection with the NSGA-III algorithm was performed in both electrode space and source space. In electrode space using the selected electrodes as opposed to all electrodes yielded an increase in accuracy of only 3%. The source space channel selection led to no increase in accuracy. The hypothesis was that the channel selection would improve classification, by removing channels that lowered the SNR of the data. That no such improvement was found could indicate that the DNNs are robust to channels that do not contribute information about RGB responses. This would explain why the performance is similar when using all channels and a selected optimal subset.

The GA used in the channel selection returned a set of solutions. Each solution was a configuration of channels and their fitness (accuracy and number of channels). These were the results presented in Figure 3.12 and Figure 3.15. It was pointed out that when these configurations were tested again, the results had a slightly lower accuracy. This trend was observed for all configurations in both source- and electrode space. For each individual in each generation of the GA, the DNN is trained on the training examples. This training is based on the stochastic optimizer ADAM [43]. Since there is a stochastic element to the training phase, the same channel configuration will not give exactly the same results if trained twice. This could explain why the GA returned slightly better results than those observed during later testing. Since the GA trains and tests very many configurations, it is likely that some of them end up having an especially favorable training phase, and that these will be returned as solutions. Hence, the GA might yield somewhat optimistic results.

Certain aspects of the implementation could contribute to the limited improvement obtained with channel selection. For instance, a template head was used to create a common source space for all subjects. The SR will be less accurate with this approach, as the template head always deviates somewhat from the actual head geometries of the subjects. One method to remedy the issue with a template head is morphing all the individual source spaces [32] into a common space. With this method, the source estimates are computed using the individual forward models. Then the source estimates are morphed into a template head. This way the individual MRI can be taken advantage of, while still having the same source space definition across subjects.

That the channel selection yielded better results in electrode space than in source space could also indicate some shortcomings in the implementation. As touched upon in subsection 3.3.3, the optimization problem becomes drastically more complex in source space, as the number of channels is substantially larger. Due to the computational complexity, the same population size and number of generations in the GA were used for both electrode and source selection. It is reasonable to assume that the number of source configurations explored with these parameters is too small to converge on an optimal solution. This would also explain why the full brain source selection seemed to distribute sources across the whole brain. There could be that the channel selection would converge on a small subset of sources in certain ROIs if only it had a larger population and a larger number of generations.

Due to the large number of sources in the source space, it was not computationally feasible to perform source selection on the complete source space. A routine for selecting a subset of sources based on electrodes from electrode selection was developed. With this method, 474 of the 8196 sources were selected, and channel selection with GA was performed on those 474 sources. The routine for selecting those 474 sources was relatively pragmatic. It is possible that this routine excludes some sources that would contribute relevant information for the classification. For instance, since the electrodes are on the surface of the head, the volume selection around these electrodes will favor sources closer to the surface. Since the V8 region described in Figure 2.4 is on the underside of the brain, sources in this region were not selected with this routine.

5.2.4 Structured Spatial Source Representation

The voxelization and pixelization of source space were performed to obtain a source space representation that allowed spatial convolutions. The aim was to explore DNN architectures with spatial

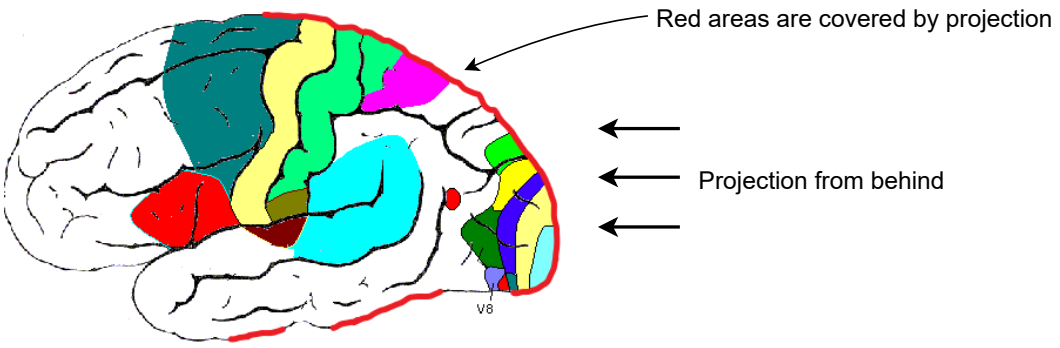


Figure 5.2: The effects of pixelization by projection from behind. The area V8, involved in color perception, is badly covered by such a projection. Adapted from [10].

convolutions, that could take advantage of the high spatial resolution in source space, by finding more complex spatial features. Although this approach has demonstrated good results in other studies [50, 24], the three architectures explored with this approach in this work were all outperformed by deepConvNet.

Tests T4.1, T4.2, and T4.3, testing the three architectures with spatial convolutions, were all performed using the full preprocessing pipeline. When comparing them to the only test of deepConvNet using full preprocessing (T1.1), the difference in performance is not as big. T4.3, using the ST3DCNN architecture got an average accuracy of 48% compared to 54% achieved in T1.1. Since the change to minimal preprocessing yielded a substantial increase in performance with deepConvNet, it is natural to assume that the other architectures would also benefit from this pipeline. The higher sampling rate of the minimal preprocessing made the voxelized and pixelized source space considerably larger, making the training phase very computationally heavy. Due to this, the 3M3DCNN, 2DCNN, and ST3DCNN architectures were only tested with full preprocessing.

One factor that could limit the performance of the 2DCNN and ST3DCNN architectures, is the geometry of the brain in the ROIs for color responses. Both these architectures expect an input with two spatial dimensions. To achieve this representation a projection from behind is performed on the voxelized data. The V8 area, which has been proven to be involved in color perception, is not well covered by this projection, due to the geometry of the brain. Figure 5.2 illustrates this problem. In the paper introducing 2DCNN, the projection is done from above, to cover the areas of the brain related to movement, as the activity to decode was motor imagery. The shape of the top of the brain is flatter, and thus the 2D projection will get better coverage. This could be part of the reason that 2DCNN and ST2DCNN did not perform well on the task of decoding RGB color responses.

Another conclusion that could be drawn from the results, is that spatial features are not as descriptive for RGB color responses as for motor imagery tasks. The 3M3DCNN and 2DCNN architectures were both used to classify motor imagery in their original papers. The 3M3DCNN architecture is not limited by the projection, as it takes 3D voxel grids as input, yet it still performs significantly worse than the deepConvNet architecture. This could indicate that temporal features are more relevant for distinguishing color responses than spatial features.

