Darmstadt University of Applied Sciences

– Faculty of Mathematics and Natural sciences –
– Faculty of Computer Science –

EEG-Based Eye Tracking for Consumer-Grade BCIs: Evaluation of Different Approaches with a Focus on Functional Data Analysis

Submitted in partial fulfillment of the requirements for the degree of

Master of Science (M.Sc.)

by

Tiago Vasconcelos Afonso

Matriculation number: 1122173

Supervisor	:	Prof. Dr. Florian Heinrichs
Co-Supervisor	:	Prof. Dr. Timo Schürg

Date of registration	:	April 24, 2024
Date of submission	:	October 9, 2024

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 09.10.2024

Tiago Vasconcelos Afonso

ABSTRACT

EEG-based eye tracking (ET) is an emerging application of brain-computer interfaces (BCIs). EEG, typically used for recording brain activity, also captures eye movement artifacts that can be leveraged for tracking eye movement. This approach offers advantages over traditional camera-based ET, especially in poor lighting or with closed eyes. It also reduces hardware requirements and simplifies experiments that require both eye movement and EEG data. Despite its potential, EEG-based ET research has been limited to laboratory settings and expensive equipment. This thesis investigates its feasibility using consumer-grade hardware under more realistic conditions, filling a gap in current research. For this, methods based on Functional Data Analysis (FDA) are explored, a framework well-suited to the continuous nature of eye movement data, but one that has yet to be applied in this context.

The contents of this thesis are primarily structured around two main points: the creation of a novel EEG-ET dataset using consumer-grade hardware and the evaluation of models on this dataset, including the SpatialFilterCNN, a deep learning model and BMOTE, a white-box model that incorporates the physical processes of the eye through explicit modeling, as well as newly developed Functional Neural Networks (FNNs) based on FDA.

A major outcome is the introduction of the currently largest known EEG-ET dataset using consumer-grade hardware, with 11 hours and 45 minutes of continuous data from 113 participants. Results showed that EEG-based eye tracking is feasible with consumer-grade hardware, with the SpatialFilterCNN achieving a Mean Euclidean Distance (MED) of 109.0 mm and 99.38 mm on the most challenging tasks of the dataset and FNNs achieving a MED of 127.5 mm and 100.8 mm on the same tasks, in both cases beating the mean baseline. The BMOTE displayed random performance on all tasks. Evaluating the FNNs on the related EEGEyeNet dataset revealed state-of-the-art performance, though similar results were obtained from non-functional models, leaving the role of functional layers inconclusive and suggesting further research is needed.

ZUSAMMENFASSUNG

EEG-basiertes Eye-Tracking (ET) ist ein vielversprechendes neues Anwendunggebiet von Brain-Computer-Interfaces (BCIs). EEG, das normalerweise zur Erfassung der Gehirnaktivität verwendet wird, zeichnet auch elektrische Artefakte der Augenbewegungen auf, die für das Tracking dieser Bewegungen genutzt werden können. Dieser Ansatz bietet gegenüber herkömmlichem kamerabasiertem Eye-Tracking mehrere Vorteile, insbesondere unter schwierigen Bedingungen wie schlechten Lichtverhältnissen oder geschlossenen Augen. Außerdem können die Hardwareanforderungen reduziert und Versuchsaufbauten, die sowohl Augenbewegungen als auch EEG-Daten erfassen, vereinfacht werden. Trotz des Potenzials war EEG-basiertes Eye-Tracking bislang auf Anwendungen mit teurer Ausrüstung und unter Laborbedingungen beschränkt. Diese Arbeit untersucht die Machbarkeit des EEG-basierten Eye-Tracking mit kostengünstiger Hardware unter altagsnahen Bedingungen und schließt damit eine Lücke in der aktuellen Forschung. Zu diesem Zweck werden Methoden basierend auf der Funktionalen Datenanalyse (FDA) untersucht, einem Rahmenwerk, das besonders gut für kontinuierliche Daten wie Augenbewegungen geeignet ist, jedoch bisher in diesem Zusammenhang nicht angewendet wurde.

Der Inhalt dieser Arbeit gliedert sich hauptsächlich in zwei Punkte: die Erstellung eines EEG-ET-Datensatzes auf Basis von kostengünstiger Hardware und die Evaluierung von Modellen auf diesem Datensatz, darunter das SpatialFilterCNN, ein Deep-Learning-Modell, und BMOTE, ein White-Box-Modell, das die physikalischen Prozesse des Auges explizit modelliert, sowie neu entwickelte Funktionale Neuronale Netze (FNNs) auf Basis von FDA.

Ein wesentliches Ergebnis dieser Arbeit ist die Einführung des derzeit größten bekannten EEG-ET-Datensatzes, der mit verbrauchertauglicher Hardware erstellt wurde. Dieser umfasst 11 Stunden und 45 Minuten an kontinuierlichen Aufzeichnungen von 113 Teilnehmern. Die Ergebnisse zeigten, dass EEG-basiertes Eye-Tracking mit kostengünstiger Hardware möglich ist. Das SpatialFilterCNN erzielte dabei eine mittlere euklidische Distanz (MED) von 109.0 mm und 99.38 mm bei den anspruchsvollsten Aufgaben des Datensatzes, während die FNNs eine MED von 127.5 mm und 100.8 mm erreichten. Das BMOTE-Modell hingegen zeigte in allen Aufgaben zufällige Ergebnisse. Eine zusätzliche Evaluation der FNNs auf dem verwandten EEGEyeNet-Datensatz führte zu den bisher besten Ergebnissen auf diesem Datensatz, wobei ähnliche Resultate auch mit nicht-funktionalen Modellen erreicht wurden. Somit kann der Einfluss der funktionalen Schichten nicht abschließend geklärt werden, was weitere Forschungen erforderlich macht.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

BCI Brain-Computer Interface
BMOTE Battery Model of the Eye
CNN Convolutional Neural Network
CPU Central Processing Unit
CSV Comma Separated Values
EEG Electroencephalography
ELU Exponential Linear Unit
EOG Electrooculography
ET Eye Tracking
FDA Functional Data Analysis
fMRI Functional Magnetic Resonance Imaging
FNN Functional Neural Network
GPU Graphics Processing Unit
HD High Definition
MAE Mean Absolute Error
MED Mean Euclidean Distance
MEG Magnetoencephalography
MRS Magnetic Resonance Spectroscopy
MSE Mean Squared Error
PET Positron Emission Tomography
RAM Random Access Memory
ReLU Rectified Linear Unit
REM Rapid Eye Movement
RMSE Root Mean Squared Error
SARIMA Seasonal Autoregressive Integrated Moving Average
TPE Tree-structured Parzen Estimator
XDF Extensible Data Format

Part I

THESIS

1 INTRODUCTION

In the future, Brain-Computer Interfaces (BCIs) might offer us the fascinating possibility of restoring or augmenting our sensory perception, such as allowing a blind person to see [29], or even converting our thoughts into text [45], enabling real telepathic communication. Today this technology already holds enormous potential for a wide range of applications, from assisting people with disabilities [4, 6] to controlling devices in smart homes [27]. However, there is still a long way to go before BCIs are reliable and easy enough to use such that a wide spread adoption is possible. An important milestone on this path could be eye tracking based on Electroencephalography (EEG). To this end, the collection of large amounts of data is necessary to develop and evaluate new models. This is addressed in this work by creating the Consumer EEG-ET dataset. Additionally, new models based on Functional Data Analysis (FDA) are developed, evaluated, and compared to existing models on the new dataset.

1.1 MOTIVATION

EEG-based eye tracking is emerging as a promising application of BCIs [12, 15, 23, 40]. While EEG is typically used to record the electrical activity of the brain, it also captures eye movement artifacts due to the inherent electrical activity of the eyes. Although these signals are usually considered noise in other BCI applications and are often removed [10], they can be effectively used to track eye movements. These signals are also easier to decode than brain activity, as they are not complicated by the complexity and noise associated with brain signal interpretation. In addition, achieving reliable and accurate eye tracking using EEG technology could significantly enhance existing consumer BCIs, opening up a wide range of new applications.

Apart from the potential for BCI applications, EEG-based eye tracking is an interesting alternative to eye tracking in its own right, offering several advantages over camera-based eye tracking, which is the predominant method used for eye tracking today. Compared to camera-based methods, EEG-based eye tracking works even in poor lighting conditions or with closed eyes, such as during sleep. Additionally, performing experiments that require both eye movement and EEG data are more practical with EEG-based eye tracking, as no additional hardware is required. Such experiments are quite common in cognitive neuroscience or psychology research. The same holds for consumer BCIs, where the integration of eye tracking is more practical with EEG-based eye tracking than with camera-based eye tracking. However, the development of EEG-based eye-tracking is still lagging behind. Previous results have been achieved under laboratory conditions with expen-

sive hardware. These systems are impractical for everyday use (e.g., because a gel must be applied between the electrodes and the scalp before use).

Finally, in addition to their applications, joint EEG and eye tracking data is itself of interest to the field of FDA. Before many FDA methods can be applied, a suitable transformation of time is necessary to align the data, which is called "registering a function". This is, however only possible if meaningful features can be identified in the data or if the onset of the event of interest is externally triggered. In the case of EEG data, where the event of interest is often a self-paced action and the signal is noisy and complex, curve registration is infeasible. Therefore, EEG-based eye tracking data may be of interest for developing and testing new methods that work with unregistered data.

All of this motivates the development of a new dataset where EEG and eye tracking data are collected simultaneously. Even though such datasets already exist, they are based on expensive hardware and are not suitable for consumer BCIs. Also those datasets mostly only contain data where the eyes are either fixating or jumping to a new location. Smooth pursuit data, which is especially interesting for FDA, is missing in current datasets. Furthermore, with such a dataset at hand, a new model based on FDA can be developed and evaluated.

1.2 GOAL

A key objective of this work is to create the Consumer EEG-ET dataset, a dataset that contains EEG and eye tracking data collected simultaneously. This dataset is intended to be comparable in structure to the already existing EEGEyeNet dataset [23] but will be based on consumer hardware with dry electrodes. Since no such dataset currently exists and the feasibility of reconstructing gaze position from EEG data recorded with consumer hardware is uncertain, the dataset will be divided into different "difficulty levels". The first level will include data where eye movements occur along a single axis, either up, down, left, or right. The second level will then include data where eye movements occur along both axes. In order to assess the quality of the dataset, the performance of two existing models [2, 15] will be evaluated on the Consumer EEG-ET dataset. For this, both methods will be reimplemented and evaluated on the new dataset. Additionally, new models based on FDA will be developed and also evaluated on the new dataset. FDA is a promising approach for EEG-based eye tracking, as smooth movements are naturally represented as functions and can be modeled in this framework with much fewer parameters than other methods, which might make it more robust to noise. However, FDA has not been used for eye movement data yet. This work aims to close this gap and to evaluate the performance of FDA on EEG-based eye tracking data. For this purpose, the new model will also be evaluated on already existing EEGEyeNet dataset.

1.3 STRUCTURE

This work is structured as follows: In Chapter 2 we provide all necessary background information to understand the work. This includes an introduction to Electroencephalography (EEG) and Eye Tracking (ET) and a more detailed explanation on how EEG is actually used for eye tracking. We also provide an overview of related work and explain the methods "SpatialFilter-CNN" and the Battery Model of the Eye (BMOTE) that are reimplemented in this work and explained in detail. Finally, we introduce the basics of Functional Data Analysis (FDA). In Chapter 3, we introduce the Consumer EEG-ET dataset. We explain how the dataset was collected, key facts like the number of participants, the hardware used, *etc.* We also explain the stimuli presentation and experimental paradigms in detail. Finally, we explain the structure in which the dataset will be made available and provide guidelines on how to benchmark models on the dataset, ensuring that the dataset can be used by other researchers. In Chapter 4 we present the new FDA methods that are evaluated in the next chapter as well as any preprocessing or other methods that are used in the evaluation. We also explain how the reimplemented models were validated, briefly explain the experimentation setup and what the evaluation metrics are. Following this, in Chapter 5 we present and discuss the results of the evaluation in the following order:

1. The results of the reimplemented models on the Consumer EEG-ET dataset
2. The results of the new FDA model on the Consumer EEG-ET dataset
3. The results of the new FDA model on the EEGEyeNet dataset

Finally, in Chapter 6 we summarize and discuss the implications of the results. We also discuss the limitations of the work and give an outlook on future work and finish with a short conclusion.

2 BACKGROUND

This chapter provides the necessary background for the work presented in this thesis. It starts by introducing the fundamentals of Electroencephalography (EEG) and Eye Tracking (ET) in Section 2.1. Section 2.2 presents related work in the field of EEG-based ET. The chapter concludes with Section 2.3 and Section 2.4 where the basics of Functional Data Analysis (FDA) are introduced and the concept of Functional Neural Networks (FNNs) is explained.

2.1 EEG AND ET FUNDAMENTALS

Electroencephalography (EEG) is a non-invasive method to measure the electrical activity of the brain. It is based on the recording of electrical potentials generated by the brain's neurons. The electrical activity is measured by placing electrodes on the scalp of the subject. The electrodes are connected to an amplifier, which amplifies the electrical signals. Various fields such as medicine, psychology, and neuroscience use EEG for different purposes. It is used to diagnose epilepsy [6], sleep disorders [4], and brain death [26], or to monitor the effect of sedative/anesthesia in patients in medically induced coma [21]. In psychology, EEG is used to study cognitive processes such as attention [24], memory [8], and language [36]. In neuroscience, EEG is used to study brain function and brain disorders [1].

The main limitation of EEG is its spatial resolution. An action potential of a single neuron is too weak to be detected by an electrode, a signal thus always represents the sum of the electrical activity of a large number of neurons that are aligned with the electrode. Furthermore, neural activity that occurs below the upper layers of the brain is poorly measured by EEG. Another limitation is a large amount of noise that is present in the signal. The noise can come from the environment (*e.g.* electrical devices, lights, *etc.*) or from the subject (*e.g.* muscle activity, eye movements, *etc.*).

Despite the drawbacks mentioned above, EEG has several advantages. It is easy to use, and is relatively inexpensive and compact compared to other methods such as Functional Magnetic Resonance Imaging (fMRI), Positron Emission Tomography (PET), Magnetic Resonance Spectroscopy (MRS), or Magnetoencephalography (MEG), which require bulky and immobile equipment. For example, MEG requires equipment consisting of liquid helium cooled detectors at a total cost of several million dollars [19] and fMRI requires the use of a 2-3 meter long hollow cylindrical magnet [39]. In addition, both only work in magnetically shielded rooms. Furthermore, EEG has a high temporal resolution, allowing the measurement of brain activity with millisecond precision.

Eye Tracking

Eye Tracking (ET) is a method to measure the gaze of a person. It is used in various fields such as marketing, psychology, and Human Computer Interaction. For example, to study consumer behavior in marketing [16] or to study attention, perception, and memory in psychology [28]. Eye tracking is used to enable people with disabilities to interact with computers or in Augmented Reality or Virtual Reality environments [13]. With products like *Meta Quest* or *Apple Vision Pro* being released in the last years, eye tracking is becoming more and more relevant in the consumer market.

The dominant approaches in eye tracking are video-based. A camera is focused on one or both eyes and the eye movement is recorded. From this data, the person's gaze can be calculated. Most modern eye trackers use infrared/near infrared light to create bright reflections on the cornea making it particularly easy to detect the position and orientation of the eye. It also allows tracking in very bright and dark lighting conditions. Eye trackers can be head-mounted or remote. Remote eye trackers are placed in front of the subject and either require the head to be still or use a head tracking system to compensate for head movements. Head-mounted eye trackers move with the head and thus automatically compensate for head movements. Due to the reduced accuracy at high off-axial angles, camera-based eye trackers need to be placed within your field of vision, which means they will always block a portion of it. This is especially true for head-mounted eye trackers. Prices for hardware specifically designed for camera-based eye tracking range from \$300 [42] to \$25,000 [43]. It is also possible to use a regular webcam for eye tracking. In this case hardware starting at \$50 might be sufficient, but the accuracy will be reduced compared to special hardware. Additionally, software is needed to process the video stream and calculate the gaze. For this there are many solutions available, such as RealEye, Eyezag, WebGazer.js, Eyeware beam, EYEVIDO, GazeRecorder and many more, of which some are free and some are paid. The software used in this thesis is the free software GazePointer from GazeRecorder.

The accuracy of an eye tracker is typically measured in degrees of visual angle. Conceptually the visual angle is the angle that two points on a screen make with the eye, as shown in Figure 2.1. Given the position of two points on the screen and the distance between the eye and the screen, the visual angle can be calculated. The easiest way is probably using the geometric definition of the dot product

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos(\theta),$$

where $\mathbf{a}$ and $\mathbf{b}$ are the vectors from the eye to the two points and θ is the angle between them. Solving for θ leads to

$$\theta = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}\right).$$

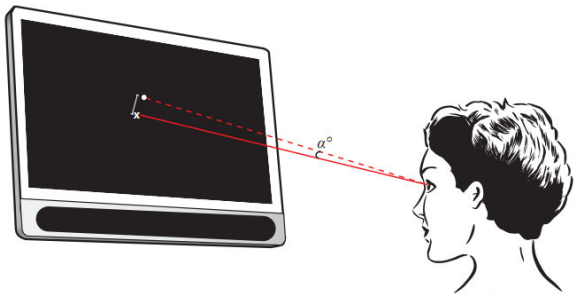

Figure 2.1: The visual angle is the angle that two points on a screen make with the eye. Image taken from [41]

Given a set of targets and corresponding predictions of the eye tracker, the accuracy can then be calculated by taking the average of all visual angles between the targets and predictions. Note that in general the visual angle depends on the specific position of both points not just their distance $\|\mathbf{a} - \mathbf{b}\|_2$. For example, placing two points near the middle of the screen will result in a larger visual angle than when the points are placed at the same distance near the edges of the screen, due to the greater distance from the screen. This means that one cannot simply convert between accuracy based on distance and accuracy in degrees.

High-end camera-based eye trackers can be accurate to within 0.25°-0.50° [35]. Using a webcam, the accuracy is generally greater than 1°. In a report comparing the accuracy of the GazeRecorder software based on images from webcams to the SMI RED 250 eye tracker, a standard infrared light camera-based eye tracker, the accuracy of the webcams was found to be less than 0.9°-1°, which is comparable to the SMI RED 250 device [11]. In this thesis we could not reproduce these results. The accuracy of the webcam running the GazePointer software was found to be much worse than 1°.