5.2.5 Data Augmentation

On average, neither of the data augmentation methods employed led to any notable increase in classification accuracy. Still, it only affects the time it takes to train the classifiers, it does not add any computational complexity when using the classifiers, and so does not affect the applicability to online uses. Given this, and its ease of implementation, the methods seem sensible to employ in any EEG decoding paradigm. For subject 30, the accuracies with no data augmentation, noise injection, and sliding window were 71%, 91%, and 99% respectively. It is not clear why this subject had such an increase in performance, but it is possible that it is partly due to the stochastic nature of the training and testing process. The random segmentation into training and testing sets, and the training procedure itself will cause the results to vary somewhat. Thus, it could be that tests T3.1 and T3.2 had very advantageous outcomes from these stochastic operations for subject 30. This being said, the performance increase was so significant for both tests, that it is natural to assume that data augmentation was especially favorable for the data from subject 30. It indicates that data augmentation may be of great advantage in some cases.

5.2.6 A Review of BCI Applicability

One of the main motivations for exploring the decoding of color responses in EEG signals is to assess the possibility of it being used in a BCI system. The work in this thesis has led to insight into the applicability of both the RGB stimuli paradigm and the methods employed.

RGB Stimuli as Control Task

Naturally, the decoding accuracy of the control tasks is an important measure of their usability. What should be considered an acceptable accuracy depends on the system to be controlled and on the user's requirements. If the BCI turns on and off lights, one might accept a lower accuracy than if it is used to turn on an alarm requesting immediate assistance. The best results reported in this work (T1.3) had an accuracy of 52% for the worst-performing subjects, and 97% for the best-performing subject. An accuracy of 97% might be considered useful in some scenarios, but 52% should certainly be considered unacceptable. However, all the subjects with the worst performance in this test had faulty Oz and O2 electrodes or were noted to be sleepy or sleeping. The worst subject, if disregarding the ones with faulty electrodes and incorrect behavior, had an accuracy of 75%. This is still low, as it means one in every four inputs to the BCI would result in the wrong command. Lastly, it should be reinstated that the data used in this research is likely unobtainable in a real-world scenario. The recording was done under optimal conditions, with the stimuli coming from a screen placed directly in front of the subjects. If the stimuli were coming from colored signs taking up less of the subjects' field of view, it is only natural to assume the decoding would perform worse. In a realistic scenario, the user would also be presented with several colors, and only focus on one of them, it is also likely that this would make the decoding more difficult.

Feasibility of Methods in an Online Scenario

Although many of the methods employed in this work were computationally demanding, they might still be useful for online applications. For instance, the channel selection is a method that would be

employed offline during development, and which actually would reduce the computational complexity for the online scenario. It has been demonstrated in this work that channel selection can be successfully employed to reduce the number of channels, without loss of accuracy. Another important result from the channel selection was that the validation subjects and the test subjects had similar performances using the selected channels. This means that a selection of sources or electrodes that are favorable for one person tends to be favorable for other persons as well. For BCIs, this is relevant, as it means channel selection can be performed as part of the development, without needing data from the users of the BCI.

The use of SR would require a bit more computation also for the online scenario. Computing the forward model, the most demanding part of the SR process, only has to be done once, so it can be part of an offline setup process. Still, computing the inverse solution has to be done online, and the resulting data will be much larger than the raw electrode data. Since SR did not improve classification in this research, it would not be sensible to employ it in a BCI, but for approaches where SR successfully improves classification, source selection could be a great way to make it more suitable for online applications. The discovery that the minimal preprocessing approach yielded better results is also relevant for BCI applications. Since the minimally preprocessed data has a higher sampling frequency, it is not necessarily a less computationally costly approach, so this would have to be investigated further to determine the optimal preprocessing for an online scenario.

5.3 Suggestions for Further Work

5.3.1 Inter-Subject Classifiers

This work has only explored intra-subject classifiers. If such classifiers are to be realized in a BCI, each new user is required to go through the process of recording their responses to RGB stimuli, in a similar fashion to the subjects used for the dataset in this research. Creating an inter-subject classifier would alleviate new users from an exhausting and potentially costly recording session. One possible way to implement such an inter-subject classifier would be to train a classifier on a pool of subjects during development, and afterwards adapt the classifier to a new user by training it on a relatively small set of examples for this new user. Such an approach has been explored with success in previous studies [23].

5.3.2 Investigate Different Source Reconstruction Methods

The comparison between source selection in the subject-specific source space and the template source space, indicated no advantage with using the individual source space. This could be a sign that the SR is not optimal. An interesting subject for further study would be a systematic evaluation of different SR methods and parameters. Several sets of data could be created using different SR approaches and then tested using the classifiers explored in this project. The classification accuracy could then serve as an indirect measure of how successful the SR was. The dSPM method was employed in this study, but MNE-Python also has implementations of the sLORETA [58] and eLORETA [57] methods. Experimenting with these methods and their parameters, could lead to more accurate SR and thus better classification performance.

5.3.3 4D Spatiotemporal Convolutions

The complete voxel grid representation of source space data has four dimensions: three spatial and one temporal. Still, none of the DNNs employed in this work has performed 4D convolutions. ST3DCNN explored the combination of spatial and temporal convolutions, but was limited to two spatial dimensions, as Keras does not provide 4D convolutional layers. Developing an architecture that can perform 4D convolutions on source space data would be an interesting topic for future studies.

5.3.4 More Extensive Source Selection

The electrode selection led to several configurations, one of which used only 4 of the 58 available electrodes. This configuration performed only marginally worse than the other configurations when testing on the validation subjects. These four electrodes were all in the occipital region, indicating that the channel selection could be used not only to improve classification, but also to localize areas of interest for a given control task paradigm. In source space, the channel selection did not single out areas in the same manner, but seemingly returned sources with a quite even distribution throughout the original pool of sources. As discussed, this could be due to the limited scope of the GA. Performing source selection on the full source space, with a larger population size and a larger number of generations, might result in a more distinct selection of sources. Such a selection could be interesting not only in the context of classification, but could also reveal the spatial properties of color responses with high spatial resolution. Such knowledge could for instance be used to optimize the positioning of EEG electrodes for the RGB control task.

5.4 Conclusion

The work presented in this thesis has provided grounds for assessing certain methods in the context of classification of RGB visual stimuli in EEG signals, both in terms of their ability to yield high classification accuracies and in the feasibility of their implementation in a BCI system. The work has shown that channel selection by use of a GA could reduce the number of channels in both electrode- and source space, without loss of classification accuracy. Channel selection was most effective in electrode space, where it had a channel reduction of 83%, and caused an increase in accuracy of 3%. That channel selection was less advantageous in source space could be attributed to the larger number of channels to choose from. Performing channel selection in source space with a larger population and for more generations may result in a larger channel reduction, a higher classification accuracy, and a better understanding of the spatial distribution of color responses in the brain. The use of source space data did not yield an increase in performance in this work, despite its high spatial resolution. This being said, the DNNs implemented to take specific advantage of the source space representation were somewhat limited due to computational complexity and framework restrictions, and suggestions for further work on the topic have been provided. Lastly, the results of this work demonstrate that responses to RGB visual stimuli can be classified from EEG signals with an accuracy of 88%, averaged across subjects, under the correct conditions. This does not yet provide evidence that RGB stimuli can serve as the control tasks in a robust BCI, especially considering that the data was recorded under optimal conditions. However, the best subject had an accuracy of 97%, demonstrating that higher accuracies are obtainable.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[2] Mohammed Abo-Zahhad, Sabah M Ahmed, and Sherif N Abbas. A new eeg acquisition protocol for biometric identification using eye blinking signals. *International Journal of Intelligent Systems and Applications*, 7(6):48, 2015.

[3] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[4] Sara Hegdahl Åsly. Supervised learning for classification of eeg signals evoked by visual exposure to rgb colors. Master's thesis, NTNU, 2019.

[5] Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: exploring the brain, enhanced edition: exploring the brain*. Jones & Bartlett Learning, 2020.

[6] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8: 89497–89509, 2020.

[7] Marc H Bornstein. Qualities of color vision in infancy. *Journal of experimental child psychology*, 19(3):401–419, 1975.

[8] François Chollet et al. Keras. `https://keras.io`, 2015.

[9] Febo Cincotti, Donatella Mattia, Fabio Aloise, Simona Bufalari, Laura Astolfi, Fabrizio De Vico Fallani, Andrea Tocci, Luigi Bianchi, Maria Grazia Marciani, Shangkai Gao, Jose Millan, and Fabio Babiloni. High-resolution eeg techniques for brain–computer interface applications. *Journal of Neuroscience Methods*, 167(1):31–42, 2008. ISSN 0165-0270. doi: https://doi.org/10.1016/j.jneumeth.2007.06.031. URL `https://www.sciencedirect.com/science/article/pii/S0165027007003378`. Brain-Computer Interfaces (BCIs).

[10] Wikimedia Commons. Constudproc, 2005. URL `https://upload.wikimedia.org/wikipedia/commons/d/db/Constudproc.png`. (Accessed on 06.05.2023).

[11] Wikimedia Commons. Cone-fundamentals-with-srgb-spectrum, 2009. URL https://upload.wikimedia.org/wikipedia/commons/0/04/Cone-fundamentals-with-srgb-spectrum.svg. (Accessed on 06.05.2023).

[12] Wikimedia Commons. Action potential, 2013. URL https://upload.wikimedia.org/wikipedia/commons/9/95/Action_Potential.gif. (Accessed on 06.05.2023).

[13] Wikimedia Commons. Human visual pathway, 2015. URL https://upload.wikimedia.org/wikipedia/commons/b/bf/Human_visual_pathway.svg. (Accessed on 06.05.2023).

[14] Wikimedia Commons. Eeg 10-10 system with additional information, 2020. URL https://upload.wikimedia.org/wikipedia/commons/f/fb/EEG_10-10_system_with_additional_information.svg. (Accessed on 06.05.2023).

[15] Ciaran Cooney, Attila Korik, Raffaella Folli, and Damien Coyle. Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech eeg. *Sensors*, 20(16):4629, 2020.

[16] Anders M Dale and Martin I Sereno. Improved localizadon of cortical activity by combining eeg and meg with mri cortical surface reconstruction: a linear approach. *Journal of cognitive neuroscience*, 5(2):162–176, 1993.

[17] Anders M Dale, Bruce Fischl, and Martin I Sereno. Cortical surface-based analysis: I. segmentation and surface reconstruction. *Neuroimage*, 9(2):179–194, 1999.

[18] R Grave de Peralta-Menendez and Sara L Gonzalez-Andino. A critical analysis of linear inverse solutions to the neuroelectromagnetic inverse problem. *IEEE Transactions on Biomedical Engineering*, 45(4):440–448, 1998.

[19] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[20] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

[21] Christophe Destrieux, Bruce Fischl, Anders Dale, and Eric Halgren. Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature. *Neuroimage*, 53(1): 1–15, 2010.

[22] Bradley J Edelman, Bryan Baxter, and Bin He. Eeg source imaging enhances the decoding of complex right-hand motor imagery tasks. *IEEE Transactions on Biomedical Engineering*, 63(1): 4–14, 2015.

[23] Fatemeh Fahimi, Zhuo Zhang, Wooi Boon Goh, Tih-Shi Lee, Kai Keng Ang, and Cuntai Guan. Inter-subject transfer learning with an end-to-end deep convolutional neural network for eeg-based bci. *Journal of neural engineering*, 16(2):026007, 2019.

[24] Tao Fang, Zuoting Song, Gege Zhan, Xueze Zhang, Wei Mu, Pengchao Wang, Lihua Zhang, and Xiaoyang Kang. Decoding motor imagery tasks using esi and hybrid feature cnn. *Journal of Neural Engineering*, 19(1):016022, 2022.

[25] Bruce Fischl. Freesurfer. *NeuroImage*, 62(2):774–781, 2012. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2012.01.021. URL `https://www.sciencedirect.com/science/article/pii/S1053811912000389`. 20 YEARS OF fMRI.

[26] S P Fitzgibbon, D MW Powers, K J Pope, and C R Clark. Removal of eeg noise and artifact using blind source separation. *Journal of Clinical Neurophysiology*, 24(3):232–243, 2007.

[27] Simen Fløtaker, Marta Molinas, and Andres Soler. Deep learning for classification of primary color responses in eeg signals using source reconstruction. Project work at NTNU, 2022.

[28] Manfred Fuchs, Jörn Kastner, Michael Wagner, Susan Hawes, and John S Ebersole. A standardized boundary element method volume conductor model. *Clinical neurophysiology*, 113(5):702–712, 2002.

[29] Noha H Ghanem, Ahmed S Eltrass, and Nour H Ismail. Investigation of eeg noise and artifact removal by patch-based and kernel adaptive filtering techniques. In *2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 1–5. IEEE, 2018.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[31] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267): 1–13, 2013. doi: 10.3389/fnins.2013.00267.

[32] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S Hämäläinen. Mne software for processing meg and eeg data. *Neuroimage*, 86:446–460, 2014.

[33] Matti Hämäläinen, Riitta Hari, Risto J Ilmoniemi, Jukka Knuutila, and Olli V Lounasmaa. Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain. *Reviews of modern Physics*, 65(2):413, 1993.

[34] Riitta Hari and Aina Puce. *Meg-EEG Primer*. Oxford University Press, 2017.

[35] Bin He and Jie Lian. Electrophysiological neuroimaging. *Neural engineering*, pages 221–261, 2005.