EEG based Eye Tracking

In addition to camera-based eye tracking, it is also possible to track eye movements using electrodes placed around the eyes. This method is called Electrooculography (EOG). It is much less commonly used than camera-based eye tracking. Its main applications are in sleep research, where it can be used to detect Rapid Eye Movement (REM) sleep [17], and in the diagnosis of certain eye diseases [7]. The method is based on the fact that the cornea of the eye is positively and the retina negatively charged making the eye a dipole. Thus, when the eye moves, electrodes placed around the eyes will record a change in voltage that is proportional to the eye movement. This is depicted in Figure 2.2. Those changes in voltage are also present in EEG recordings, where they are typically considered an artifact that needs to be removed.

However, recently different authors have shown that it is also possible to use this signal to track eye movements [12, 15, 23, 40]. A working EEG-based eye tracking system promises a few advantages over camera-based

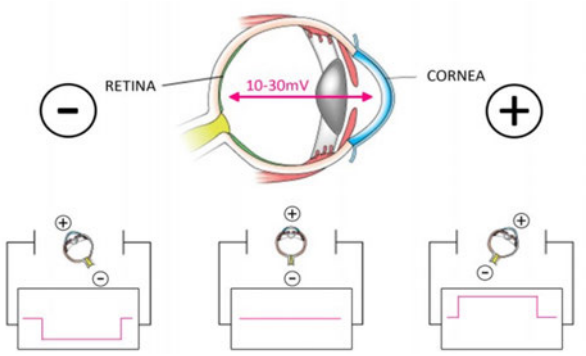

Figure 2.2: The eye as a dipole. The image shows how the voltage measured with electrodes placed around the eyes changes when the eye moves due to the cornea being positively and the retina negatively charged. Image taken from [12].

eye tracking. For example, it does not block a portion of the field of view and it can be used when the eyes are closed. Furthermore, it would simplify experiments that require both EEG and eye tracking (which are common in psychology and neuroscience) by removing the need for a separate eye tracking system that frequently needs to be recalibrated.

Other ways of getting the gaze position from EEG, that are not based on the electrical potential of the eye, have also been explored. For example, high blinking frequencies, on graphical user interfaces. When the user looks at graphical elements like a button that blinks at a certain frequency, a corresponding frequency can be extracted from the EEG signal, even when the blinking is not consciously perceived by the user [30]. Another approach tries to differentiate visual exploration fixations (*i.e.* looking at something without a specific goal, *e.g.* looking at different parts of a picture) and control fixations (*i.e.* intentionally looking at something, *e.g.* a button) just from the electrical activity of the brain [38].

2.2 RELATED WORK

In this section, we review work related to EEG-based eye tracking. We begin by introducing the EEGEyeNet dataset, a large-scale dataset collected using high-end equipment under laboratory conditions, which shares similarities with the dataset introduced in this thesis. In addition to our new dataset, the EEGEyeNet dataset will be used to evaluate the models developed in this work. We will also discuss two existing methods for EEG-based eye tracking: the SpatialFilterCNN, a deep learning model, and BMOTE, a white-box model that incorporates the physical processes of the eye through explicit modeling. As these methods have been shown to enable EEG-based eye tracking on other datasets, they will be used to assess whether our new dataset, collected with consumer-grade hardware, is suitable for this task, as well as to provide a basis for comparison with the new functional models developed in this thesis.

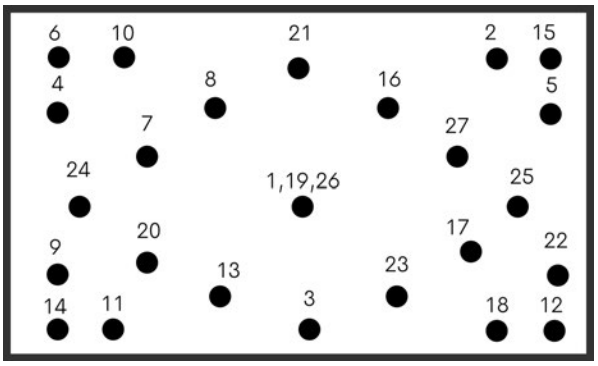

Figure 2.3: The layout of the grid used in the "Large Grid" paradigm of the EEGEyeNet study. Image taken from [23].

2.2.1 *Related Dataset: EEGEyeNet*

In the paper titled "EEGEyeNet: a Simultaneous Electroencephalography and Eye-tracking Dataset and Benchmark for Eye Movement Prediction" [23], the authors introduce a new dataset called EEGEyeNet together with an accompanying benchmark for eye movement prediction. In their work they collected data in three different experimental paradigms, from 356 participants comprising a total of 47 hours of data. This makes it the largest known dataset specifically collected for EEG-based eye tracking to date.

The recording setup consisted of a 128-channel EEG Geodesic Hydrocel system using wet electrodes at a sampling rate of 500 Hz and an infrared video-based eye tracking EyeLink 1000 Plus from SR Research at a sampling rate of 500 Hz and an accuracy of 0.25°-0.50° [35]. The eye tracker was carefully calibrated before each recording, repeating the calibration until the error between two measurements at any point was less than 0.5°, or the average error for all points was less than 1°. During recordings the head position was stabilized using a chin rest, placing the participants at a distance of 68 cm away from a 24 inch monitor, with a resolution of 800 px × 600 px.

Experimental Paradigms

The three different experimental paradigms were called "Pro- and Antisaccade", "Large Grid" and "Visual Symbol Search". In the "Pro- and Antisaccade" paradigm participants were first asked to direct their gaze to the center of the screen. After a randomized time-period, a dot appeared on the left or right hand-side of the central fixation square. The participants were then asked to either look at the dot (prosaccade) or to look at the opposite side of the screen (antisaccade). In both cases the dot was shown for 1 s before disappearing. The participants were then asked to return their gaze to the center of the screen.

In the "Large Grid" paradigm the subjects had to fixate on a dot which repeatedly changed its position to one of 25 points on a grid. The layout of the grid is shown in Figure 2.3. The order of the points at which the dot

Paradigm	Total Time
Pro- and Antisaccade	38 h
Large Grid	7 h 52 min
Visual Symbol Search	1 h 29 min

Table 2.1: Total time of data collected for each paradigm in the EEGEyeNet dataset using minimal preprocessing.

appeared on the screen was the same for every participant, but an effort was made to make it hard for the participant to predict the next position.

In the final paradigm "Visual Symbol Search" participants were shown 15 rows of symbols, each row containing 5 symbols and 2 target symbols. The participants were tasked with finding the target symbols in each row. If the target symbols were present in a row, the participant was asked to confirm this with a click on a button next to the row saying "YES", otherwise they were asked to click on a button saying "NO".

Data for each paradigm is available in two different levels of preprocessing: minimal and maximal. The minimal preprocessing consists of the removal and interpolation of bad electrodes as well as filtering the data with a 40 Hz low-pass filter and a 0.5 Hz high-pass filter. The maximal preprocessing additionally includes the removal of various different artifacts, such as muscle, heart, line noise, channel noise, but, importantly, also eye movement artifacts. The amount of data collected varies between the different paradigms. In Table 2.1 the total time per paradigm is shown for the minimally preprocessed data.

Benchmark

In addition to the dataset, the authors also introduce a benchmark for EEG-based eye tracking based on their dataset. This benchmark consists of three tasks of increasing difficulty. In each task the minimally preprocessed data is used and the recordings are cut into 1 s windows of shape 500×128 (time points $\times$ channels). The first task called "Left-Right" is based on the data from the "Pro- and Antisaccade" paradigm, specifically the data from the prosaccade trials. Windows in this task are created in such a way that they start with the appearance of the dot on the left or right side of the screen. The task is to predict one of the classes "left" or "right" based on the data in the window. Performance is evaluated using the standard accuracy metric (#True Predictions / #All Predictions).

The second task called "Angle/Amplitude" is based on the data from the "Large Grid" paradigm. Windows in this task are created in such a way that the saccade is in the middle of the window. The task is to predict the angle and amplitude of the saccade. This is a regression task and performance is evaluated using the Root Mean Squared Error (RMSE) for the angle (in radi-

ans along the shortest path of the unit circle) and amplitude (in millimeters) separately.

The third task is called "Absolute Position". It is again based on data from the "Large Grid" paradigm. In this task the windows only contain the fixation period after a saccade. The task is to predict the absolute position of the gaze on the screen in millimeters. Performance is measured as the Mean Euclidean Distance (MED) between the predicted and the true gaze position.[1] The benchmark does not include a task for the "Visual Symbol Search" paradigm.

Together with the dataset the authors also published a file for every task with the correct windows already cut out and sorted in a way to be split in a clearly defined and reproducible way into the train, validation and test set. The split is done across participants with 70% of participants in the train set, 15% in the validation set and 15% in the test set.

Baseline Results

Finally, the authors also present baseline results for each of the tasks for different models. The models are generally divided into two categories: "Machine Learning" and "Deep Learning". The "Machine Learning" models are based on hand-crafted features and include models like Random Forest, K-Nearest Neighbors and Linear Regression. The "Deep Learning" models are based on neural networks and include a standard Convolutional Neural Network (CNN), a pyramidal shaped CNN, the EEGNet model by Lawhern et al. [25], an InceptionTime model [14] and an Xception model [9]. Every model is evaluated five times and the mean performance and standard deviation are reported. In addition to results for each of those models, a naive baseline is established using the mean of the training set for regression tasks and the majority class for the classification task.

In their paper the authors find that the "Deep Learning" models outperform the machine learning models on all tasks. A "reduced" version of the results from the paper is shown in Table 2.2 focusing on the "Left-Right" and "Absolute Position" tasks, which are the most relevant tasks for this thesis as well as only showing the performance of the "Deep Learning" models which are most strongly related to the other methods introduced or developed in this thesis.

2.2.2 *Other Method: SpatialFilterCNN*

A successor to the EEGEyeNet paper is the paper titled "One step closer to EEG based eye tracking" [15] in which the authors propose a new method for EEG-based eye tracking and evaluate it on the "Absolute Position" task of the EEGEyeNet benchmark. The authors introduce a novel neural network

1 Even though the euclidean distance is stated as the metric that should be used to evaluate the performance on this task, the authors themselves do not report the euclidean distance in their baseline results, but the RMSE instead.

Model	Left-Right (Accuracy)	Absolute Position (RMSE)
CNN	98.3 ± 0.5	70.2 ± 1.1
PyramidalCNN	98.5 ± 0.2	73.6 ± 1.9
EEGNet	98.6 ± 0.1	81.7 ± 1.0
InceptionTime	97.9 ± 1.1	70.8 ± 0.8
Xception	98.8 ± 0.1	78.7 ± 1.6
Naive Baseline	52.3	123.3

Table 2.2: Excerpt of the results from [23] focusing on the "Left-Right" and "Absolute Position" tasks and the "Deep Learning" models.

architecture, where the design of the first layers is inspired by spatial filtering methods. A spatial filter is a filter that acts across different electrodes, combining signals at a single time point. This is in contrast to temporal filtering, which is also common in EEG data processing, and often includes filtering out certain frequency ranges using bandpass or notch filters.

In every case the goal is to enhance the signal to noise ratio. The authors explain that in their network the first layer is inspired by Canonical Correlation Analysis, in which a linear transformation is used to maximize the correlation between two sets of variables. In their case this is done by a convolutional layer that — in theory — transforms the EEG data into a space that maximizes the correlation between the transformed EEG data and the eye movement data. The first layer should thus act as a learnable spatial filter. The authors did not give their method a name, but for the purpose of this thesis it will be referred to as "SpatialFilterCNN", which reflects the main idea behind the method.

The detailed architecture of the SpatialFilterCNN model is shown in Figure 2.4. The model consists of a spatial filtering layer, two residual blocks and two fully connected layers at the output. In the case of the EEGEyeNet data, the input is a matrix of shape 500×128 (time points $\times$ channels). The spatial filtering layer is a one-dimensional convolutional layer with 16 filters and a kernel size of 1. It is followed by a batch normalization layer and a Rectified Linear Unit (ReLU) activation function. The output of this layer is then passed through two residual blocks. Each residual block consists of two one-dimensional convolutional layers, with either 32 filters for the first residual block or 64 filters for the second. The first convolutional layer in each residual block has a kernel size of 9, and uses padding to keep the first dimension of the output shape the same as the input shape.[2] This is followed by a Batch Normalization layer and a ReLU activation function.

The second convolutional layer has a kernel size of 1 and is followed by another Batch Normalization layer. The output of the second convolu-

2 Some details of the architecture are not mentioned in the paper, but were necessary to implement the model or reproduce the reported parameter count. This includes the use of padding in the first convolutional layer of the residual blocks.

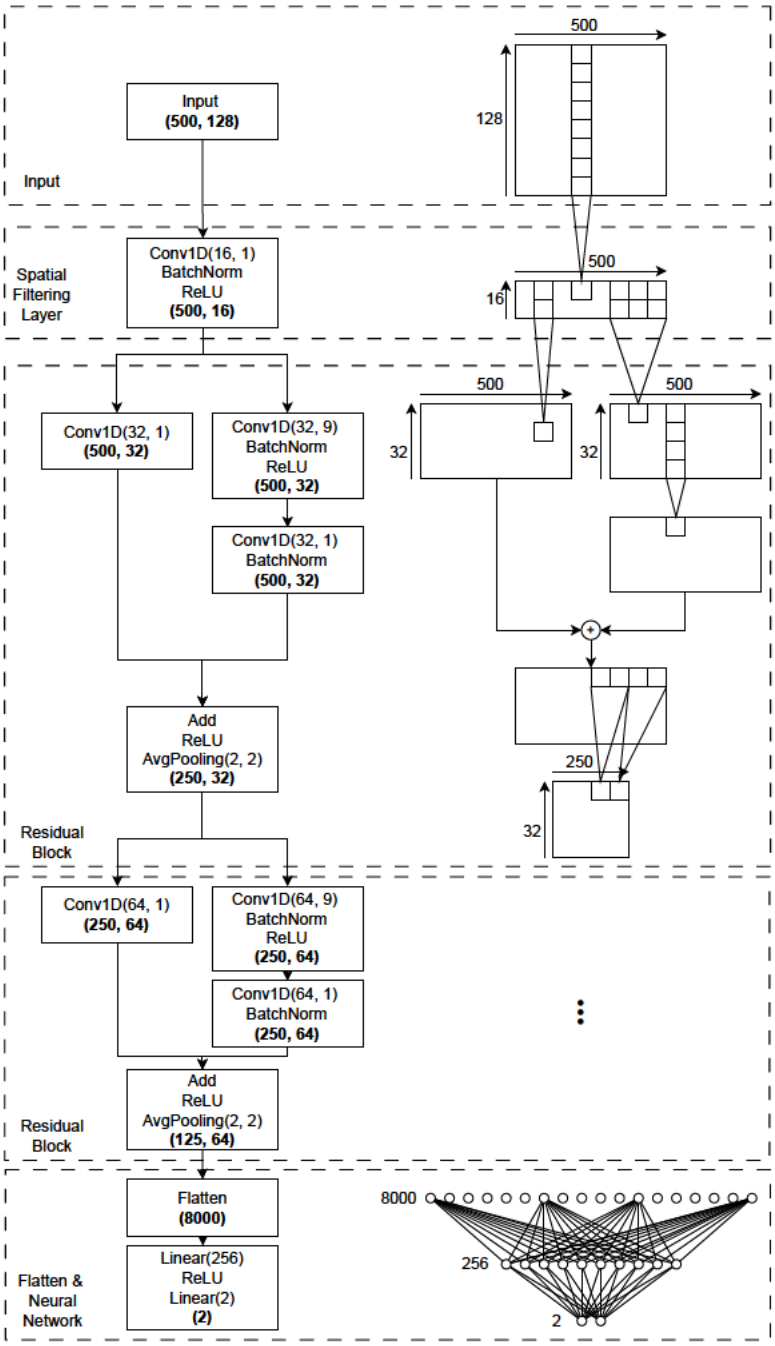

Figure 2.4: The architecture of the SpatialFilterCNN model, which on a high level consists of the input, a spatial filtering layer, two residual blocks and a neural network at the output. On the left side those components are broken down further into smaller blocks. The numbers in brackets at the end of each block denote the output shape of the block. Conv1D(n, m) stands for a one-dimensional convolutional layer with n filters and a kernel size of m, AvgPooling(n, m) for average pooling with pool size n and stride m and Linear(n) for a layer of n fully connected neurons. On the right side the same layers are illustrated graphically.

Model	Absolute Position MAE
CNN	86.61 ± 1.12
PyramidalCNN	90.25 ± 0.50
EEGNet	97.03 ± 1.07
InceptionTime	88.38 ± 1.31
Xception	94.65 ± 1.59
SpatialFilterCNN	49.20 ± 0.60

Table 2.3: The MAE of the "Deep Learning" models and the new SpatialFilterCNN on the "Absolute Position" task of the EEGEyeNet benchmark as reported in [15].

tional layer is then added to the input of the residual block, which is passed through another one-dimensional convolutional layer in order to match the output shape of the second convolutional layer.[3] The added output is then passed through a ReLU activation function and an average pooling layer with a pool size of 2 and a stride of 2.[4] After the second residual block the output is flattened and passed through a fully connected layer with 256 neurons and a ReLU activation function. The output of this layer is then passed through another fully connected layer with 2 neurons, which is the output of the model and represents the prediction of the gaze position in pixels.

The authors also experimented with slight variations of this architecture. They tested the model with and without the spatial filtering layer, and with both convolutional layers of the residual blocks having a kernel size of 9. Using the variant with a spatial filtering layer (`spatial_filtering`: true), and unequally sized convolutional layers (`equally_sized`: false), a Mean Absolute Error (MAE) of 49.2 on the "Absolute Position" task of the EEGEyeNet benchmark is reported, which is state of the art at the time of publication.

The authors do not give a reason for why they did not use the MED, which in the EEGEyeNet paper was defined to be the metric for this task, or the RMSE which was used in the baseline results of the EEGEyeNet paper, but instead use the MAE. The MAE for the other "Deep Learning" models are also reported in the paper and are shown in Table 2.3.

In an interesting note at the end of the paper, the authors acknowledge that the EEGEyeNet dataset only allows the evaluation of EEG-based eye tracking under laboratory conditions with wet electrodes, their goal however, is to use dry electrodes under more general conditions. This thesis hopes to address this issue by introducing the missing dataset and evaluating the SpatialFilterCNN model on it.

In total, the SpatialFilterCNN has the following hyperparameters:

3 The use of a convolutional layer to match the shapes at the add connection was not mentioned in the paper, but was necessary to implement the model

4 The use of a stride of 2 in the average pooling layer was not mentioned in the paper, but was necessary to reproduce the reported parameter count.