[36] Tommy Hinks, Hamish Carr, Linh Truong-Hong, and Debra F Laefer. Point cloud data conversion into solid models via point-based voxelization. *Journal of Surveying Engineering*, 139(2):72–83, 2013.

[37] Yu Huang, Lucas C Parra, and Stefan Haufe. The new york head—a precise standardized volume conductor model for eeg source localization and tes targeting. *NeuroImage*, 140:150–162, 2016.

[38] Ramy Hussein, Hamid Palangi, Rabab Ward, and Z Jane Wang. Epileptic seizure detection: A deep learning approach. *arXiv preprint arXiv:1803.09848*, 2018.

[39] Alice F Jackson and Donald J Bolger. The neurophysiological bases of eeg and eeg measurement: A review for the rest of us. *Psychophysiology*, 51(11):1061–1071, 2014.

[40] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation*, 18(4): 602–622, 2013.

[41] Herbert H Jasper. Ten-twenty electrode system of the international federation. *Electroencephalogr. Clin. Neurophysiol.*, 10:371–375, 1958.

[42] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2021.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[44] Ioannis Kouretas and Vassilis Paliouras. Simplified hardware implementation of the softmax activation function. In *2019 8th international conference on modern circuits and systems technologies (MOCAST)*, pages 1–4. IEEE, 2019.

[45] Ichiro Kuriki. Emergence and separation of color categories: An nirs study in prelingual infants and a k-means analysis on japanese color-naming data. *Current Opinion in Behavioral Sciences*, 30:21–27, 2019.

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

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

[48] Dae-Hyeok Lee, Ji-Hoon Jeong, Hyung-Ju Ahn, and Seong-Whan Lee. Design of an eeg-based drone swarm control system using endogenous bci paradigms. In *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–5. IEEE, 2021.

[49] Sabine Leske and Sarang S Dalal. Reducing power line noise in eeg and meg data via spectrum interpolation. *Neuroimage*, 189:763–776, 2019.

[50] Ming-ai Li and Zi-wei Ruan. A novel decoding method for motor imagery tasks with 4d data representation and 3d convolutional neural networks. *Journal of Neural Engineering*, 18(4): 046029, 2021.

[51] Nikos K Logothetis. Vision: a window on consciousness. *Scientific American*, 281(5):68–75, 1999.

[52] Sara L Ludvigsen, Emma H Buøen, Andres Soler, and Marta Molinas. Searching for unique neural descriptors of primary colours in eeg signals: a classification study. In *Brain Informatics: 14th International Conference, BI 2021, Virtual Event, September 17–19, 2021, Proceedings 14*, pages 277–286. Springer, 2021.

[53] Christoph M Michel and Denis Brunet. Eeg source imaging: a practical review of the analysis steps. *Frontiers in neurology*, 10:325, 2019.

[54] Luis Alfredo Moctezuma and Marta Molinas. Eeg channel-selection method for epileptic-seizure classification based on multi-objective optimization. *Frontiers in neuroscience*, 14:593, 2020.

[55] Standard Electrode Position Nomenclature. American electroencephalographic society guidelines for. *Journal of clinical Neurophysiology*, 8(2):200–202, 1991.

[56] Robert Oostenveld and Peter Praamstra. The five percent electrode system for high-resolution eeg and erp measurements. *Clinical neurophysiology*, 112(4):713–719, 2001.

[57] Roberto D Pascual-Marqui, Dietrich Lehmann, Martha Koukkou, Kieko Kochi, Peter Anderer, Bernd Saletu, Hideaki Tanaka, Koichi Hirata, E Roy John, Leslie Prichep, et al. Assessing interactions in the brain with exact low-resolution electromagnetic tomography. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1952): 3768–3784, 2011.

[58] Roberto Domingo Pascual-Marqui et al. Standardized low-resolution brain electromagnetic tomography (sloreta): technical details. *Methods Find Exp Clin Pharmacol*, 24(Suppl D):5–12, 2002.

[59] Christophe Phillips, Michael D Rugg, and Karl J Friston. Systematic regularization of linear inverse solutions of the eeg source localization problem. *NeuroImage*, 17(1):287–301, 2002.

[60] Saim Rasheed, Cristian Bonanomi, Davide Gadia, Alessandro Rizzi, Daniele Marini, et al. Eeg spectral analysis of visual evoked potential produced by rgb color stimuli. In *Atti della Sesta Conferenza Nazionale del Gruppo del Colore*, volume 6, pages 160–171, 2010.

[61] Isabelle A Rosenthal, Shridhar R Singh, Katherine L Hermann, Dimitrios Pantazis, and Bevil R Conway. Color space geometry uncovered with magnetoencephalography. *Current Biology*, 31 (3):515–526, 2021.

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

[63] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. Epic: An energy-efficient, high-performance gpgpu computing research infrastructure. *arXiv preprint arXiv:1912.05848*, 2019.

[64] 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.

[65] Matti Stenroos, Alexander Hunold, and Jens Haueisen. Comparison of three-shell and simplified volume conductor models in magnetoencephalography. *NeuroImage*, 94:337–348, 2014.

[66] Desney Tan and Anton Nijholt. *Brain-computer interfaces and human-computer interaction*. Springer, 2010.

[67] Luca Tonin, Robert Leeb, Aleksander Sobolewski, and J Del R Millán. An online eeg bci based on covert visuospatial attention in absence of exogenous stimulation. *Journal of neural engineering*, 10(5):056007, 2013.

[68] Mikko A Uusitalo and Risto J Ilmoniemi. Signal-space projection method for separating meg or eeg into components. *Medical and biological engineering and computing*, 35:135–140, 1997.

[69] Fang Wang, Sheng-hua Zhong, Jianfeng Peng, Jianmin Jiang, and Yan Liu. Data augmentation for eeg-based emotion recognition with deep convolutional neural networks. In *MultiMedia Modeling: 24th International Conference, MMM 2018, Bangkok, Thailand, February 5-7, 2018, Proceedings, Part II 24*, pages 82–93. Springer, 2018.

[70] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[71] Xinqiao Zhao, Hongmiao Zhang, Guilin Zhu, Fengxiang You, Shaolong Kuang, and Lining Sun. A multi-branch 3d convolutional neural network for eeg-based motor imagery classification. *IEEE transactions on neural systems and rehabilitation engineering*, 27(10):2164–2177, 2019.