- **Number of Spatial Filters**: The number of filters N_S in the spatial filtering layer.
- **Number of Filters in 1st Convolutional Layer**: The number of filters N_1 in the first convolutional layer.
- **Number of Filters in 2nd Convolutional Layer**: The number of filters N_2 in the second convolutional layer.
- **Equally Sized Convolutional Layers**: A boolean value `equally_sized` indicating whether the two convolutional layers in the residual blocks have the same kernel size.
- **Inclusion of Spatial Filtering Layer**: A boolean value `spatial_filtering` indicating whether the spatial filtering layer is included.

2.2.3 *Other Method: Battery Model of the Eye*

In the paper titled "EOG-Based Gaze Angle Estimation Using a Battery Model of the Eye" [2] the authors develop a method to estimate the gaze angle using Electrooculography (EOG). In this document we will refere to this method as "BMOTE" short for "Battery Model of the Eye". Despite the fact that this method is based on EOG and not EEG, it is still relevant for this thesis and can easily be adapted. This is because the main principle to detect eye movements behind both methods is the same: the eye as a dipole. In contrast to other methods which are based on "black-box" models (like the ones presented in the previous sections), this method is an explicit, physically-driven, "white-box" model. That is, the eye is modeled as a battery with a positive pole at the cornea and a negative pole at the retina. In detail the authors extend the model proposed by Shinomiya et al. in [37] to account for both eyes.

In the extended model the voltage V_i at the i-th electrode is given by

$$V_i = \frac{I}{4\pi\sigma}\left(\frac{1}{d_c^{(l)}} - \frac{1}{d_r^{(l)}}\right) + \frac{I}{4\pi\sigma}\left(\frac{1}{d_c^{(r)}} - \frac{1}{d_r^{(r)}}\right), \tag{2.1}$$

where $d_c^{(l)}$ and $d_r^{(l)}$ are the distances from the i-th electrode to the the cornea and retina of the left eye, $d_c^{(r)}$ and $d_r^{(r)}$ are the distances from i-th electrode to the cornea and retina of the right eye, I is the electrical current flowing inside the eyeball from the retina to the cornea, radially out of the cornea, and back inside the eyeball through the retina, and σ denotes the conductivity of the eyeball. In [2] the constant $I/4\pi\sigma$ is empirically determined, and set to $25\,\mu\mathrm{V\,m}$.

Writing out the distances in detail we get:

$$d_c^{(l)} = \|\mathbf{p}_i - \mathbf{p}_c^{(l)}(\boldsymbol{\theta})\|_2 \qquad d_c^{(r)} = \|\mathbf{p}_i - \mathbf{p}_c^{(r)}(\boldsymbol{\theta})\|_2 \tag{2.2}$$

$$d_r^{(l)} = \|\mathbf{p}_i - \mathbf{p}_r^{(l)}(\boldsymbol{\theta})\|_2 \qquad d_r^{(r)} = \|\mathbf{p}_i - \mathbf{p}_r^{(r)}(\boldsymbol{\theta})\|_2, \tag{2.3}$$

where $\mathbf{p}_i$ is the position of the i-th electrode and $\mathbf{p}_c^{(l)}$, $\mathbf{p}_c^{(r)}$, $\mathbf{p}_r^{(l)}$ and $\mathbf{p}_r^{(r)}$ are the positions of the left and right cornea and retina respectively, which are functions of the gaze angle $\boldsymbol{\theta}$. The detailed relationship between retina/cornea positions and gaze angle are described in [2]. We will only note that they depend on the radius r of the eyeball and the interpupillary distance d_{PD}, which are assumed to be known.

Therefore, equation (2.1) relates the following three quantities: the voltage at the electrode, the position of the electrode, and the gaze angle (via the position of the cornea/retina). This means that in order to estimate the gaze angle from the voltage at the electrodes, the electrode positions need to be known. But at the same time, given data of the voltages at the electrodes and the gaze angle that induced them, the electrode positions can be estimated! Thus, the authors propose a two-step algorithm to estimate the gaze angle. First the electrode positions are estimated from the data. Then in a second step the gaze angle can be calculated based on the estimated electrode positions and voltage readings.

The electrode positions can then be estimated by solving the following least squares problem:

$$\hat{\mathbf{p}}_i = \arg\min_{\mathbf{p}_i} \left\| \mathbf{V}_i^{(\text{mod})}(\boldsymbol{\theta}; \mathbf{p}_i) - \mathbf{V}_i^{(\text{rec})} \right\|_2^2, \tag{2.4}$$

where $\mathbf{V}_i^{(\text{mod})}$ is the vector of voltages at the electrode i for the gaze angle $\boldsymbol{\theta}$, calculated using the model, and $\mathbf{V}_i^{(\text{rec})}$ is the vector of voltages at the electrode i for the gaze angle $\boldsymbol{\theta}$, as recorded in the data. This is however not what the authors propose in [2]. We still mention the first method as it resulted in more understandable results in our experiments.

The authors propose to solve the following least squares problem:

$$\hat{\mathbf{p}}_i = \arg\min_{\mathbf{p}_i} \left\| \Delta\mathbf{V}_i^{(\text{mod})}\left(\boldsymbol{\theta}^{(s)}, \boldsymbol{\theta}^{(f)}; \mathbf{p}_i\right) - \Delta\mathbf{V}_i^{(\text{rec})} \right\|_2^2, \tag{2.5}$$

where $\Delta\mathbf{V}_i^{(\text{mod})}$ is a vector made up of the terms $V_i(\boldsymbol{\theta}_k^{(f)}) - V_i(\boldsymbol{\theta}_k^{(s)})$, *i.e.* the voltage difference at the electrode i for the k-th saccade at the start and end of the saccade, calculated using the model and $\Delta\mathbf{V}_i^{(\text{rec})}$ is the vector of voltage differences at the electrode i for the k-th saccade at the start and end of the saccade, as recorded in the data. In the paper, the authors did not provide a reason for why they chose this method over the first one, but reaching out to them, they mentioned that the second method, based on $\Delta\mathbf{V}_i$ instead of $\mathbf{V}_i$, was used to mitigate the impact of potential errors in the voltage measurements. These errors could arise due to baseline drift, which can skew the absolute voltage values over time. By using the $\Delta\mathbf{V}_i$ values, the authors hoped to reduce the impact of such errors.

After the electrode positions $\hat{\mathbf{P}} := (\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_{N_e})$ of all N_e electrodes are estimated, the gaze angle can be calculated by solving the following least squares problem:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}} \left\| \mathbf{V}^{(\text{mod})}\left(\boldsymbol{\theta}; \hat{\mathbf{P}}\right) - \mathbf{V}_t^{(\text{rec})} \right\|_2^2, \tag{2.6}$$

where $\mathbf{V}^{(\text{mod})} := (V_1(\boldsymbol{\theta}_t), \ldots, V_{N_e}(\boldsymbol{\theta}_t))^\top$ is the vector of voltages at all electrodes for the gaze angle $\boldsymbol{\theta}_t$, calculated using the model, and $\mathbf{V}_t^{(\text{rec})}$ is the vector of voltages at all electrodes for the gaze angle $\boldsymbol{\theta}_t$, as recorded in the data. This is very similar to (2.4), but instead of fixing the gaze angle and estimating the electrode positions, the electrode positions are fixed and the gaze angle is estimated.

2.3 BASICS OF FUNCTIONAL DATA ANALYSIS

This section provides a basic overview of Functional Data Analysis (FDA), which underlies the new methods developed in this thesis. Put very generally — though admittedly not very helpful — the field of FDA is concerned with "the analysis of functional data". For this statement to be useful one first has to know what functional data even is.

Functional data consist of observations that can be viewed as functions $x_i(t)$, where t belongs to some continuous domain (e.g., time, space, wavelength), and i indexes the individuals or units. In practice, the data is usually discrete, meaning that we only have a finite number of observations at certain points, however the underlying assumption is that these observations are samples from a function. This function is assumed to be smooth, *i.e.* differentiable to some degree. This means that adjacent data values are linked together to some extent and unlikely to be too different from each other. If this smoothness property did not apply, there would be nothing much to be gained by treating the data as functional rather than just multivariate.

In order to better understand what functional data really is and differentiate it from multivariate data, an example might be useful. For this, the "weather station data" frequently meantioned in [32] should be well suited. The data consists of monthly averages of various weather related variables like temperature, precipitation, *etc.* collected from many weather stations across Canada. The temperature curves of a few different weather stations can be seen in Figure 2.5.

Treating this as multivariate data, one would have a separate variable for each measurement at each month, *i.e.* $X_{\text{Temp,Jan}}$, $X_{\text{Temp,Feb}}$, $\ldots$, $X_{\text{Precipitation,Jan}}$, *etc.* Looking at the data in this way, one would neglect the intrinsic order of the variables and the spacing between measurements, as well as the smooth nature of the data that connects the values at the different time points together. The very intuitive plot shown in Figure 2.5 is actually very unusual under the multivariate framework, as we are plotting the realisation of a variable against its index. This is not normally done for multivariate data, as the order of the variables is arbitrary. All of this suggests that we should not sim-

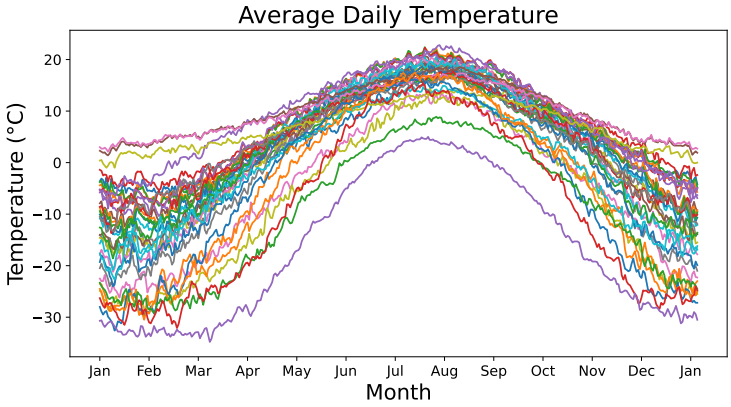

Figure 2.5: Daily temerature averages of 35 canadian weather stations, downloaded from https://www.psych.mcgill.ca/misc/fda/downloads/. This dataset is prominently used by Ramsay and Silverman in [32] to introduce concepts of FDA.

ply treat the data as multivariate, but rather as functional. It is much more natural to think of the temperatures at the different time points to come from a continuous function $x_{\text{Temp}}(t)$, and the precipitation from another function $x_{\text{Precipitation}}(t)$, *etc.*

Functional data is actually quite common, think of images, audio, etc, these are all functional data. But typically such data is not explicitly modelled as functional data (*i.e.* the discrete samples are used) and the tools for analysing this data ignore the functional structure. This is where FDA comes in. It is the toolbox to explicitly treat the observations as functions and to apply methods that "exploit" the functional structure.

Smoothing Functions

Since the data typically comes in discrete form, the first step in FDA is often to turn the samples into a function. As stated before, in the functional framework, it is assumed that the observations $y_1, \ldots, y_N$ are samples from a smooth function $x(t)$. Incorporating the noise that is typically present in the raw data, we write

$$y_i = x(t_i) + \epsilon_i, \tag{2.7}$$

where ϵ_i is the noise. The goal is now to estimate the underlying function $x(t)$ from the noisy data y_i, which is quite a challenge as the space of functions is infinite-dimensional. For this purpose one typically uses basis expansions. That means that the function $x(t)$ is represented by

$$x(t) = \sum_{m=1}^{M} c_m \phi_m(t), \tag{2.8}$$

where $\phi_m(t)$ are the basis functions and c_m are the coefficients. With this approach, we can effectively bridge the gap between the infinite-dimensional world of functions and the finite-dimensional world of vectors. This becomes

especially apparent when we consider that the coefficients c_m can be seen as a vector $\mathbf{c}$ and the basis functions $\phi_m(t)$ as a *functional vector* $\mathbf{\Phi}_M(t)$, so that we can write

$$x(t) = \mathbf{c}^\top \mathbf{\Phi}_M(t). \tag{2.9}$$

A typical choice are the normed Fourier basis functions on the interval $[0, 1]$, which are especially well suited for periodic data. Those are the functions

$$1, \sqrt{2}\sin(2\pi n x), \sqrt{2}\cos(2\pi n x) \quad \text{for} \quad n \in \mathbb{N}.$$

Another common choice are the Legendre polynomials

$$1, t, \frac{1}{2}3t^2 - 1, \frac{1}{2}5t^3 - 3t, \ldots,$$

which are frequently used to represent functions of the form $1/t$, commonly encountered in physics. One example is in electrostatics, which is also where the Legendre polynomials were first introduced by Adrien-Marie Legendre.

Both the Fourier basis and the Legendre basis are *complete orthogonal* basis for the space of square-integrable functions $L^2([a, b]; \mathbb{R})$ equipped with the inner product $\langle f, g \rangle = \int_a^b f(t)g(t)dt$. The Fourier basis is complete on the interval $[0, 2\pi]$ and the Legendre basis on the interval $[-1, 1]$. This means that for any $f \in L^2([a, b]; \mathbb{R})$ it holds that

$$\lim_{M \to \infty} \left\| f(t) - \mathbf{c}^\top \mathbf{\Phi}_M(t) \right\|_2 = 0, \tag{2.10}$$

where $\|\cdot\|_2 = \sqrt{\langle f, f \rangle}$ is the L^2-norm and c_m are the coefficients of the basis expansion, which (because of the orthogonality) are simply $\langle f, \phi_m \rangle = \int_a^b f(t)\phi_k(t)dt$.

There are various methods to estimate the coefficients c_m from the data. The simplest one is probably the least squares method, which finds the coefficients that minimize the squared error between the data and the basis expansion

$$\sum_{i=1}^{N} \left(y_i - \mathbf{c}^\top \mathbf{\Phi}_M(t_i) \right)^2 = \left(\mathbf{y} - \mathbf{\Phi}_M \mathbf{c} \right)^\top \left(\mathbf{y} - \mathbf{\Phi}_M \mathbf{c} \right), \tag{2.11}$$

where $\mathbf{y} = (y_1, \ldots, y_N)^\top$ is the vector of the data and $\mathbf{\Phi}_M$ is the matrix of column vectors $\mathbf{\Phi}_M(t_i)$. As stated in [32, p. 60], the solution to this problem is given by

$$\hat{\mathbf{c}} = \left(\mathbf{\Phi}_M^\top \mathbf{\Phi}_M \right)^{-1} \mathbf{\Phi}_M^\top \mathbf{y}. \tag{2.12}$$

By changing the objective function, one can get different estimators. For example, locally adaptive methods are possible by weighting the observa-

tions close to the point of interest more heavily. The corresponding objective function is then

$$\sum_{i=1}^{N} w_i(t) \left(y_i - \mathbf{c}^\top \mathbf{\Phi}_M(t_i)\right)^2 = (\mathbf{y} - \mathbf{\Phi}_M \mathbf{c})^\top \mathbf{W}(t) (\mathbf{y} - \mathbf{\Phi}_M \mathbf{c}), \tag{2.13}$$

where $\mathbf{W}(t)$ is a diagonal matrix with the weights $w_i(t)$ on the diagonal. In this case the solution is given by

$$\hat{\mathbf{c}}(t) = \left(\mathbf{\Phi}_M^\top \mathbf{W}(t) \mathbf{\Phi}_M\right)^{-1} \mathbf{\Phi}_M^\top \mathbf{W}(t) \mathbf{y}, \tag{2.14}$$

where we refer to [32, p. 77] for more details. Note, that using this method a different coefficient vector $\hat{\mathbf{c}}(t)$ needs to be estimated for each point t.

Another very common alternative adds a "roughness penalty" to the objective function. Measuring the roughness of a function by the squared L^2-norm of its second derivative, the objective function becomes

$$\sum_{i=1}^{N} \left(y_i - \mathbf{c}^\top \mathbf{\Phi}_M(t_i)\right)^2 + \lambda \left\| \frac{d^2}{dt^2} \mathbf{c}^\top \mathbf{\Phi}_M(t) \right\|_2^2 = (\mathbf{y} - \mathbf{\Phi}_M \mathbf{c})^\top (\mathbf{y} - \mathbf{\Phi}_M \mathbf{c}) + \lambda \mathbf{c}^\top \mathbf{R} \mathbf{c}, \tag{2.15}$$

where $\mathbf{R}$ is the matrix with entries

$$R_{ij} = \int_a^b \frac{d^2}{dt^2} \phi_i(t) \frac{d^2}{dt^2} \phi_j(t) dt \tag{2.16}$$

and λ controls the trade-off between the fit to the data and the smoothness of the function. The solution to this problem can be found in [32, p. 87] and is given by

$$\hat{\mathbf{c}} = \left(\mathbf{\Phi}_M^\top \mathbf{\Phi}_M + \lambda \mathbf{R}\right)^{-1} \mathbf{\Phi}_M^\top \mathbf{y}. \tag{2.17}$$

Having estimated the underlying function $x(t)$ is already quite useful. For example, one can now evaluate the function at any point in the domain, thus, be able to handle missing data or irregularly sampled data. Furthermore, it is now much more feasible to calculate derivatives of the function, which due to the noise can be quite challenging to do directly from the raw discrete data.

Registering Functions

For many applications one more step is necessary before functional methods can be applied. In order to compare functions, it is often necessary to align them, such that evaluating the functions at the same point t corresponds to the same feature. This is called "registering" the functions. This is especially relevant for data where the event of interest does not happen at the same point in time for all observations. For example, in the EEGEyeNet data saccades do not always happen at the same point in time for all participants. The simplest way to register functions is by shifting them until a certain

criterion is met. *e.g.* shifting them until a specific feature (*e.g.* a peak, a zero-crossing, *etc.*) is at the same point t for all functions, or shifting them such that the L^2-distance to the mean function is minimized.

This is however only possible if meaningful features can be identified in the data or if the onset of the event of interest is externally triggered. In the case of EEG data, where the event of interest is often a self-paced action and the signal is noisy and complex, curve registration is infeasible. To address this issue, a shift invariant method has been proposed in [20] which is based on the idea of Convolutional Neural Networks (CNNs). The new class of neural networks is called "Functional Neural Networks" and will be introduced in the next section.

2.4 FUNCTIONAL NEURAL NETWORKS

In [20] the authors extend both the classical fully connected/dense layers and the convolutional layers to functional data. We will first introduce the functional convolutional layers as those only differ slightly from the classical convolutional layers. Then we will introduce the functional dense layers, which are rather different from the classical dense layers.