Appendix A

List of Abbreviations

BCI brain-computer interface

ML machine learning

GA genetic algorithm

MOGA multiobjective genetic algorithm

NS non-dominated sorting

EEG electroencephalography

MRI magnetic resonance imaging

DNN deep neural network

BEM boundary element method

ROI region of interest

MEG magnetoencephalography

CNS central nervous system

AP action potential

CNN convolutional neural network

EOG electrooculography

AWGN additive white Gaussian noise

SR source reconstruction

TOI time of interest

MNE minimum norm estimate

SNR signal-to-noise ratio

RGB red, green, and blue

NN neural network

MI motor imagery

2D two-dimensional

4D four-dimensional

3D three-dimensional

ReLU rectified linear unit

4DDFM 4D dipole feature matrix

V1 visual area one

NTNU Norwegian University of Science and Technology,

GPU graphics processing unit,

CWT continuous wavelet transform

SSP signal-space projection

PTP peak-to-peak

VAL8 the eight validation subjects

Appendix B

Abstract for BCI Society Meeting

Decoding primary color responses in EEG signals with deep learning in the source space

Simen Fløtaker[1*], Andres Soler[1], Marta Molinas[1]
[1] *Norwegian University of Science and Technology, Trondheim, Norway*
[*] O. S. Bragstads Plass 7034, Trondheim, Norway. E-mail: simenpf@stud.ntnu.no

Introduction: The brain's response to visual stimuli of different colors might be used in a brain-computer interface (BCI) paradigm. Allowing the user to control certain elements in its environment by looking at corresponding signs of different colors could serve as an intuitive interface. This paper presents work on the development of a classifier for red, green, and blue (RGB) visual evoked potentials (VEPs) in recordings performed with electroencephalography (EEG).

Material, Methods and Results: The classifiers developed in this work were trained and tested on a dataset of primary colors (RGB) visual stimulation. The dataset contains 60-channel EEG recordings from 31 subjects. The RGB colors were displayed on a screen in front of the subjects for intervals of 1.3 seconds, in random order with 140 repetitions for each color. Three convolutional neural networks (CNNs) were explored for this classification task: A graph CNN (GCNN) [1], EEGnet [2], and deep convNet [3]. Intra-subject classifiers were developed for all 31 subjects. EEGnet and DeepCNN were trained in both electrode and source space. The best classifier, deep convNet using all electrodes, yielded an average accuracy of 77%. A previous study developing classifiers for the same dataset, using conventional machine learning, reported an average accuracy of 74.43% for a subset of subjects [4]. In this study, the same subset achieved an average accuracy of 84%.

Discussion: The results indicate that it is possible to distinguish between the primary color responses. The hyperparameters of the three networks employed in this work have been left unchanged (except for some modifications necessary for integration). Considering this, it is reasonable to assume that some tuning of these hyperparameters could yield better results. The classifiers were expected to perform better in source space than electrode space, however, this was in general not the case. This unexpected result could be attributed to the fact that all three neural networks were originally developed for use in electrode space.

Significance: The results of this work demonstrate that it is possible to classify between primary color responses in EEG recordings. The results also show that deep learning methods can be suitable alternatives to traditional machine learning for decoding primary color responses.

References:

[1] Neeraj Wagh and Yogatheesan Varatharajah. Eeg-gcnn: Augmenting electroencephalogram-based neurological disease diagnosis using a domain-guided graph convolutional neural network. In Emily Alsentzer, Matthew B. A. McDermott, Fabian Falck, Suproteem K. Sarkar, Subhrajit Roy, and Stephanie L. Hyland, editors, *Proceedings of the Machine Learning for Health NeurIPS Workshop*, volume 136 of *Proceedings of Machine Learning Research*, pages 367–378. PMLR, 11 Dec 2020.

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

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

[4] Sara L Ludvigsen, Emma H Buøen, Andres Soler, and Marta Molinas. Searching for unique neural descriptors of primary colours in eeg signals: a classification study. In *International Conference on Brain Informatics*, pages 277–286. Springer, 2021.

Appendix C

Paper for EMBC Conference

Primary color decoding using deep learning on source reconstructed EEG signal responses

Simen Fløtaker[1], Andres Soler[1] and Marta Molinas[1]

Abstract— **The brain's response to visual stimuli of different colors might be used in a brain-computer interface (BCI) paradigm, for letting a user control their surroundings by looking at specific colors. Allowing the user to control certain elements in its environment, such as lighting and doors, by looking at corresponding signs of different colors could serve as an intuitive interface. This paper presents work on the development of an intra-subject classifier for red, green, and blue (RGB) visual evoked potentials (VEPs) in recordings performed with an electroencephalogram (EEG). Three deep neural networks (DNNs), proposed in earlier papers, were employed and tested for data in source- and electrode space. All the tests performed in electrode space yielded better results than those in source space. The best classifier yielded an accuracy of 77% averaged over all subjects, with the best subject having an accuracy of 96%.**

Clinical relevance— **This paper demonstrates that deep learning can be used to classify between red, green and blue visual evoked potentials in EEG recordings with an average accuracy of 77%.**

I. INTRODUCTION

Brain-computer interfaces (BCIs) are systems built to let the user control devices with their brain activity. Such systems can be of great assistance for persons with physical disabilities, as an alternative to traditional systems, which often require physical interaction with the device. For a BCI to be implemented, it requires some form of measurement of brain activity. One common way of doing this is electroencephalography (EEG). EEG can be non-invasive and it can meet high real-time demands, making it a suitable component for a BCI [1].

After recording the brain activity, the BCI also has to interpret that data. This often involves classifying the data into a set of classes, each corresponding to a desired action of the BCI. Performing this classification is a crucial component of the BCI. Without a robust classification method, the actions performed by the BCI might be spurious, which is unacceptable for most control systems. Creating a robust classifier for any signal requires firstly identifying the features of the signal that are relevant to the task, and secondly performing a classification based on these features. Traditionally, feature extraction is done manually and the classification by machine learning (ML). Understanding what features are relevant for which tasks may require expertise in the field, and can be very difficult for novel tasks. Deep learning is an interesting alternative to traditional ML, as it learns both features and classification from data [2]. This facilitates finding novel features for any task, as well as lessening the need for expertise in the field.

For any BCI paradigm, one has to establish a set of brain activities the user should exert, each different enough to be distinguishable. Eliciting such signals may be nonintuitive and difficult for the user [1]. Moreover, Allison et al. [1] point out that the signals for different users might be very different, even if attempting to elicit the same brain activity. As humans, our response to colors is very well-trained, we can identify whether something is blue or red without thinking. Since color vision is such a primal part of human life and we easily distinguish between colors, it is natural to assume that color stimuli elicit distinguishable brain activity. A classifier able to separate what color a subject is looking at could be of use in a BCI. For instance, looking at signs of different colors could allow control of the user's environment, such as opening and closing doors, and turning on and off lights.

Several previous studies have explored classification of brain activity in subjects visually stimulated with red, green and blue (RGB) colors. One study achieved an accuracy of 58%, for a naive Bayes RGB classifier [3]. In [4], the same dataset used in this study was used to train and test machine learning classifiers. Their best results were obtained with a minimum distance to mean with geodesic filtering (Fg-MDM) Riemannian classifier, yielding an average accuracy of 74.48% per subject. In this study, in an attempt to better untangle the encoded colors, source reconstruction, a method for estimating the magnitude and location of neural activity in the brain from EEG signals, was used in combination with deep learning.

This paper is structured into four main sections: This introduction, material and methods, results, and discussion. The dataset used in this paper, the source reconstruction, and classification methods are all described in the section materials and methods. Finally, a conclusion is provided. A more exhaustive report on this work is available online [5].

II. MATERIAL AND METHODS

A. DATASET

The classifiers developed in this work were trained and tested on a dataset where the participants were exposed to primary colors (RGB). The colors were displayed on a screen in front of the participant for intervals of 1.3 seconds, in random order with 140 repetitions for each color. Between each repetition, a gray screen with a cross in the middle was displayed for a random interval of 1.3-1.6 seconds. This

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway. simenpf@stud.ntnu.no, andres.f.soler.guevara@ntnu.no, marta.molinas@ntnu.no

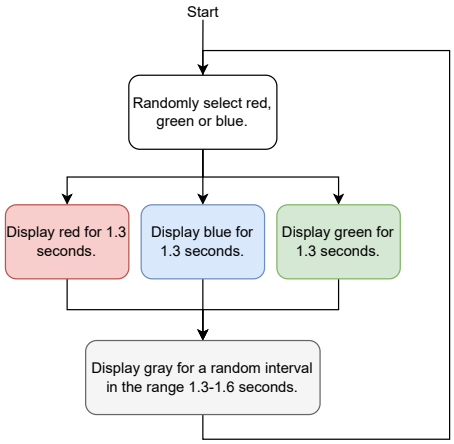


Fig. 1. The stimulus protocol during EEG recording.

protocol is illustrated in Fig. 1. The dataset consists of 60-channel EEG recordings during color presentation and structural MRI from 31 participants (10 females) with an average age of 28.8 (sd 7.4) years old, the participants had normal or corrected-to-normal vision without color impairments. The study was carried out in accordance with the Declaration of Helsinki and all participants provided their informed consent prior to participation. The study was approved by the Data Protection Authority (NSD, reference number 968653). The dataset was recorded at the Aalto NeuroImaging facility of Aalto University.

B. PREPROCESSING

The raw EEG recordings were preprocessed before being used to train and test the neural network classifiers. A notch filter of 50Hz was applied, in order to reject the interference from the powerline. A bandpass filter with the frequency range 0.1-45Hz was applied, in order to filter out frequencies not of interest. After the filtering, the data were downsampled to 200Hz. The recordings were split into separate epochs, one for each stimulus event. The epoch interval was chosen to be from -0.2 seconds before the stimulus to 1.25 seconds

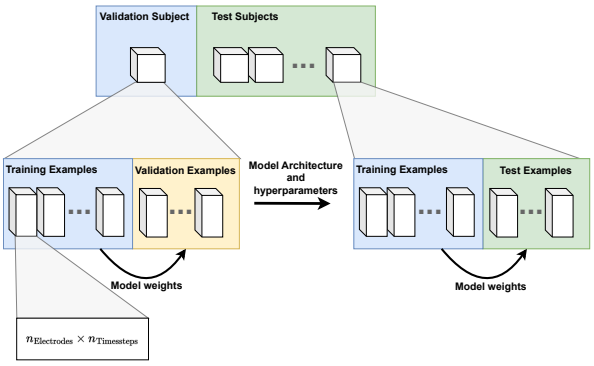


Fig. 2. The structure of the dataset. The validation subject is used to test different hyperparameters and architectures, and the best choice is selected for testing. The validation subject (sub-18), was randomly selected.

after. Baseline correction was applied, by calculating the mean of the 0.2 seconds of data from all channels and then subtracting these means from their respective channels throughout the whole epoch. Blinking artifacts were detected by a peak-finding algorithm. Epochs were discarded if a blink artifact was found within a 200ms interval centered around the onset of the stimulus. Signal-space projection (SSP) was employed to reduce the remaining blink artifacts. A criterion for the maximal acceptable peak-to-peak amplitude of 150 µV within each epoch was set. Thus, any epoch where the difference between the maximal and minimal value for at least one EEG channel was larger than 150 µV, was discarded. All preprocessing was done using MNE-Python [6].

C. SOURCE RECONSTRUCTION

The forward model was created using individual magnetic resonance imaging (MRI) data for each subject. Coregistration was manually performed for each subject, such that digitized electrode positions were best transformed into the MRI frame for the forward modeling. A boundary element model (BEM) was used to define the conduction of the brain volume. The sources were distributed on the surface of the white matter. The inverse problem was then solved with the dSPM method [7]. The source space data were aggregated into regions of interest (ROIs) using the automatic parcellation of the brain volume proposed by [8], resulting in 150 dipoles (75 in each hemisphere). In order to aggregate the sources in a ROI into one single value, two steps were taken: First, find the sign of the value for each source and select the sign most represented as the dominant sign. Flip all source values that do not have the dominant sign. Use the mean of all resulting source values as the ROI value. For most applications, the direction of the dipoles is not relevant [9], rather the amplitude is the important information. By flipping the signs, an average value of the amplitudes in the ROI is obtained, avoiding the cancellation of opposing signs during the averaging. The source reconstruction and parcellation was done using MNE-Python [6].

D. CLASSIFICATION METHODS

The work was focused on exploring EEG classification using source space representation. Since the data is of a similar nature in both source- and electrode space ($n_{\text{channels}} \times n_{\text{times}}$, with n_{channels} being the number of dipoles or electrodes for source- and electrode space respectively), the same neural network architectures can be used for both types of data, by modifying the input layer of the network. Three neural networks were employed in this work, using Keras [10]:

- **Shallow EEG-GCNN** (Graph Convolutional Neural Network), [11]
- **EEGNet** (Convolutional Neural Network), [12]
- **Deep ConvNet** (Convolutional Neural Network), [13]

All three networks were implemented with the same hyperparameters as in the papers they were originally proposed (except for some minor modifications necessary for integration). The graph convolutional neural network (GCNN) uses

an adjacency matrix to represent the data structure as a graph. Each dipole is treated as a node and the time series of that dipole is treated as its feature vector. The edges between node i and j is represented by the value of element a_{ij} in the adjacency matrix. In this work, if the ROIs represented by two nodes i and j share a border, a_{ij} was set to 1, otherwise it would be set to 0.