Functional Convolutional Layers

We will focus on one dimensional convolutions as those are sufficient for the EEG data handled in this thesis. However, everything discussed can easily be extended to higher dimensions. Furthermore the batch dimension will be ignored in the following discussion, as it only serves technical and performance reasons and does not change the underlying principles.

The input to a one dimensional convolutional layer is a matrix $\mathbf{X} \in \mathbb{R}^{S_{in} \times C_{in}}$, where S_{in} is the number of "steps" at the input and C_{in} is the number of "input channels". The steps correspond to the domain of the function (*e.g.* time or space) and the channels to the number of functions (*e.g.* different variables like temperature, precipitation, *etc.* or different electrodes of the EEG).

A convolutional layer is a function that maps such an input matrix to an output matrix $\mathbf{Y} \in \mathbb{R}^{S_{out} \times C_{out}}$ by convolving the input matrix with a set of filters. Here S_{out} is the number of steps at the output and C_{out} is the number of "output channels" which corresponds to the number of filters. The mapping is given by

$$Y_{s,k} = \sigma\left(b_{s,k} + \sum_{j=1}^{C_{in}} \sum_{t=1}^{K} w_{t,j,k} \cdot X_{s-t,j}\right), \tag{2.18}$$

where s is the step and k the channel index at the output, K is the kernel size and $w_{t,j,k}$ is the weight at the input channel j for of the filter k at position t of the kernel. The activation function σ, which is applied at the end, is there to introduce non-linearity.

One important thing to note, is that a convolutional layer already expects a functional input (*i.e.* an ordered list of scalars where entries that are close together are related to each other). Thus by substituting the input matrix $\mathbf{X}$ with a functional input $x(t)$, the weights $w_{t,j,k}$ with functional weights $w_{j,k}(t)$ defined on the interval $[0,1]$, the bias $b_{s,k}$ with a functional bias $b_k(t)$ and the sum with an integral, we already get the scaffold for a functional convolutional layer

$$y_k(s) = \sigma\left(b_k(s) + \sum_{j=1}^{C_{in}} \int_0^1 w_{j,k}(t) x_j(s-t) dt\right). \tag{2.19}$$

In order to learn the weight functions $w_{j,k}(t)$ and bias functions $b_k(t)$ from the data, several methods are possible. One could for example train the model through backpropagation based on Fréchet derivatives which is the approach taken in [34] and [33]. Alternatively, one can use a basis expansion, as in (2.7), to represent the weight and bias functions

$$w_{j,k}(t) = \sum_{m=1}^{M} c_m^{(j,k)} \phi_m(t) \quad \text{and} \quad b_k(t) = \sum_{m=1}^{M} c_m^{(k)} \phi_m(t) \tag{2.20}$$

and learn the coefficients of the expansion via standard backpropagation. This is the approach taken in [20] which simplifies the computations and simultaneously reduces the dimension of the parameter space.

The way (2.19) is implemented in practice is by discretizing the integral and replacing it by a sum. Furthermore, the level of discretization can be controlled by a "resolution" parameter R, which determines the number of samples the discretization uses. With this the mapping becomes

$$y_k(s) = \sigma\left(b_k(s) + \sum_{j=1}^{C_{in}} \sum_{t=1}^{R+1} w_{j,k}((t-1)/R) x_j(s-t)\right), \tag{2.21}$$

which is very similar to the classical convolutional layer (2.18). Thus, the real practical difference between the two is the way the weights are represented and learned. In the functional convolutional layer the weights and biases are always "smooth" along the "steps"-dimension and can only be set indirectly by changing the coefficients of the basis expansion. This reduces the number of parameters, which might allow for a more interpretable model and result in a kind of regularization that might help against overfitting.

A functional convolutional layer has the following hyperparameters:

- **Resolution**: The resolution of the basis functions R, which is comparable to the kernel size in classical convolutional layers.
- **Type of Basis Functions**: The type of basis functions used to represent the weight and bias functions (*e.g.* Fourier or Legendre).
- **Number of Basis Functions**: The number of basis functions M used to represent the weight and bias functions.

- **Number of Filters**: The number of filters C_{out} used in the layer.
- **Activation Function**: The activation function σ used to introduce non-linearity.

Functional Dense Layers

A classical fully connected or dense layer maps an input vector $\mathbf{x} \in \mathbb{R}^{N_{in}}$ to an output vector $\mathbf{y} \in \mathbb{R}^{N_{out}}$ by the function

$$y_k = \sigma\left(b_k + \sum_{j=1}^{N_{in}} w_{j,k} x_j\right), \tag{2.22}$$

where $W_{j,k}$ are the weights, b_k are the biases and σ is the activation function of the layer. There are many ways to extend this to functional data. The authors in [20] settled on the following approach: the neurons x_j at the input are replaced by functions $x_j(t)$, the weights $w_{j,k}$ by functional weights $w_{j,k}(t)$ defined on the interval $[0, 1]$ and the biases b_k by bias functions $b_k(t)$. The mapping then becomes

$$y_k(t) = \sigma\left(b_k(t) + \sum_{j=1}^{C_{in}} w_{j,k}(t) x_j(t)\right), \tag{2.23}$$

where we renamed N_{in} to C_{in} since every neuron now corresponds to a complete function occupying its own channel. Just like before we represent the weights and biases by a linear combination of basis functions in order to simplify the gradient computation and reduce the number of parameters.

Note that in contrast to the functional convolutional layer, where the input and output of the classical counterpart were already functional, the input and output to a classical dense layer is not functional. This changes however for the functional dense layer, where the input and output are now functional. This makes the functional dense layer, in its application, more similar to the classical convolutional layer than to the classical dense layer, taking in inputs from $\mathbb{R}^{S_{in} \times C_{in}}$ and outputting to $\mathbb{R}^{S_{out} \times C_{out}}$.

It is also possible to define a functional dense layer with functional inputs and *scalar* outputs. This is done by taking the scalar product between the weights and input functions and reverting the bias function to a scalar bias. This can be useful for example to produce a scalar output at the last layer of a neural network. The mapping is given by

$$y_k = \sigma\left(b_k + \sum_{j=1}^{C_{in}} \int_0^1 w_{j,k}(t) x_j(t) dt\right). \tag{2.24}$$

This is implemented in practice by approximating the integral using the mean, leading to

$$y_k = \sigma\left(b_k + \sum_{j=1}^{C_{in}} \frac{1}{S_{in}} \sum_{t=1}^{S_{in}+1} w_{j,k}((t-1)/S_{in})x_j(t)\right). \tag{2.25}$$

This can be seen as a global average pooling operation, where the input is weighted by a smooth function that is learned, before the average is taken.

A functional dense layer has the following hyperparameters:

- **Type of Basis Functions**: The type of basis functions used to represent the weight and bias functions (*e.g.* Fourier or Legendre).
- **Number of Basis Functions**: The number of basis functions M used to represent the weight and bias functions.
- **Number of Neurons**: The number of neurons (functions), C_{out}, at the output of the layer.
- **Activation Function**: The activation function σ used to introduce non-linearity.
- In this case, the resolution R is not a hyperparameter, as it must always equal the number of steps S_{in} at the input.

3

DATASET

Because of the lack of existing datasets that combine EEG and eye tracking data, where the EEG is collected using a consumer-grade device, a new dataset was created. The dataset, which will be made freely available, is called the "Consumer EEG-ET dataset". Using this dataset, the performance of existing EEG-ET methods as well as new ones can be evaluated on consumer-grade EEG. Furthermore, it is intended to serve as a new challenging benchmark for FDA techniques as well as future research in the field of EEG-ET. In this chapter, the dataset is described in detail. The structure of the sessions, the stimuli presentation, the data acquisition and hardware used are explained. Finally, a benchmark is proposed to evaluate the performance of existing and new methods on this dataset.

3.1 DESCRIPTION

The dataset consists of simultaneously recorded EEG and eye tracking data from 113 participants, collected during 116 sessions, in which 4 experimental paradigms were presented to the participants. The experimental paradigms are designed in a way that it becomes increasingly challenging to reconstruct the gaze position from the EEG data as the eye movement becomes less restricted. In every paradigm, the participants were instructed to follow a dot moving on the screen as accurately as possible with their eyes. The presentation was also designed in a way that helped the participants follow the dot very closely.

The participants were seated in front of a 24 inch desktop monitor with a resolution of 2560 px $\times$ 1440 px and a 60 Hz refresh rate, placed 60 cm away

Figure 3.1: Recording setup of the Consumer EEG-ET dataset. The person in the image is the author wearing the Muse S 2 Headband.

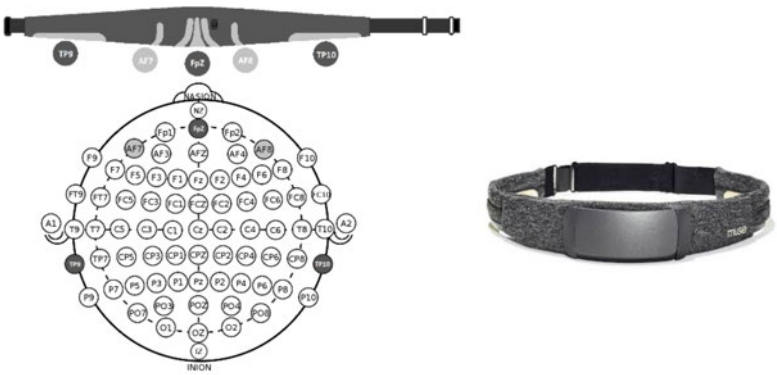

Figure 3.2: Electrode placement on the Muse S 2 Headband. The left image shows the headband laid out flat with the electrodes facing up together with a schematic of the electrode placement in the 10-20 system. The right image shows the headband from the front. Image adapted from [31].

from the participants and raised high enough such that the center of the screen matched the eye height of the subject. A webcam was mounted on a stand in front of the screen, adjusted to be as high as the lower edge of the screen, looking up at the subject. The recording setup is illustrated in Figure 3.1.

The webcam used was the Logitech StreamCam, which records in Full HD (1920 px × 1080 px), at 60 frames per second. The EEG data was recorded using the Muse S 2 Headband, a consumer-grade EEG-Headset with five dry electrodes (four EEG-channels and one reference electrode) used by more than 500,000 users, according to the manufacturer.[1] The headband records at a sampling rate of 256 Hz. An image of the headband and the electrode placement in the standard 10-20 system is shown in Figure 3.2. Two laptops were used: one for controlling the stimuli presentation and recording the data and one for showing the stimuli to the participants. Both laptops communicated via a socket connection.

In total, the dataset contains 11 hours and 45 minutes of EEG and eye tracking data, which amounts to 1.54 GB in size or 10,495,839 rows of data. The total time recorded per paradigm and the duration in one session can be seen in Table 3.1.

In addition to the EEG and eye tracking data, demographic data was collected from the participants. The demographic data includes the age, gender, handedness, vision correction, neurological disorders, and color blindness of the participants. The demographic data is summarized in Table 3.2, the age distribution is shown in Figure 3.3.

3.2 SESSION STRUCTURE

Most sessions took place during 90 minute practice groups of different artificial intelligence related courses at the Hochschule Darmstadt. The partici-

1 Information obtained from https://choosemuse.com/products/muse-s-gen-2, accessed on October 8, 2024.

Experimental Paradigm	Duration	Total Duration
level-1-smooth	56 s	1 h 49 min
level-1-saccades	1 min	1 h 56 min
level-2-smooth	2 min 8 s	4 h 8 min
level-2-saccades	2 min	3 h 52 min
Total	6 min 4 s	11 h 45 min

Table 3.1: Duration of the experimental paradigms in the Consumer EEG-ET dataset.

Demographic	No. of Participant	Percentage
Gender (male/female/diverse)	91/22/0	81%/19%/0%
Handedness (left/right/ambidextrous)	12/100/1	88%/11%/1%
Vision Correction (yes/no)	54/59	48%/52%
Neurological Disorder (yes/no)	6/107	5%/95%
Color Blind (yes/no)	2/111	2%/98%

Table 3.2: Demographic data of the participants in the Consumer EEG-ET dataset.

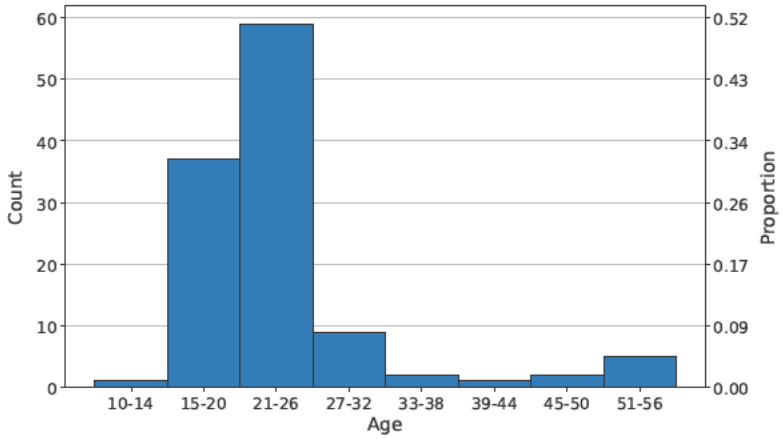

Figure 3.3: Histogram of the age distribution of the participants in the Consumer EEG-ET dataset. The left y-axis shows the number of participants in each age group, the right y-axis shows the proportion of participants in each age group.

pants were seated in the back of the room in which the practice group took place next to a big window. At the start of the group exercise, before any session started, all participants were briefed about the study and the data collection. The participants were informed about the purpose of the study, the data that would be collected, the duration of the study, that participation was voluntary, and that any collected data would be anonymized. After the briefing, the first subject was seated in front of the screen and the first session started.

Each session (nominally) lasted ten minutes in total and consisted of the following parts: introduction, practice, and recording. The introduction included the calibration of the camera-based eye tracker, fitting the EEG headset to the participants head, and checking signal quality. Furthermore, the distance to the screen and the height of the screen were adjusted to match the eye height of the subject. The participants also were instructed not to move or speak during the recording. Before each recording, a short practice session was conducted to prepare the participants for the respective experimental paradigm.

During the actual data collection, EEG and eye tracking data were recorded in four different situations: level-1-smooth, level-1-saccades, level-2-smooth, and level-2-saccades. The duration of each paradigm is shown in Table 3.1. A level-1 experiment only required the subject to move their eyes left, right, up or down for one minute. Data from those experiments is meant to serve as a first stepping stone for potential EEG-ET techniques. In the level-2 experiments the participants were presented with a dot that could move in any direction. Level-2 experiments took 2 minutes.

In a "smooth" experiment the dot moved across the screen in a continuous manner. The subject followed the dot using eye movement called "smooth pursuit", in which the eyes move smoothly to follow a moving object. A "saccades" experiment required the subject to follow the dot using a series of quick eye movements called "saccades". Here the dot jumped between predefined points on the screen. For all paradigms the stimuli presentation was designed in such a way, that the user could always anticipate the movement or a jump of the dot. This way the position of the dot is always close to the ground truth gaze position. If the participant experienced eye strain or fatigue, a break was taken before the next experiment.

Every participant was asked to sign two documents. The first document was a consent form, where the participants agreed to take part in the study, and the data collection. The second document was the demographic questionnaire, where the participants provided information about their age, gender, handedness, vision correction, neurological disorders, and color blindness. Both documents can be found in the appendix in English and German. (A.1 and A.2.) Depending on the language in which the practice group was held, either a German or an English version of the documents was used.

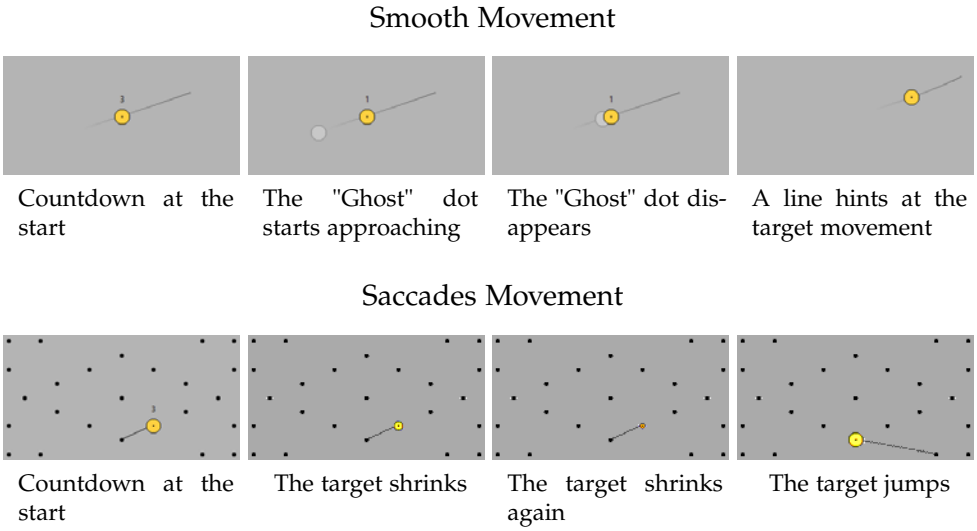

Figure 3.4: Stimuli presentation in the Consumer EEG-ET dataset. The presentation of the stimulus in a smooth experiment is shown on the top, the presentation of the stimulus in a saccades experiment is shown on the bottom.

3.3 STIMULI PRESENTATION

In addition to the EEG and eye tracking data, the position of the dot on the screen and the presentation state were recorded. All four data streams were synchronized using the Lab Streaming Layer Protocol. The lsl streams were recorded using the LabRecorder software, which properly resolves the synchronization information of multiple streams. The gaze position was calculated from the webcam video using the GazePointer software.

The dot was presented to the participants using a custom stimuli presentation software written in Python using the Qt framework. The presentation differed between the "smooth" and "saccades" paradigms. In a "smooth" experiment, in addition to the yellow dot that moved across the screen, which the participant was instructed to follow, a line was shown, that hinted the path of the movement. Furthermore, before the yellow dot started moving a "ghost" dot was shown, that hinted the start and speed of the movement. See Figure 3.4 for an example of the stimuli presentation in a "smooth" experiment.

In a "saccades" experiment a line connected the current position of the dot with the target position, *i.e.* the position to which the dot jumped to. To indicate the moment when the jump occurred the dot got scaled down 3 times, completely disappearing the last time. The moment the dot disappeared was the moment the jump happened. This is shown in the lower half of Figure 3.4. Those design choices were made to help the participants anticipate the movement of the dot, such that the position of the dot was always as close as possible to the ground truth gaze position.