Only intra-subject classifiers have been explored in this work. Hence, each classifier had to be trained and tested on data from only one subject. To explore different configurations and hyperparameters before testing, the following routine was developed: The subjects are randomly divided into two groups: validation subject and test subjects. The dataset from the validation subject are each randomly segmented into its own training set and validation set. The datasets from the test subjects are each randomly segmented into a training set and a test set. Fig.2 depicts this structure. In practice, the segmentation within each subject was done with cross-validation. The purpose of separating into validation subject and test subjects is to explore different DNN architectures and hyperparameter configurations. Different configurations can be trained on the validation subject, and then evaluated on its validation set. Several configurations can be found by iterating this process. However, the accuracy found for the validation subject must be considered overly optimistic due to possible overfitting. So the final configuration is tested on the test subject, to see how well it generalizes.

Using only a subset of ROIs/electrodes for classification may be beneficial. Some regions of the brain may be more descriptive than others for the task at hand, thus using only those could reduce the overall signal-to-noise ratio in the data, and improve the classification. For instance, since the aim is to discriminate visual evoked potentials (VEPs), the occipital lobe might be such a region, as it is the brain area that interprets visual stimuli [14]. No optimal selection was performed to select a certain set of ROIs/electrodes in this work, but two different configurations were tested: using all ROIs/electrodes and using a selected set of ROIs/electrodes. In source space, the selected set was all 24 ROIs of the occipital lobe. In electrode space, the selected set was eight electrodes placed in the vicinity of the occipital lobe.

III. RESULTS

EEGNet and deep ConvNet (DCN) classifiers were trained and tested in both source- and electrode space. All these classifiers were developed for both channel configurations (all channels and selected channels). The GCNN was only built for source space using all ROIs. The results presented are the accuracies and standard deviations of intra-subject classifiers tested using a 5-fold cross-validation, those accuracies are presented in Table II. Two trends can be observed from the results: DCN performs better than EEGNet, and electrode space classifiers tend to perform better than source space classifiers. For all classifiers, especially those performing well, there is a noticeable difference between the best and the worst-performing subjects. This can be partially explained by some subjects not having correct behavior during the EEG recording. The subjects were observed during the recording, and notes were taken of some subjects being sleepy or moving excessively. The results show that these subjects tend to perform worse than the average. In addition, for one subject two of the EEG channels located in the vicinity of the occipital lobe did not function correctly and delivered no signal. This subject was among the worst-performing subjects in all tests. In a previous study using the same dataset [4], a choice was made to leave out a set of subjects. The motivation for leaving out these subjects was a set of requirements for a session to be allowed in their study, such as correct behavior of the subjects and no flat channels on the visual cortex. For the same subset of subjects used in this previous study, the best results in this paper (using deepConvnet with all electrodes) yield an average accuracy of 84%. Table I compares the results of this study to those reported in [4].

TABLE I
CLASSIFICATION RESULTS USING THE SAME SUBJECT SUBSET AS IN [4]

	This study		**Previous study [4]**
	Electrode space	**Source Space**	
Best acc.	0.96	0.87	0.93
Average acc.	0.84	0.58	0.75
Average std.	0.04	0.05	0.08
Worst acc.	0.66	0.38	0.54

IV. DISCUSSION

All average performances of EEGNet and DCN were better in electrode space than source space, regardless of electrode and ROI selection. This was not the expected result, seeing as the source space representation theoretically has a higher spatial resolution [9], and thus different conditions should be easier to discriminate. One possible explanation for this unexpected result is that both networks were developed for electrode space classification. Although both representations have a similar nature, it is not necessarily the case that a given DNN architecture is equally suitable for source- and electrode space. When averaging the sources much of the spatial resolution might be lost. It is effectively a spatial downsampling of the source space. Thus, some of the advantages gained in spatial resolution, by using source reconstruction, may be lost. As mentioned, the use of only some specific ROIs may serve as a method for improving classification. This method also has the advantage of reducing data dimensionality. Instead of reducing the data size by averaging into different regions, one could rather select a subset of sources from the entire set of sources (ca. 8000). By selecting certain regions expected or demonstrated to be relevant for decoding color stimuli, one could keep a high spatial resolution in those areas, while also reducing the data size.

The architectures evaluated in this study have been used with their original hyperparameters, the results also serve as an example that certain DNNs can be applicable across different tasks. One of the concerns regarding deep learning,

TABLE II

CLASSIFICATION RESULTS

Classifier	Accuracy best	Accuracy average	SD average	Accuracy worst
DCN, source space (selected, 24)	0.87	0.54	0.05	0.38
DCN, source space (all, 150)	0.79	0.53	0.05	0.43
EEGNet, source space (selected, 24)	0.71	0.48	0.05	0.37
EEGNet, source space (all, 150)	0.68	0.53	0.05	0.42
GCNN, source space (all, 150)	0.47	0.36	0.05	0.29
DCN, electrode space (selected, 8)	**0.96**	**0.77**	**0.04**	**0.46**
DCN, electrode space (all, 60)	0.95	0.72	0.04	0.37
EEGNet, electrode space (selected, 8)	0.92	0.67	0.05	0.41
EEGNet, electrode space (all, 60)	0.88	0.59	0.06	0.34

raised in [2], is the often unjustified selection of parameters in DNNs, making it difficult to rule out that some tuning based on the test set has occurred. That EEGNet and DCN, with their original hyperparameters, classify with higher accuracy than previous studies on RGB stimuli, is further evidence that these architectures are suitable for EEG decoding, and that their parameters are not overfitted to the test sets of the original papers

There are several reasons to believe that further work can achieve better performance than that reported in this project. Both EEGNet and DCN performed reasonably well, however, no modifications have been done to the design or hyperparameters of these architectures. It is reasonable to assume that these architectures could be tailored more specifically to the task of classifying RGB stimuli, and thus achieve better results. Such tailoring could for instance involve testing different numbers and widths of layers in the model.

The results show that choosing a subset of channels in the vicinity of the occipital lobe does yield an increase in performance. In light of this, it seems natural that there is an optimal subset of channels to use. This should be explored by employing a structured search for an optimal channel subset. Not only could this help develop a better-performing classifier, but choosing a subset of channels would also result in less data and fewer electrodes needed. This would make the classifier more applicable to a BCI, where few electrodes and low computational cost are important factors.

Variability of light conditions and color tones were not considered in this study. Towards a BCI implementation the variability of these parameters should taken into account, further studies should clarify the color classification in more naturalistic scenarios. The color vision impairment can be a limitation of the usability of a BCI based on colors, future evaluations on color blind participants could help to clarify the boundaries of this approach.

V. CONCLUSIONS

In conclusion, the results reported in this study suggest that deep learning can be a suitable approach for classifying RGB stimuli. All architectures employed in this project have been implemented with minimal adaption to the task at hand. Thus, the level of accuracy of both EEGNet and DCN suggests that some DNNs can be suitable across different tasks. Moreover, it is reasonable to believe that modifying these architectures for the classification of RGB responses would yield better results.