The path along which the dot moved in the "smooth" experiments was created using a parameterized curve. In detail, for a given $t \in [0, 1]$ the x

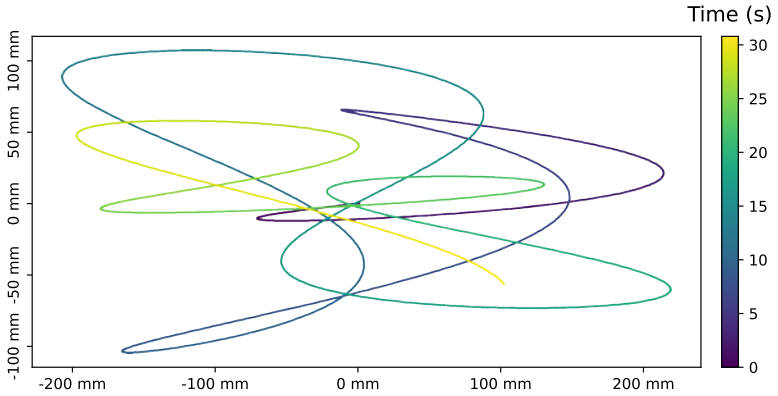

Figure 3.5: Example of a curve in the Consumer EEG-ET dataset that was presented to a participant. It was created using the parameterized function (3.1).

and y position of the dot in millimeters relative to the center of the screen were given by:

$$f : [0,1] \to \mathbb{R}^2, \quad t \mapsto \begin{pmatrix} (\cos(at)\sin(bt) + et) \cdot \frac{w}{2} \\ (\cos(ct)\sin(dt) + ft) \cdot \frac{h}{2} \end{pmatrix}, \tag{3.1}$$

where w and h are the width and height of the bounding box in millimeters in which the curve is drawn. The time in seconds the dot takes to traverse the complete curve can be controlled by the parameter T. The value of t is incremented by $\frac{1}{120}/T$ every frame, resulting in a smooth movement of the dot. For all level-1 experiments $a = b = c = d = 0$, $e, f \in \{-1, 0, 1\}$ and $T = 1$ or 2 depending on the direction. This results in paths where the dot moves continuously from the center to one of the edges of the specified bounding box. For level-2 experiments a, b, c, d were randomly selected from the interval $[-50, 50]$, $e = f = 0$ and $T = 28.5$. This results in paths where the dot moves in a more complex manner. An example of a path created using this method is shown in Figure 3.5. Multiple curves were shown in each experiment. Before the dot started to move along the next curve, the dot waited for 2 seconds in the center of the screen.

The grid point positions in the level-1 saccades and level-2 saccades experiments are shown in Figure 3.6. In a level-1 experiment, there was a saccade to the edge of the bounding box every second, followed by a saccade back to the center of the screen in the next second. In a level-2 experiment the dot jumped to any of the grid points every 1.5 seconds.

A new set of curves and new jump sequences was generated before every practice group. In a level-1 smooth experiment, only four possible curves existed, one for each direction. The order in which the curves were shown was determined by randomly sampling without replacement from a pool where every direction existed 4 times, resulting in a sequence of 16 curves in total. In a level-2 smooth experiment four curves were shown which were manually created by randomly picking values for a, b, c, d from the interval $[-50, 50]$. A level-1 saccades experiment was created by repeatedly sampling

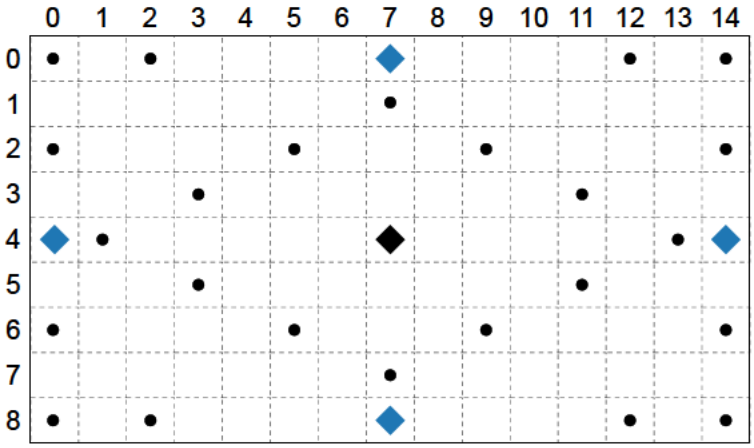

Figure 3.6: Grid point positions in the saccades experiments of the Consumer EEG-ET dataset. Blue diamonds are the grid points in the level-1 saccades experiment, black dots are the grid points in the level-2 saccades experiment. The black diamond in the center of the screen belongs to both experiments. In total there are 5 grid points in the level-1 saccades experiment and 25 grid points in the level-2 saccades experiment (both including the center).

without replacement from the four positions until 30 saccades were generated. Note that every saccade to the edge of the screen was followed by a saccade back to the center, which means that the dot jumped 60 times in total. In a level-2 saccades experiment saccades were generated by sampling without replacement from the 25 possible positions. This was repeated until 80 saccades were generated.

3.4 DATASET STRUCTURE

The dataset is available as a single denormalized Comma Separated Values (CSV) file, which contains all the data collected during the sessions. The CSV file has the following columns: `Participant_no`, `Task`, `Session_no`, `Timestamp`, `EEG_TP9`, `EEG_AF7`, `EEG_AF8`, `EEG_TP10`, `Presentation_State`, `Quality`, `Gaze_x`, `Gaze_y`, `Stimulus_x`, `Stimulus_y`. Any samples recorded before or after the stimulus presentation are removed.

In addition to the CSV file, a "csv" and an "xdf" folder are provided. Inside both folders, there are four subfolders, one for every experimental paradigm. Furthermore, every subfolder contains two more folders called "train" and "test". Those contain one file for every recording that can either be used for training a model or should be used to evaluate a model. In the "csv" folder, those files will be CSV files with the columns

```
Timestamp, EEG_TP9, EEG_AF7, EEG_AF8, EEG_TP10, Presentation_State,
Quality, Gaze_x, Gaze_y, Stimulus_x, Stimulus_y.
```

In the "xdf" folder, they will be Extensible Data Format (XDF) files with the raw recorded data. The filename will always follow the following naming scheme PXXX_YY.csv or PXXX_YY.xdf, where XXX is the participant number

and YY is the session number for this participant. Writing out the folder structure in a tree-like manner one would get the following:

```
(csv or xdf)/
    level-1-smooth/
        train/
            P001_01.(csv or xdf)
            P001_02.(csv or xdf)
            P002_01.(csv or xdf)
            ...
        test/
            P037_01.(csv or xdf)
            P042_01.(csv or xdf)
            ...
    level-1-saccades/
        ...
    level-2-smooth/
        ...
    level-2-saccades/
        ...
```

Both the CSV file containing the complete dataset, and the "csv" and "xdf" folders will be made freely available.

3.5 BENCHMARK

In addition to the dataset, a benchmark is proposed to evaluate the performance of existing and new methods on the Consumer EEG-ET dataset. The benchmark consists of four tasks, one for every experimental paradigm. The tasks are designed in a way that it becomes increasingly challenging to reconstruct the gaze position from the EEG data as the eye movement becomes less restricted.

The necessary data to benchmark a model on a given task is provided in the "csv" folder, which was described in the last section. The data is already split into train and test, with the twelve participants 21, 27, 35, 38, 46, 55, 60, 68, 73, 81, 84, 94 being used for testing and all other participants for training. The selection of the participants for the test data was based on the quality of the data, and the demographics of the participants. The participants were selected in a way that the test data is representative of the training data and recordings with low quality were excluded from the test data. Table 3.3 shows the exact proportions of the train and test data for every task.

A model is evaluated on a task by predicting the stimulus position (columns `Stimulus_x` and `Stimulus_y`) at every timestamp given the current and past EEG data (columns `EEG_TP9`, `EEG_AF7`, `EEG_AF8`, `EEG_TP10`). This is done for every recording in the test data. In order to simplify data pipelines the train and test data have the same columns. That means that the test data also includes the columns `Gaze_x` and `Gaze_y`, *i.e.* the gaze position as measured by the webcam. Those should not be used as input features.

Experimental Paradigm	Train Data Duration / Percentage	Test Data Duration / Percentage
level-1-smooth	1 h 38 min / 90%	11 min / 10%
level-1-saccades	1 h 44 min / 90%	12 min / 10%
level-2-smooth	3 h 42 min / 90%	26 min / 10%
level-2-saccades	3 h 28 min / 90%	24 min / 10%
Total	10 h 32 min / 90%	1 h 13 min / 10%

Table 3.3: Train and test data proportions for the benchmark tasks. The total duration is given as well as the percentage of the total data.

The performance of the model is evaluated using the Mean Euclidean Distance (MED) between the predicted and the ground truth stimulus position. The MED is calculated as:

$$\text{MED}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}, \tag{3.2}$$

where N_k is the number of samples in the k-th recording of the test data, x_i and y_i are the ground truth stimulus position and $\hat{x}_i$ and $\hat{y}_i$ are the predicted stimulus position. The final score of a model is the MED over all recordings in the test data, which is equal to:

$$\text{MED} = \frac{1}{\sum_{k=1}^{12} N_k} \sum_{k=1}^{12} N_k \text{MED}_k. \tag{3.3}$$

The MED was chosen as the evaluation metric because of its easy and clear interpretation. That is, the MED will tell you the average radius of a circle around the ground truth stimulus position in which the predicted stimulus position will lie. Other possible metrics would have been the Mean Squared Error (MSE), RMSE, or MAE, but compared to the MED they do not result in values that are easily interpretable, as they correspond to the mean of their respective one-dimensional metrics for the x- and y-axis.

Test data, even from other tasks, must never be used for training. Otherwise, the benchmark has no restrictions on training data. This means that training data from any or all tasks can be used to train a model for any task. When reporting the performance of a model, one should always report the MED for every task. This allows for a more nuanced comparison of the performance of different models, as models could overfit to one task and perform poorly on others, making them unsuited for general purpose eye tracking.

4 METHODS

This chapter will cover the methods used during experimentation. This includes the preprocessing of the data, the Functional Neural Network (FNN) architectures that were evaluated, and the metrics used to evaluate the performance of the models. We also cover how the two reimplemented methods, the SpatialFilterCNN and Battery Model of the Eye (BMOTE) were validated and go over the experimental setup.

4.1 PREPROCESSING

Not all the preprocessing steps described in this section were applied in all experiments, with the exception of windowing and missing value imputation, which were applied in all experiments. The exact preprocessing steps applied to the data are described in the respective sections of the experiments.

Manual Recording Exclusion

During the data recording process, live EEG measurements and gaze data (tracked via the webcam) were continuously monitored. This real-time monitoring enabled early detection of recording issues, allowing the task to be restarted when anomalies were observed. In certain cases, it was not possible to achieve reliable readings from specific electrodes. These instances, and similar issues, were documented during the recording sessions, and the corresponding recordings were excluded from all subsequent experiments. The full list of excluded recordings along with reasons for exclusion, is shown in Table 4.1. In total 14 recordings were excluded from the dataset due to various issues.

Recording(s)	Reason for Exclusion
P002_01	Webcam calibration quality declined significantly
P004_01	`Gaze_x` attained unrealistically high values
P016_01-P020_01	A lot of missing values due to bluetooth interference
P050_01 level-1-saccades	EEG disconnected before the recording finished
P062_01-P067_01	Very high AF8 readings due to faulty hardware
P079_01	High TP9 and TP10 readings from wearing a hijab

Table 4.1: List of recordings excluded from the dataset due to various issues.

Additionally, model experimentation began before the complete dataset was available, leading to all models being trained on a slightly smaller dataset than what is currently available. To maintain consistency and comparability across experiments, we continued using the smaller dataset for all experiments, even after the full dataset became available. As a result, participants 103-113 were excluded from the dataset used in these experiments.

Windowing

The models evaluated in this thesis received the EEG data in the form of windows. The length of the windows was controlled by a hyperparameter `window_size`. The windows were created by sliding a window of size `window_size` over the four EEG channels with a step size of 4 samples during training and 1 sample during validation and testing. At a sampling rate of 256 Hz this corresponds to steps of 15.625 ms and 3.906 ms respectively. The step size of 4 samples during training was chosen to reduce the amount of data that needed to be processed during training, effectively speeding up one epoch of training by a factor of 4 compared to training with a step size of 1, while still providing enough data for the model to learn from. Windows at the intersection of two recordings were discarded. The target was the last stimulus of the window

Missing Value Imputation

Because missing values were recorded as zeros, which are also valid values, the first step was to distinguish actual missing values from legitimate zero readings. To address this, zero values that occurred as part of an uninterrupted sequence of at least three zeros were identified as missing values. That means a zero was flagged as missing if it was preceded, followed, or surrounded by two other zeros. This approach was adopted to ensure that isolated zero values were retained as valid data points while sequences of zeros (which are unlikely to occur naturally in the data) were treated as missing data.

Once the missing values were identified, they were imputed using a Kalman smoother. In this case, an Seasonal Autoregressive Integrated Moving Average (SARIMA) model was used as the underlying state-space model for the Kalman smoother. The optimal SARIMA model for each recording was determined automatically using the `auto_arima` function from the `pmdarima` library. This function performs differencing tests to decide the order of differencing required for stationarity, together with a stepwise algorithm, as outlined by [22], to select the p, q, and seasonal P and Q orders based on the Akaike Information Criterion.

The `auto_arima` function was applied with default parameters, with the exception of the seasonal lag, which was set to 5. This choice was made due to the particularly strong presence of the 50 Hz powerline noise signal in

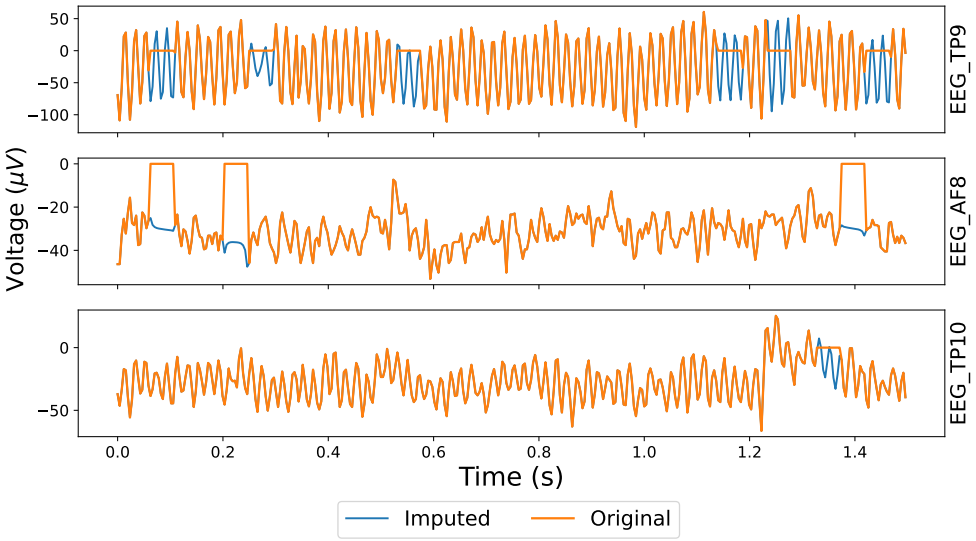

Figure 4.1: Effect of missing value imputation on the example of recording "P045_01 level-1-saccades". No missing values were present in electrode AF7, which is why it is not shown.

the data, which, at a sampling rate of 256 Hz, corresponds to a period of approximately 5 samples ($256/50 = 5.12 \approx 5$).

Figure 4.1 illustrates the effect of the missing value imputation on the data.

Filtering

As EEG data is highly prone to noise from various different sources, filtering is a common preprocessing step in EEG analysis. However, filtering turned out to be only sometimes beneficial, which is why it was not applied in all experiments. When the preprocessing included filtering, a combination of multiple filters was applied to the data. First a 60 Hz notch filter was applied to remove the noise from the monitor refresh rate. This was followed by another 50 Hz notch filter to remove the noise from the power lines. Finally a 0.5 Hz to 40 Hz butterworth bandpass filter was applied to remove noise from other sources (*e.g.* muscle activity, baseline drift, *etc.*). All filters were applied forwards and backwards to avoid phase distortion.

The filters were implemented using second-order sections instead of directly applying the difference equation, as this was found to be more stable during testing. In Figure 4.2 the effect of applying the filters on the data is shown. The result of filtering without second-order sections is also shown.

Even though the bandpass filter already limits the frequency range of the data between 0.5 Hz and 40 Hz — which should already exclude the 50 Hz and 60 Hz noise from the power lines and monitor — we have found that applying the bandpass alone was insufficient to remove the noise from the power lines. This can be seen in Figure 4.3 where the noise from the power lines is still present in the data after applying the bandpass filter. Filtering

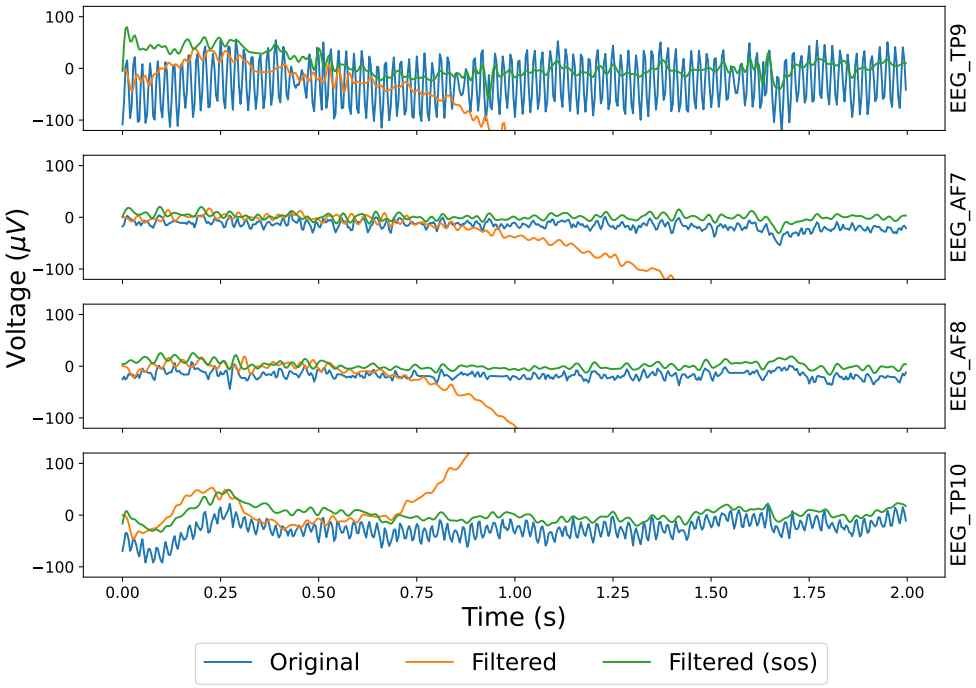

Figure 4.2: Effect of filtering on recording "P045_01 level-2-saccades", where missing values were imputed as described in the above section. It is evident that the filter without second-order sections causes the signal to diverge over time, while the filter with second-order sections keeps the signal stable. Both filters used the combination of a 60 Hz and a 50 Hz notch filter and a 0.5 Hz to 40 Hz bandpass filter as described in the text. However, the first filter used an 8th order bandpass filter and applied all filters only forwards, as applying them backwards as well would cause the signal to attain extreme values throughout. The filter with second-order sections used a 4th order bandpass filter and applied all filters both forwards and backwards doubling the order of the filter and correcting phase shift.

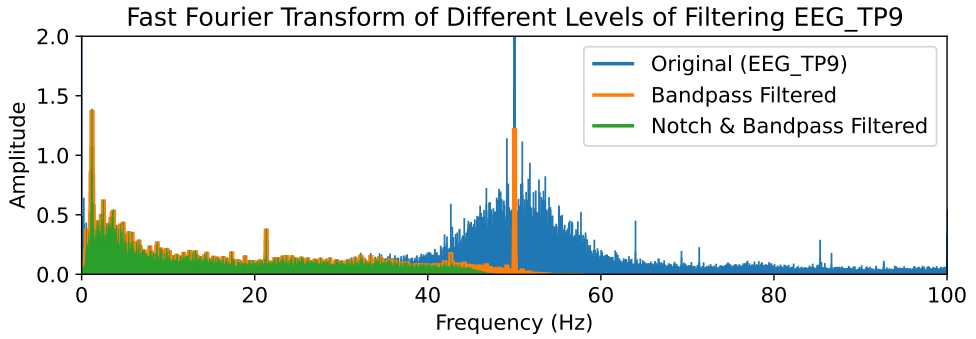

Figure 4.3: Effect of the bandpass filter on the TP9 electrode of recording "P045_01 level-2-smooth". The noise from the power lines is still clearly present in the data after applying the bandpass filter. In order to better see the bandpass filtered data (orange), the line width was increased.

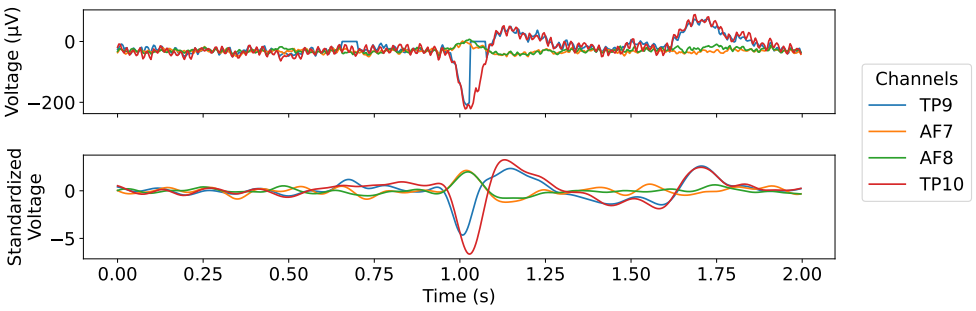

Figure 4.4: A visual comparison of the signal before and after filtering. The window contains a blink right in the middle, which does not get lost after filtering.

was applied after the missing value imputation step but before the windowing step for better runtime performance and to avoid edge effects.

Blink Detection

Participants were allowed to blink naturally during the recordings. For tasks involving smooth and continuous eye movements — particularly the level-2 tasks, which required 30 seconds of uninterrupted movement — expecting participants to refrain from blinking for such extended periods would have been unreasonable. Also, allowing blinks more accurately reflects real-world conditions, which makes testing more representative of the practical application. However, the presence of blinks might complicate the analysis, particularly in the early stages of model training. Thus a method was developed to automatically label blinks.

Manual labelling of blinks was not feasible due to the large amount of data collected. As an alternative, a blink detection model was developed that was trained on a specially collected dataset in which the author was the sole participant. This model is a simple neural network that works in a similar manner to template matching algorithms.

The custom dataset consists of various 5-minute recording sessions. During these sessions, the head and eyes were maintained in different states, including relatively stable positions, movement of the eyes to their extreme positions within the socket, random eye movements, and tilting the head. Throughout these recordings, blinks happened randomly, and the occurrence of each blink was manually marked in real-time. Subsequently, with the help of the blink markers, the exact blink ranges were determined through visual inspection of the data, and annotated and saved in the XDF file format using custom software.

For each recording extensive filtering was applied, including a bandpass filter ranging from 1 Hz to 10 Hz and notch filters at 50 Hz and 60 Hz, to enhance the clarity of the signal relevant for blink detection. Additionally, all EEG channels were standardized to have mean 0 and a standard deviation of 1. A visual comparison of the signal before and after filtering and stan-

Layer	Parameters	Output Shape
Input	—	(batch_size, 128, 4)
Conv1D	padding: same, activation: elu, filters: 1, kernel_size: 64	(batch_size, 128, 1)
Flatten	—	(batch_size, 128)
Dense	activation: sigmoid	(batch_size, 1)

Table 4.2: The general structure of the blink detection models. Layer and parameter names generally follow the naming conventions of the TensorFlow library.

dardization can be seen in Figure 4.4. The data was split into a training and a test set using a 80% / 20% split. Windows of size 128 samples, with a step size of 1 sample were used, following the procedure described in the section *Windowing*.

The final model used a quite simple architecture that is shown in Table 4.2. This architecture effectively serves as a (simple) extension of a template matching algorithm, where the template, *i.e.* the filters in the convolutional layer, are learned from the data and matches can be differently weighted by the dense layer. We also experimented with an architecture using a functional convolutional layers, followed by a global average pooling layer and a sigmoid activation, resulting in an architecture even more similar to a template matching algorithm.

Both models were trained for 2 epochs with binary crossentropy as the loss function. Upon evaluation, the convolutional model outperformed the functional convolutional model, achieving a recall of 0.99 and a precision of 0.93 at a threshold of 0.5 on the test set. Although the functional model produced much more interpretable filters, the superior accuracy of the standard convolutional model made it the preferred choice for further analysis. However, near perfect accuracy is not expected, as the true blink ranges are manually labeled and will not always be perfectly accurate. Therefore, very high accuracy could actually indicate overfitting. On the other hand, it is important to note that blink detection is a relatively straightforward task, as blinks are highly distinctive in the data and despite the presence of noise remain clearly visible, even to the human eye (see Figure 4.4). This generally allows for high accuracy.

Figure 4.5 displays the learned filters for both models, while Figure 4.6 illustrates a segment of the data with detected blinks.

Once the model was validated, it was applied to the actual EEG-ET data, where the blink probability predicted by the model was added as an additional feature to the data. This could then be used by the downstream data processing pipeline to remove windows with a high probability of containing a blink.

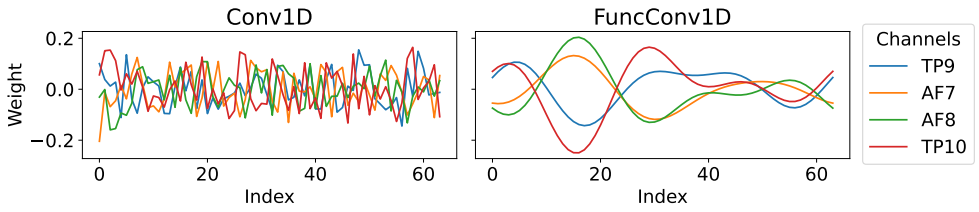

Figure 4.5: Learned filters of the blink detection models. The filters of the functional model clearly resemble the shape of a blink in the EEG data (*e.g.* the one in Figure 4.4), while the filters of the convolutional model are not interpretable.

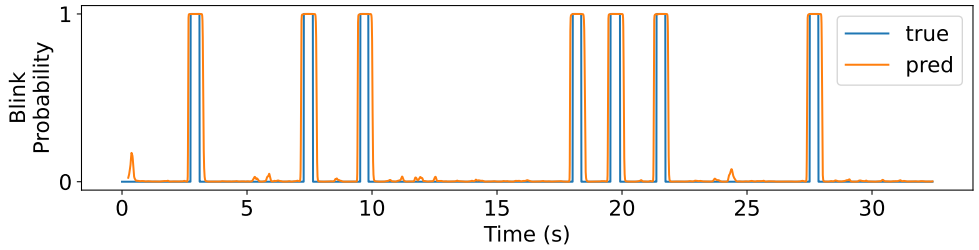

Figure 4.6: A segment of the data with detected blinks. The true blink probability (which is either 100% or 0%) is shown in blue, while the predicted blinks are shown in orange. The accuracy will never be 100%, as the true blink ranges are manually labeled.

4.2 VALIDATION OF MODEL IMPLEMENTATIONS

Both the SpatialFilterCNN and BMOTE were implemented from scratch in Python. To ensure the accuracy and reliability of our implementations, we conducted a series of validation tests for each model.

Validation of the SpatialFilterCNN Model

The implementation of the SpatialFilterCNN was relatively straightforward to implement using the deep learning libraries TensorFlow and Keras, as the model architecture can be constructed using standard building blocks like convolutional layers, dense layers, max pooling layers, *etc.* However, certain implementation details were not explicitly stated in the original paper, including

1. The mode of padding used ("valid" or "same")
2. The stride of the pooling layers
3. The method for transforming the input to match the correct dimensions for addition to the output of the residual blocks

To address these gaps, we made informed decisions based on common practices in CNN design and the parameter count stated in the original paper:

1. We opted for "same" padding such that the sample size of the output of a residual block matched the input size.

2. For the pooling layers, we used a stride of 2, reducing the input size by a factor of 2 at each pooling step.

3. To transform the input for the residual blocks, we added a convolutional layer with a kernel size of 1 to the input branch of each residual block.

These choices allowed us to successfully replicate the parameter count of $2.06 \cdot 10^6$ stated in the original paper, providing a good indication that our implementation closely matched the intended architecture.

To properly validate our implementation, we aimed to replicate the results reported in the original paper [15] that introduced the SpatialFilterCNN. In that paper, the model was evaluated on the EEGEyeNet dataset using the MAE metric. The first step to do this was to set up the EEGEyeNet Benchmark environment and ensure its correct functionality. Fortunately, the authors of the EEGEyeNet paper provided extensive code for this purpose, available on GitHub at `https://github.com/ardkastrati/EEGEyeNet`.

After setting up the benchmark environment, our first step was to test the deep learning models included in the repository. This initial assessment was important to verify the overall functionality of the environment, before proceeding with our own implementation. However, this process presented several unexpected challenges. The provided code did not calculate results in millimeters, and the conversion method from pixels to millimeters was not clearly documented. There was ambiguity regarding which specific metric was being reported in the original paper.

After extensive testing, we determined that a conversion factor of 0.5 px/mm briefly mentioned in the appendix of the EEGEyeNet paper, allowed us to replicate the naive baseline (predicting the mean of the training data) when used in conjunction with the MED metric. Using this established conversion factor, we were able to successfully replicate the naive baseline results.[1]

However, our implementation of the other deep learning models, including the SpatialFilterCNN, produced results that differed significantly from those reported in the original EEGEyeNet paper.[2] Table 4.3 presents a comparison between the original results reported in the EEGEyeNet paper and the results obtained from our implementation. The results for the MED metric with a conversion factor of 0.5 are shown on the left side of the table.

Despite these challenges, we proceeded to benchmark our implementation of the SpatialFilterCNN model on the EEGEyeNet dataset. For this evaluation, we used the MAE metric, consistent with the approach in [15]. The results of this benchmark are presented on the right side of Table 4.3. These

1 Attempting to derive a conversion factor using the screen dimensions in pixels and inches yielded a different factor (0.6096 px/mm) that did not reproduce the reported naive baseline values using any of the possible metrics.

2 This was true for all attempts of pixel to mm conversion and choice of metric.

Model	MED	MED (ours)	MAE	MAE (ours)
CNN	70.2 ± 1.1	205.4 ± 6.1	86.6	129.9 ± 4.3
PyramidalCNN	73.6 ± 1.9	101.5 ± 2.4	90.3	64.9 ± 1.2
EEGNet	81.7 ± 1.0	77.3 ± 0.3	97.0	48.7 ± 0.2
InceptionTime	70.8 ± 0.8	114.8 ± 3.3	88.4	71.8 ± 1.8
Xception	78.7 ± 1.6	116.48 ± 5.3	94.7	72.8 ± 2.9
SpatialFilterCNN	—	68.8 ± 1.4	49.2	42.9 ± 0.8
Naive Baseline	123.3	123.3	—	80.4

Table 4.3: Comparison of the original results and the results obtained by us for the Absolute Position task of the EEGEyeNet Benchmark. The results for the MED metric with a conversion factor of 0.5 are shown on the left side of the table together with the results from the EEGEyeNet paper [23]. The results for the MAE metric, also with a conversion factor of 0.5, are shown on the right side of the table together with the results from the SpatialFilterCNN paper [15] in which no standard deviations were reported.

results nearly matched those reported in the SpatialFilterCNN paper [15]. The remaining differences could potentially be attributed to a different conversion factor used for the metrics, as the SpatialFilterCNN paper also did not explicitly state the conversion factor they employed.

Notably, our implementation of the SpatialFilterCNN performed significantly better than the other models in the benchmark. This substantial performance gap provided us with confidence in the correctness of our implementation. It is worth mentioning that the discrepancies in results for the other models included in the EEGEyeNet benchmark remain unexplained. Despite our efforts, we were unable to replicate the results reported in the EEGEyeNet paper for these models.

Validation of the Battery Model of the Eye

The implementation of the BMOTE is considerably more complex, requiring a lot of custom code, rather than relying on standard deep learning libraries. Initially, our plan was to use the original implementation of BMOTE rather than creating our own. We reached out to the authors of the original paper requesting access to their implementation. However, they informed us that their code was not currently packaged in a way that would allow for easy execution. Despite this setback, the authors generously offered their assistance with the implementation process.

Given this situation, we proceeded to implement BMOTE ourselves, with the intention of testing the implementation by estimating electrode positions and evaluating the model's performance using the dataset from the original paper. However, we encountered another obstacle: the authors of the original paper were unable to provide us with their dataset. As an alternative, the authors referred us to a dataset from a follow-up paper [3].

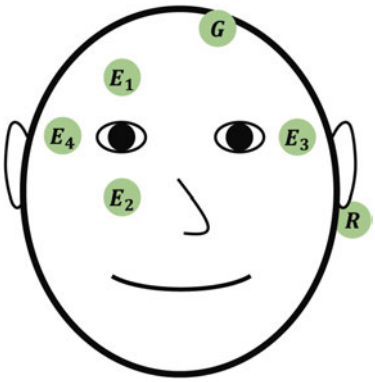

Figure 4.7: Schematic of the electrode positions used in the EOG dataset. Image taken from [3]. The ground electrode is labeled as "G" and the reference electrode as "R".

Specifically, they directed us to Dataset 2 available at `https://www.um.edu.mt/cbc/ourprojects/eyecon/eogdataset`. This dataset became our primary resource for testing the implementation of the BMOTE model.

Given the dataset, we first checked if the electrode positions estimated by our implementation aligned with the positions from the dataset specifications. For this, we first obtained values for the radius of the eye and the interpupillary distance, which are needed for the electrode position estimation. Unfortunately, the authors did not provide these values in the paper, we instead had to base these on publicly available averages.

For this we used [5], where the authors concluded that the the human eye is approximately 24.2 mm wide, 23.7 mm high and 22.0 mm to 24.8 mm deep, with no significant difference between sexes and age groups. Calculating the mean of the width, height, and the average depth, we get a diameter of approximately 24 mm or a radius of 12 mm. The interpupillary distance is assumed to be 62 mm [18], based on the average of the median values of males (64 mm) and females (62 mm). With these values, we could then estimate the positions of the electrodes.

The electrode placement according to the specifications is depicted in Figure 4.7. Using the interpupillary distance and the average eye radius, we calculated rough theoretical positions for each electrode. In those calculations we assumed that the electrodes were placed at a distance of 24 mm — the average eye diameter — away from the center of the eye. Table 4.4 presents a comparison between these calculated positions and the positions estimated by our BMOTE implementation using the first half of the dataset.

Looking at the table, we see that the estimated positions are fairly close to the theoretical positions, with most differences being less than 10 mm. After successfully estimating the electrode position, we were able to create the plot shown in Figure 4.8. In this figure, the predicted and actual voltages are plotted against the horizontal and vertical gaze angle. We can see how the model voltage adequately follows the true voltage readings, which nearly grow linearly with the gaze angle.

Position	Estimated	Theoretical	Difference
Electrode 1 (above right eye)	$\begin{pmatrix} 8.71 \\ 32.70 \\ 32.92 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 31.5 \\ 24 \end{pmatrix}$	$\begin{pmatrix} 8.71 \\ 1.20 \\ 8.92 \end{pmatrix}$
Electrode 2 (below right eye)	$\begin{pmatrix} 27.93 \\ 25.90 \\ -31.61 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 31.5 \\ -24 \end{pmatrix}$	$\begin{pmatrix} 27.93 \\ -5.60 \\ -7.61 \end{pmatrix}$
Electrode 3 (right of right eye)	$\begin{pmatrix} 4.09 \\ -58.83 \\ -3.54 \end{pmatrix}$	$\begin{pmatrix} 0 \\ -55.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 4.09 \\ -3.33 \\ -3.54 \end{pmatrix}$
Electrode 4 (left of left eye)	$\begin{pmatrix} 7.09 \\ 56.71 \\ -4.99 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 55.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 7.09 \\ 1.21 \\ -4.99 \end{pmatrix}$

Table 4.4: Comparison of the theoretical electrode positions and the estimated positions from the BMOTE model.