REFERENCES

[1] B. Z. Allison, C. Neuper, D. S. Tan, and A. Nijholt, "Could anyone use a BCI?" *Human-Computer Interaction Series*, pp. 35–54, 2010.

[2] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update," *Journal of Neural Engineering*, vol. 15, p. 031005, 4 2018.

[3] S. H. Åsly, L. A. Moctezuma, M. Molinas, and M. Gilde, "Towards eeg-based signals classification of RGB color-based stimuli." in *GB-CIC*, 2019.

[4] S. L. Ludvigsen, E. H. Buøen, A. Soler, and M. Molinas, "Searching for unique neural descriptors of primary colours in EEG signals: A classification study," *Lecture Notes in Computer Science*, vol. 12960 LNAI, pp. 277–286, 2021.

[5] S. P. Fløtaker, M. Molinas, and A. Soler, "Deep learning for classification of primary color responses in EEG signals using source reconstruction," 2022. [Online]. Available: https://www.researchgate.net/publication/367346031_Deep_learning_for_classification_of_primary_color_responses_in_EEG_signals_using_source_reconstruction

[6] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. Hämäläinen, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience*, vol. 7, no. 267, pp. 1–13, 2013.

[7] A. M. Dale, A. K. Liu, B. R. Fischl, R. L. Buckner, J. W. Belliveau, J. D. Lewine, and E. Halgren, "Dynamic statistical parametric mapping: Combining fMRI and MEG for high-resolution imaging of cortical activity," *Neuron*, vol. 26, pp. 55–67, 4 2000.

[8] C. Destrieux, B. Fischl, A. Dale, and E. Halgren, "Automatic parcellation of human cortical gyri and sulci using standard anatomical nomenclature," *NeuroImage*, vol. 53, pp. 1–15, 10 2010.

[9] C. M. Michel and D. Brunet, "EEG source imaging: A practical review of the analysis steps," *Frontiers in Neurology*, vol. 10, p. 325, 4 2019.

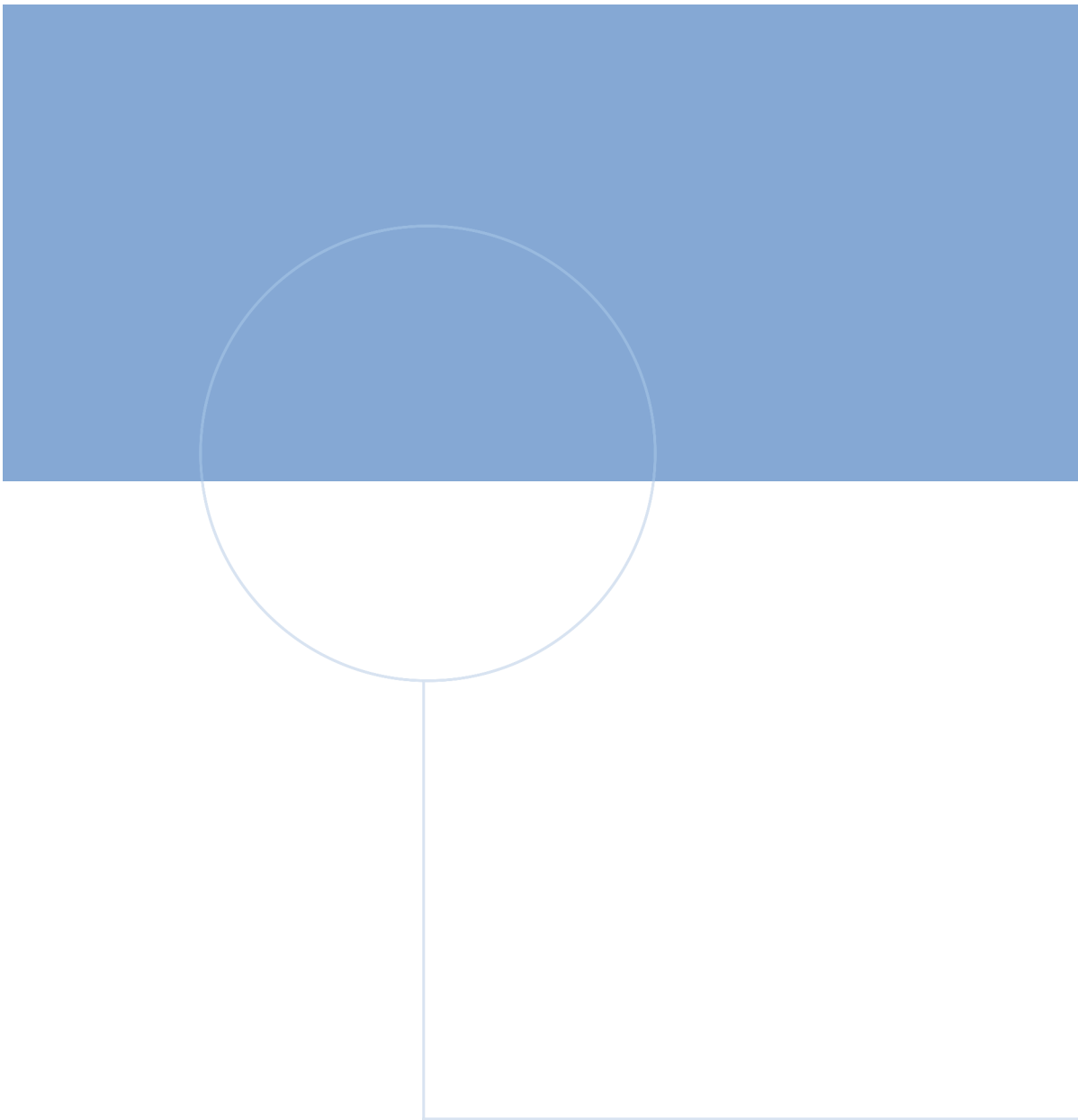
[10] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[11] N. Wagh and Y. Varatharajah, "EEG-GCNN: Augmenting electroencephalogram-based neurological disease diagnosis using a domain-guided graph convolutional neural network." *Proceedings of the Machine Learning for Health NeurIPS Workshop*, vol. 136, p. 367–378, 2020.

[12] L. Xu, M. Xu, Z. Ma, al, B. Zang, Y. Lin, Z. Liu, V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "EEGNet: a compact convolutional neural network for eeg-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 15, p. 056013, 7 2018.

[13] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for EEG decoding and visualization," *Human Brain Mapping*, vol. 38, no. 11, pp. 5391–5420, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.23730

[14] M. Bear, B. Connors, and M. Paradiso, *Neuroscience: Exploring the Brain, Enhanced Edition*. Jones & Bartlett Learning, 2020.



NTNU

Norwegian University of
Science and Technology