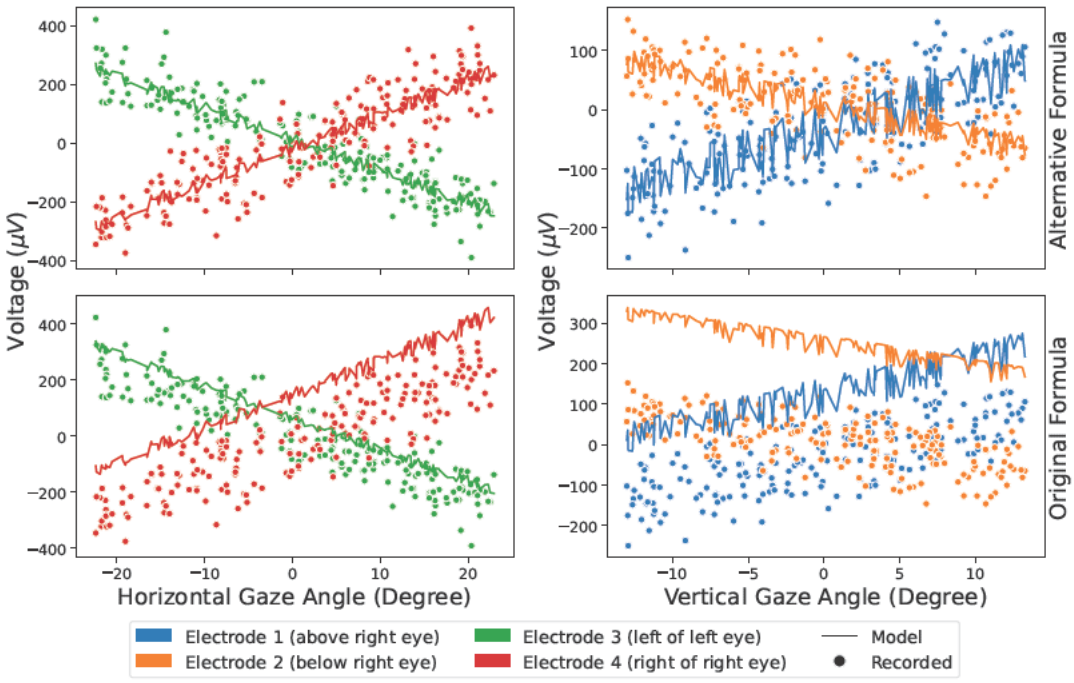

Figure 4.8: Recorded voltages vs model voltages from the BMOTE. In each panel the dots correspond to recorded voltages at a certain gaze angle, while the line shows the voltages the model would predict at the same gaze angle. On the left side voltages are plotted against the horizontal gaze angle, on the right side against the vertical gaze angle. In the top row the model uses electrodes estimated with the alternative formula (2.4), in the bottom row the model uses the electrodes estimated with the formula (2.5) from the paper.

Model	MAE Horizontal Gaze Angle	MAE Vertical Gaze Angle
Original [2]	2.42 ± 0.91	2.30 ± 0.50
Follow-up [3]	2.89	7.30
Ours	5.83	6.49

Table 4.5: The MAE of the horizontal and vertical gaze angles in degrees of visual angle. Our results, along with those from the follow-up study [3], are for Subject 3 of the dataset used in that study. The results from the original paper [2] represent the mean and standard deviation across all participants of a different dataset.

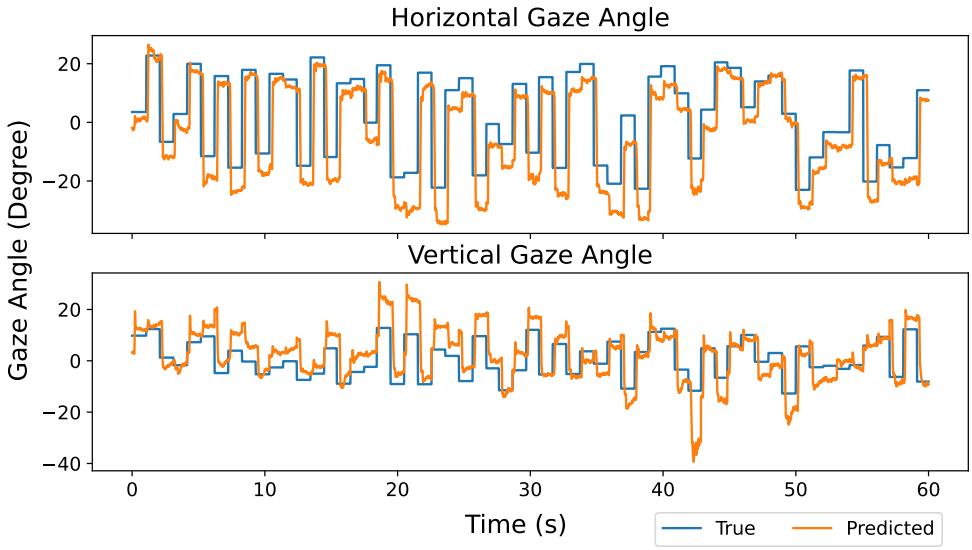

Figure 4.9: True vs Predicted Gaze Angles for the BMOTE Model. Data is from Subject 3.

We also see how the estimated electrode positions using formula (2.4) differ from the electrode positions calculated using the formula (2.5) provided in the paper. The latter resulting in the correct shape for the model voltage, but shifted up or down.

To complete our validation process, we evaluated the BMOTE model following the methodology described in [2]. This evaluation consisted of the following steps: We performed 2-fold cross-validation. The baseline drift was removed by fitting and subtracting a 20th order polynomial and gaze errors were computed based on the difference between the true gaze angle and the mean of all predictions during a fixation. The resulting horizontal and vertical errors, measured in degrees of visual angle, are presented in Table 4.5. This table also includes the results from the original paper and comparable results from a follow-up paper for comparison.

From the table, one can see that our horizontal gaze error is larger than reported in the referenced papers. The vertical gaze error fits with the one reported in the follow-up paper but is larger than the one reported in the

original paper. However, an exact comparison is difficult, as the results from the original paper are for a different dataset than the one we used, and the results from the follow-up are based on a more advanced model. We acknowledge that those results are not perfect, but combined with the reasonable electrode position estimation in Table 4.4, and the close alignment between the model's predicted and recorded voltages in Figure 4.8, we are confident that the model is implemented correctly. This is further supported when we look at the true vs. predicted gaze angles in the test data, as shown in Figure 4.9, where we can see that the model's predictions follow the true gaze angles.

4.3 FUNCTIONAL NEURAL NETWORK ARCHITECTURES

Because FNNs are rather new and effective design practices are not yet established, we relied on established CNN architectures as a reference point to guide the development of our FNN architectures. These architectures typically consist of three main components: the *stem*, *body*, and *head*.

Stem: The stem is the initial part of the network responsible for converting the input signal into a form that can be processed by the subsequent layers. In a traditional CNN architecture, the stem often includes a convolutional layer with a large kernel size, followed by a pooling layer to reduce the spatial dimensions of the input. For our FNNs, we replaced the standard convolutional layer with a functional convolutional layer, using a large resolution to mimic the large kernel size of conventional CNNs. In addition, we preceded this layer with a spatial filtering layer, inspired by early experiments with the SpatialFilterCNN, where it was found to be beneficial for enhancing the model's performance. This design formed the foundation of the stem used in all the architectures we explored. The general structure of the stem is shown in Table 4.6.

Body: The body of the network is where the bulk of the computation takes place. It is composed of multiple stages, each consisting of several blocks. For two-dimensional signals (like images) a typical block is structured as a sandwich of three convolutional layers, where the outer two use 1×1 kernels, and the middle layer uses a 3×3 kernel. In our architectures, we decided to omit the first 1×1 convolution following insights from the SpatialFilterCNN [15]. Since we are working with one-dimensional signals, we replaced the 3×3 kernel with a kernel size of 9. Each convolutional layer was followed by batch normalization before applying the activation function, which is a common practice. Additionally, we incorporated residual connections within each block, allowing the input to be added to the output of the last convolutional layer. The resulting block structure is shown in Table 4.7.

As in CNNs, each stage of our FNN body concludes with a pooling layer, that reduces the length of the input signal by a factor of two,

effectively narrowing the signal as it progresses deeper into the network. Concurrently, the number of filters in the convolutional layers was increased at each stage, resulting in a deeper network.

Head: In a typical CNN, the head consists of one or more dense layers that transform the output of the body into the final prediction. To accommodate this, the output from the body must first be flattened or aggregated (*e.g.* through global average pooling).

Based on this reference architecture, we designed three FNNs and evaluated their performance. We did this by replacing different parts of the architecture with functional layers. The functional layers are implemented using the code published by one of the authors of [20], which is available on GitHub at `https://github.com/FlorianHeinrichs/functional neural networks`. All architectures are sized to have a similar number of parameters of around 1.2×10^6, resulting in models of approximately 4 MB in size. This way the models are comparable in terms of complexity.

Layer	Parameters	Output Shape
Input	—	(batch_size, window_size, 4)
Conv1D	kernel_size: 1, filters: 16	(batch_size, window_size, 16)
BatchNorm	axis: -1	(batch_size, window_size, 16)
ReLU	—	(batch_size, window_size, 16)
FuncConv1D	padding: same, resolution: 128, n_functions: 9, basis_type: Fourier	(batch_size, window_size, 64)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/2, 64)

Table 4.6: The general structure of the stem, which is used by all three FNN architectures. Layer and parameter names generally follow the naming conventions of the TensorFlow library. FuncConv1D is a functional convolutional layer. The parameters are explained in Section 2.4.

Layer	Parameters	Output Shape
Input	—	(batch_size, steps, channels_in)
(Func)Conv1D	padding: same, filters: channels_out (*standard only*) kernel_size: 9 (*functional only*) resolution: 24, n_functions: 6, basis_type: Legendre	(batch_size, steps, channels_out)
BatchNorm	axis: -1	(batch_size, steps, channels_out)
elu	—	(batch_size, steps, channels_out)
Conv1D	padding: same, filters: channels_out, kernel_size: 1	(batch_size, steps, channels_out)
BatchNorm	axis: -1	(batch_size, steps, channels_out)
Add	—	(batch_size, steps, channels_out)
elu	—	(batch_size, steps, channels_out)

Table 4.7: The structure of a standard ResBlock and a FuncResBlock. Layer and parameter names generally follow the naming conventions of the TensorFlow library. In the ResBlock, the (Func)Conv1D layer is a standard convolutional layer with a kernel size of 9. In the FuncResBlock, the (Func)Conv1D layer is a functional convolutional layer with the parameters listed under *(functional only)*. In both cases, the "same" padding and a total of "channels_out" filters are used.

Fully Functional Architecture

The first FNN follows a "fully functional" design, meaning that every component in the network is functional. For this, the residual block from the reference architecture is replaced by a "functional residual block", where the first convolution is changed to a functional convolutional layer. The second convolution, which uses a kernel size of 1, remains unchanged, as a functional convolutional layer with resolution 1 is only a complicated way of applying a standard convolutional layer. The functional residual block is shown together with the standard residual block in Table 4.7. Parts that differ between the two blocks are marked accordingly.

Multiple of such functional residual blocks are then chained together making up one big stage, with the number of filters increasing progressively with the depth of the network. No pooling layers are used in or after the stage. There are two reasons for this: Firstly, pooling operations with a stride break the smooth structure of the signals passing through the network. Secondly, using only functional layers at the head avoids the "parameter explosion"

that would normally occur after the flattening operation. This is explained in more detail below.

In the head of the network, solely functional dense layers are employed. Because these layers expect functional inputs, there is no need to flatten the output of the body. Instead, the body's output is fed directly to the head. This leads to a significant reduction in the number of parameters. Without the need to flatten the body's output, the first dense layer in the head requires only `last_channels_out` $\times$ `neurons` weights. In contrast, if the output had been flattened, the number of weights would be `last_channels_out` $\times$ `last_steps` $\times$ `neurons`, where `last_steps` refers to the number of time steps remaining after the body.

For instance, with a window size of 512 and no pooling, the `last_steps` would be 512. Assuming `last_channels_out` is 256 and there are 256 neurons in the first dense layer, this requires only $256 \times 256 = 65,536$ weights, which equates to approximately 257 KB, assuming 4-byte floats. In contrast, if the output had been flattened, the network would require $512 \times 256 \times 256 = 33,554,432$ weights, or approximately 128 MB with 4-byte floats.

The complete architecture of the first "fully functional" FNN is shown in Table 4.8.

Architecture #1: Fully Functional

Layer	Parameters	Output Shape
STEM	—	(batch_size, window_size/2, 64)
FuncResBlock	filters: 64	(batch_size, window_size/2, 64)
FuncResBlock	filters: 96	(batch_size, window_size/2, 96)
FuncResBlock	filters: 144	(batch_size, window_size/2, 144)
FuncResBlock	filters: 216	(batch_size, window_size/2, 216)
FuncDense	neurons: 256, n_functions: 12, basis_type: Legendre, activation: elu	(batch_size, window_size/2, 256)
FuncDense	neurons: 2, n_functions: 12, basis_type: Legendre, activation: linear, pooling: True	(batch_size, 2)

Table 4.8: The detailed structure of the first "fully functional" architecture. FuncDense is a functional dense layer. The parameters are explained in Section 2.4. This model has 1,150,488 trainable parameters, amounting to 4.39 MB.

Functional Body Architecture

The second FNN architecture takes a hybrid approach, using functional layers only in the body while concluding with standard dense layers in the head. Unlike the fully functional architecture, using standard dense layers

in the head necessitates pooling layers at the end of each stage, to reduce the number of steps flowing into the head.

The body of this architecture is structured into two stages, each composed of two functional residual blocks. In the first stage, each block contains 64 filters, while in the second stage, the number of filters is increased to 112. Pooling operations at the end of each stage reduce the input size by half, helping to control the dimensionality of the data as it passes through the network.

The output of the body is then flattened and passed through the head, which consisted of two standard dense layers. The first dense layer contains 64 neurons with an Exponential Linear Unit (ELU) activation function, while the second layer has two neurons with a linear activation function, producing the final output.

Table 4.9 provides a detailed breakdown of the second FNN architecture.

Architecture #2: Functional Body

Layer	Parameters	Output Shape
STEM	—	(batch_size, window_size/2, 64)
FuncResBlock	filters: 64	(batch_size, window_size/2, 64)
FuncResBlock	filters: 64	(batch_size, window_size/2, 64)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/4, 64)
FuncResBlock	filters: 112	(batch_size, window_size/4, 112)
FuncResBlock	filters: 112	(batch_size, window_size/4, 112)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/8, 112)
Flatten	—	(batch_size, window_size/8 × 112)
Dense	neurons: 64, activation: elu	(batch_size, 64)
Dense	neurons: 2, activation: linear	(batch_size, 2)

Table 4.9: The detailed structure of the second "functional body" architecture. Layer and parameter names generally follow the naming conventions of the TensorFlow library. This model has 1,157,394 trainable parameters, amounting to 4.42 MB.

Minimally Functional Architecture

The third FNN architecture uses functional layers sparingly, with only a single functional dense layer in the head. [3] This functional layer helps avoid the need to flatten the body's output, which similar to the first architecture, reduces the number of required parameters. The "saved" parameters are reallocated to an additional larger dense layer in the head, rather than adding more blocks to the body. This approach introduces some architectural vari-

3 And the one functional convolutional layer in the stem that is part of all three architectures.

ety. The body of this architecture is similar to that of the second FNN, but it employs standard residual blocks rather than functional ones. At the head, the output from the body is first aggregated by a functional dense layer with 512 neurons. This layer includes pooling and uses the ELU activation function. It acts as a more flexible global average pooling layer, providing the flexibility to weigh different parts of the signal differently, as opposed to standard global average pooling, which averages all inputs equally. After the functional layer, the data is processed by a standard dense layer with 512 neurons, also using the ELU activation function. The architecture concludes with a final standard dense layer with two neurons and a linear activation function for the output. The details of the third FNN architecture are shown in Table 4.10.

Architecture #3: Minimally Functional

Layer	Parameters	Output Shape
STEM	—	(batch_size, window_size/2, 64)
ResBlock	filters: 64	(batch_size, window_size/2, 64)
ResBlock	filters: 64	(batch_size, window_size/2, 64)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/4, 64)
ResBlock	filters: 112	(batch_size, window_size/4, 112)
ResBlock	filters: 112	(batch_size, window_size/4, 112)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/8, 112)
FuncDense	neurons: 512, activation: elu, n_functions: 12, basis_type: Legendre, pooling: True	(batch_size, 512)
Dense	neurons: 512, activation: elu	(batch_size, 512)
Dense	neurons: 2, activation: linear	(batch_size, 2)

Table 4.10: The detailed structure of the third "minimally functional" architecture. Layer and parameter names generally follow the naming conventions of the TensorFlow library. FuncDense is a functional dense layer. The parameters are explained in Section 2.4. This model has 1,275,570 trainable parameters, amounting to 4.87 MB.

4.4 EXPERIMENTATION SETUP

The experiments were conducted on an NVIDIA DGX Workstation, the hardware specifications of which are outlined in Table 4.11. The system features four NVIDIA Tesla® V100-DGXS GPUs, each with 32 GB of memory, supported by an Intel Xeon E5-2698 v4 CPU running at 2.2 GHz with 20 physical

Component	Specification
GPUs	4 × NVIDIA Tesla® V100-DGXS-32GB
CPU	1 × Intel Xeon E5-2698 v4 2.2 GHz (20-Core/40 vCores)
RAM	256 GB ECC Registered-DIMM DDR4 SDRAM
OS	DGX OS 5.4.2 (Ubuntu 20.04)
CUDA	11.4

Table 4.11: Hardware specifications of the NVIDIA DGX Workstation used for the experiments.

cores, and 40 virtual cores. Additionally, the workstation is equipped with 256 GB of RAM.

While the workstation has the capacity to run multiple GPUs simultaneously, initial tests revealed that, using multiple GPUs did not result in a significant reduction in training time. This is likely due to the bottleneck in data loading. As a result, each experiment was conducted on a single GPU. However, multiple GPUs were used to run several experiments concurrently.

All experiments were conducted within a Docker container environment. Specifically, we utilized the NVIDIA TensorFlow container image `nvcr.io/nvidia/tensorflow:24.06-tf2-py3`, which provided an optimized runtime for TensorFlow with CUDA 11.4 support.

For experiment tracking, we utilized *MLflow*, which was self-hosted on a separate machine dedicated to logging and monitoring experimental data. Most of the key metrics and parameters were captured automatically using MLflow's autologging feature. In addition to the standard metrics, we manually logged various other metrics including the following:

- The commit hash to ensure that the exact version of the code used for each experiment was traceable.
- A fingerprint or hash of the dataset, along with accompanying metadata, to guarantee data reproducibility and traceability.
- The complete experiment configuration, including hyperparameters and other settings.
- The serialized model in Keras format (model.keras) for future inference or tests.
- Various plots, including visualizations of the dataset and model predictions.

Hyperparameter tuning was handled using *Optuna*, with the Tree-structured Parzen Estimator (TPE) sampler configured for efficient search through the hyperparameter space. Additionally, Optuna was set up to allow for the saving, loading, and continuation of tuning sessions, ensuring flexibility in managing the optimization process.

4.5 METRICS

In this section, we introduce the three key metrics used to evaluate model performance in our experiments: the Mean Euclidean Distance (MED), Precision, and the Pearson correlation coefficient. These metrics provide complementary insights into different aspects of model accuracy, helping us assess the quality of predictions across both smooth pursuit and saccadic eye movements.

Mean Euclidean Distance

In line with the benchmarking guidelines, introduced in Section 3.5, we collected the MED in all experiments. It is calculated using the formula given in Equation 3.2 and Equation 3.3. It is clear and easily interpretable, and can be used for both the saccades and smooth pursuit tasks.

Precision

In conjunction with the MED, we also calculated the precision of the model's predictions. However, here "Precision" does not refer to the precision in classification tasks, but rather to the variation in the predicted gaze positions. This "version" of the Precision metric is common in the field of metrology [44], but can also be found in related works such as [40]. The metric is calculated by taking the mean of all successive predictions differences, *i.e.*

$$\frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_{i+1} - \hat{y}_i\|_2 \,, \tag{4.1}$$

where $\hat{y}_i$ is the predicted gaze position at time step i. We adjusted this precision metric to account for moving targets. Specifically, in cases where the target gaze position is dynamic, the above formula will be incorrect. Therefore, we modified the metric to only consider deviations in predictions that are not caused by the motion of the target. The modified formula is given by

$$\text{Precision} := \frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_{i+1} - \hat{y}_i - (y_{i+1} - y_i)\|_2 \,, \tag{4.2}$$

where y_i is the true gaze position at time step i.

Correlation

We also measure the Pearson correlation coefficient, which is defined as

$$\text{corr} := \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2 \sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}}, \tag{4.3}$$

where y_i and $\hat{y}_i$ are the individual data points for the true and predicted values, and $\bar{y}_i$ and $\bar{\hat{y}}_i$ are their respective means. It measures the linear relationship between the two variables, resulting in a value between -1 and 1, with 1 indicating perfect correlation.

Since the Pearson correlation can only be computed for one-dimensional variables, we calculate the correlation separately for the x and y components of the predicted gaze positions. These separate metrics are referred to as corr_x and corr_y, respectively. To obtain a single combined correlation metric, which can be used for tasks such as hyperparameter tuning, we compute the mean of corr_x and corr_y. This combined metric is referred to simply as corr in our experiments.

While the MED serves as a primary measure of model performance, we introduced the Pearson correlation coefficient to complement the MED due to its additional, desirable properties in certain scenarios. One advantage of the correlation metric is that it easily detects when a model has "collapsed to the mean." When all predictions are the same (such as when a model predicts only the mean value from the training data), the Pearson correlation coefficient will be zero.

Another property of the Pearson correlation is that it is both translation and scale invariant. This makes it particularly useful for assessing models that are beginning to capture the right "form" of the targets, even if they have not yet learned the correct scale or position. For example, a model might predict the general shape or direction of the gaze movement but miss the exact magnitude, which would be penalized heavily by the MED but less so by the correlation. This property makes correlation especially valuable in early stages of model development, where capturing the form is a positive sign of learning.

To further illustrate this, consider a plot of predicted values against the true values, with the true values plotted along the x-axis. For a perfect model, all points would lie on the 45° line, indicating that the predicted and true values are identical. The MED essentially measures the average vertical distance that each point lies away from this ideal 45° line. However, two different models can achieve the same MED while having fundamentally different prediction patterns. Take for example the predictions in Figure 4.10. The model on the left whose predictions randomly oscillate around the 45° line with a fixed distance (random noise) will have the same MED as the model on the right where all predictions lie on a straight line parallel to the 45° line but offset (systematic bias). The latter clearly captures important patterns of the data, while the former does not, yet their MED values would be identical.

It is important to note, however, that while correlation can provide valuable insights during model training, it is not as useful for benchmarking more advanced models. As models become more capable, they should not only capture the form but also accurately predict the scale and magnitude of the values. In these cases, MED becomes more relevant as a final evaluation metric, since it directly measures how close the predictions are to the true

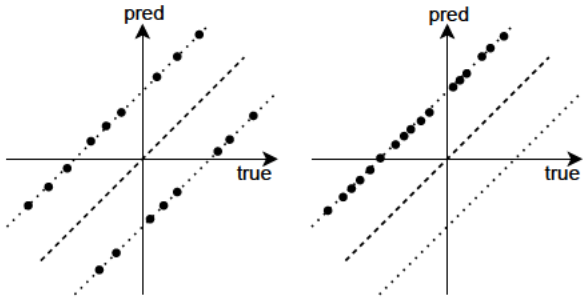

Figure 4.10: An illustration of two models with the same MED but different prediction patterns. The correlation coefficient would be 1 for the model on the right and close to zero for the model on the left, successfully distinguishing between the two.

values. Furthermore, the Pearson correlation coefficient is not well suited for the saccades task, where the target is discrete.

5 EVALUATION AND RESULTS

In this chapter, we present and analyze the results of our experiments. We begin by discussing the baseline and reference results on the Consumer EEG-ET dataset introduced in this thesis. Next, we evaluate the performance of the SpatialFilterCNN and BMOTE models on the same dataset. Finally, we conclude by showcasing the results of the FNNs described in Section 4.3, comparing their performance both on our dataset and on the EEGEyeNet dataset.

5.1 BASELINE AND REFERENCE RESULTS

Before we conducted various experiments with the proposed models, we first established different baselines and reference scores to compare the results against. Those include the following:

- *Random Baseline*: At every timestep, predict a random position.
- *Mean Baseline*: At every timestep, predict a constant: the mean of all training targets (stimulus positions).
- *Webcam*: At every timestep, predict what the webcam measured at that timestep (gaze position).

The purpose of the random baseline is to provide a lower bound to see if the model is learning anything at all. The purpose of the mean baseline is to provide a harder, more informative bound to see if the model is learning anything useful. It is rather easy for the models to collapse to the mean, so a model that beats the mean is probably learning something useful. The score of the webcam servers multiple purposes.

First, it serves as a target for "good" performance, as the webcam, from manual inspection, for the most part performs well. Second, it allows a direct comparison between "consumer-grade camera-based eye tracking" (*i.e.* webcam-based eye tracking) and "consumer-grade EEG-based eye tracking". The webcam is also a good sanity check for the metrics, *i.e.* most models probably perform worse than the webcam, the metrics should properly reflect that. The results of these baselines and reference scores are shown in Table 5.1. For the level-1 task, especially the level-1 smooth task, the mean baseline turns out to be surprisingly good, even beating the webcam in the MED metric.

There are several reasons for this outcome. First, the target remains in the center for a long time, in which cases, the mean baseline makes minimal errors. Second, after the dot jumps back to the center, the participants' reaction time causes the webcam to produce large errors. Both effects are visible in

	level-1-saccades		level-1-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
Random	165.9	177.1	143.6	177.4	−0.005	0.002
Mean	83.58	0.650	51.98	0.364	0	0
Webcam	81.89	1.479	74.59	0.890	0.687	0.347

	level-2-saccades		level-2-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
Random	199.8	177.3	164.5	177.5	0.002	0.002
Mean	162.3	0.591	104.5	0.420	0	0
Webcam	87.19	1.474	77.00	0.999	0.897	0.753

Table 5.1: Results of the baseline and reference models.

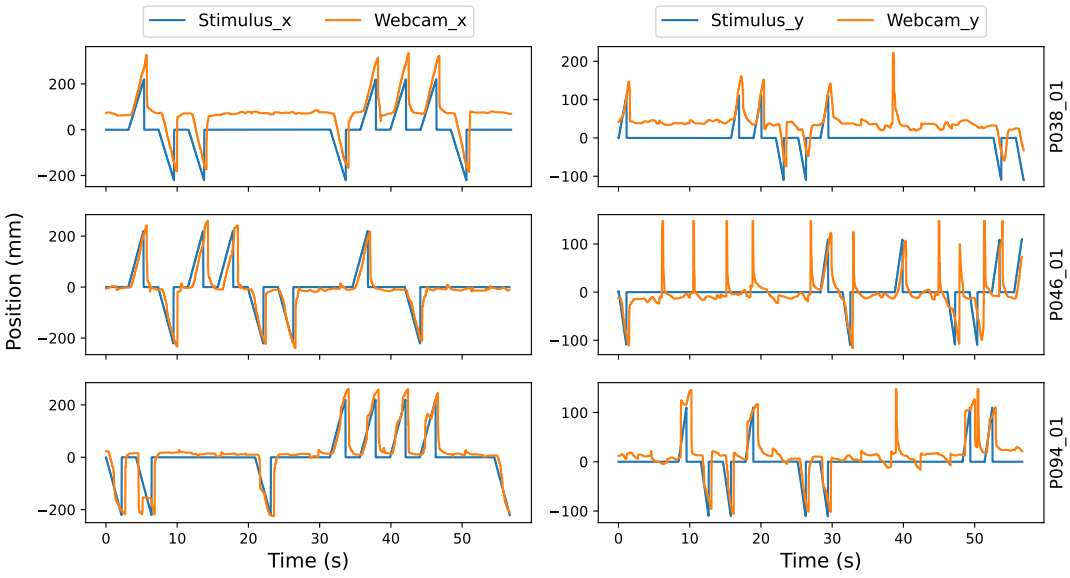

Figure 5.1: Webcam results for the level-1 smooth task. A selection of three participants from the test set is shown. Blinks are clearly visible as spikes in the vertical axis. Using those for blink detection would probably be more robust than our EEG-based blink detection approach. However, for this, webcam data together with labeled blinks would be needed, which was not recorded.

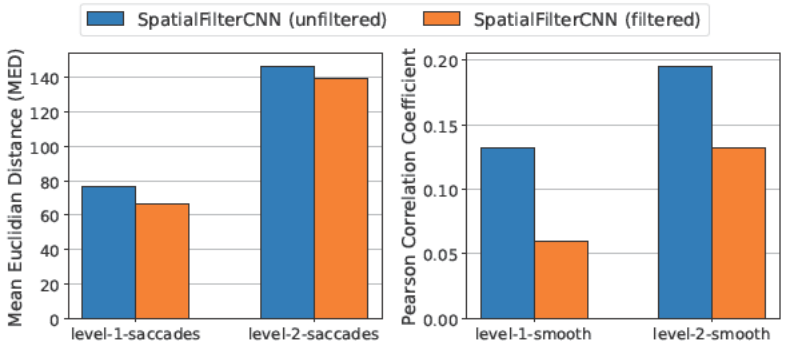

Figure 5.2: Performance of the SpatialFilterCNN model with unfiltered and filtered data. Higher correlation scores and lower MED scores are better.

Figure 5.1. This shows that the MED metric on the level-1 smooth task, fails to properly distinguish a model that is following the target (*i.e.* the webcam) from a model that just predicts the mean. This underscores the importance of using multiple metrics to get a more complete picture of the model's performance, which should be kept in mind when evaluating the results of the other models.

5.2 SPATIALFILTERCNN RESULTS

To assess the performance of the SpatialFilterCNN model on our dataset, we initially replicated the hyperparameters used in the original paper [15]. Following the paper, the model was configured with $N_S = 16$, $N_1 = 32$, $N_2 = 64$, `spatial_filtering` enabled, and `equally_sized` convolutional layers (see Section 2.2.2 for details). While the original paper used a window size of 500, we opted for 512 samples to align with our sampling rate of 256 Hz, resulting in 2-second windows consistent with previous testing experiments.

We trained the model using both unfiltered and filtered data across all tasks to determine the impact of filtering on each type of task. The resulting MED for the saccades tasks and correlation for the smooth tasks are presented in Figure 5.2. Based on these initial results, we made an informed decision to utilize filtered data exclusively for saccade tasks and unfiltered data for smooth pursuit tasks in subsequent experiments.[1]

Since the optimal hyperparameters for the EEGEyeNet dataset — which featured a higher number of EEG channels and a higher sampling rate — might not be ideal for our dataset, we proceeded with hyperparameter tuning for each task. This approach aimed to identify the most suitable configuration for our specific dataset and task requirements. The following hyperparameters were tuned, with their respective allowed ranges:

- Window size: 128 to 1024 samples
- Learning rate: 10^{-5} to 10^{-1}

1 Blinks were not filtered out for any of those experiments, as the blink detection was added in a later stage of experimentation.

- Spatial filtering: Enabled or Disabled
- Equally sized windows: Enabled or Disabled
- N_S (number of spatial filters): 4 to 64
- N_1 (number of filters in the first residual block): 8 to 128
- N_2 (number of filters in the second residual block): 16 to 256

For each task, we trained 20 models with different hyperparameter configurations, each for 30 epochs. Performance was evaluated on a validation set separate from the test set. This approach was chosen to prevent potential overfitting to the test set that could occur through repeated evaluations during the hyperparameter tuning process. The validation set was constructed to mimic the characteristics of the test set. For this purpose, we selected the recordings from participants 1, 20, 28, 42, 52, and 69, which are of similar high quality and display similar demographic characteristics as the participants chosen in the test set. After the hyperparameter tuning was complete, we selected the best model based on its performance on the validation set. This final model was retrained and then tested on the unused test set.

For the saccades tasks the MED on the validation set was used as the objective function for the optimization process. While the MED proved to be an effective tuning metric for saccade tasks, it was less suitable for smooth pursuit tasks. As we had already seen in the baseline results in Section 5.1, the mean baseline performed surprisingly well on the level-1 smooth task, even beating the webcam in the MED metric.

This observation led us to use the correlation metric instead of MED as the tuning metric for the smooth pursuit tasks, which resulted in models, that successfully learned and reproduced the underlying structure of the smooth pursuit data, as shown in the results in Figure 5.3. The figure offers a detailed comparison of predicted versus true stimulus positions of the SpatialFilterCNN model deemed best by the hyperparameter tuning procedure for the level-1 and level-2 smooth tasks. For each task, three recordings from the twelve available in the test set are shown. The recordings were selected to provide a balanced representation of the model's overall performance, highlighting both strong and weak predictions to give a comprehensive view of its capabilities.

A clear trend across both tasks is that predictions for the `Stimulus_x` (left panels) tend to be more accurate than those for `Stimulus_y` (right panels). Furthermore, there are noticeable differences between recordings. Some recordings yield better predictions than others, likely reflecting differences in the quality of the recordings themselves.

Focusing on the level-1 smooth task, the model demonstrates reasonably accurate amplitude and timing in capturing the onset of smooth movement. However, it frequently fails to detect or fully reproduce certain movements, resulting in missed or incomplete predictions. For the level-2 smooth task, the model captures the timing of movements well, particularly the changes in direction, but struggles more with predicting the correct amplitude. The

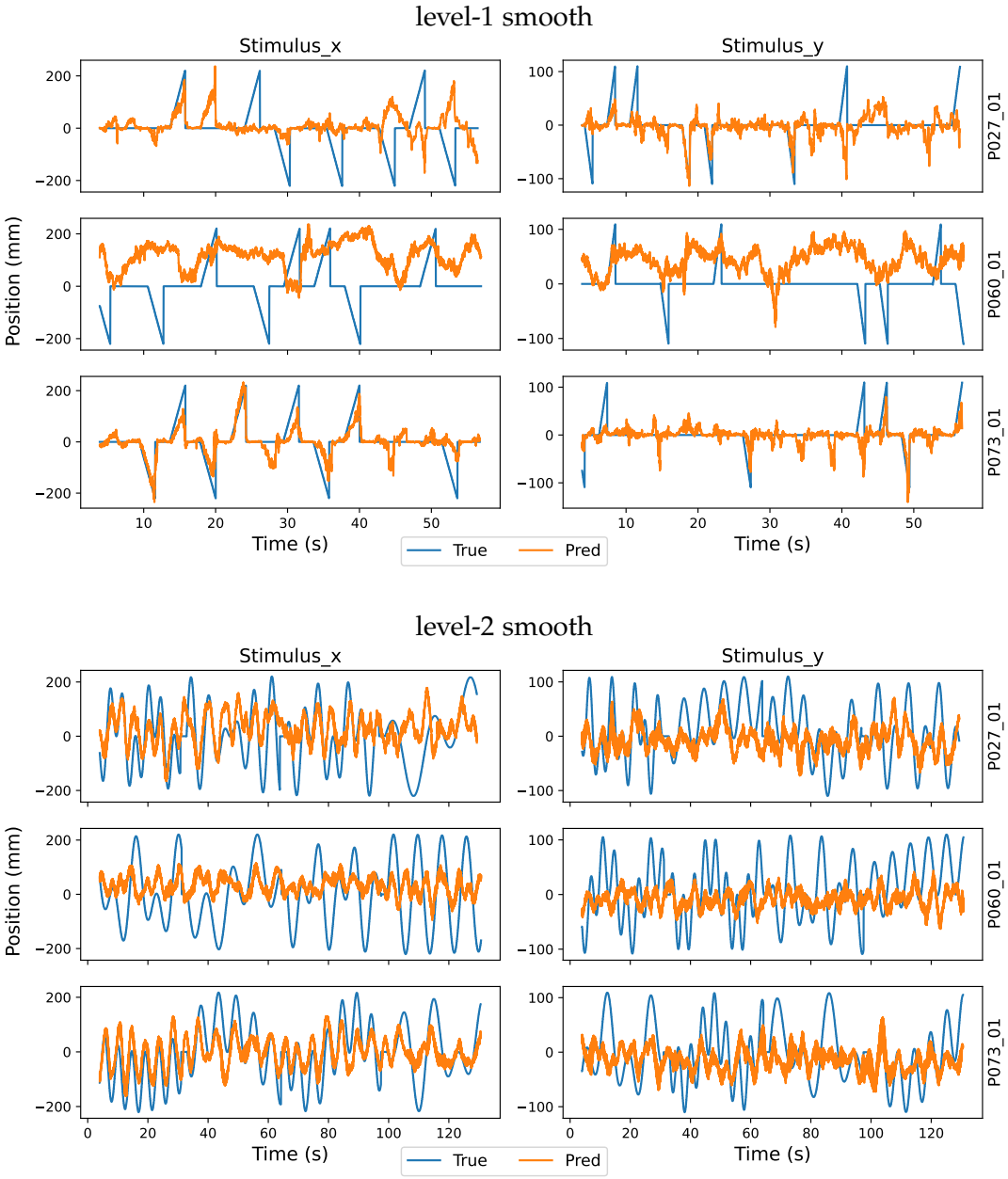

Figure 5.3: Predictions of the best SpatialFilterCNN model. Note, even though all smooth recordings started with the target at the center, this is not apparent from the plot. This is because the first value of each plot is the last value of the first window, which might not start at the center.

	level-1-saccades		level-1-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
SFCNN (unfiltered)	76.60	14.07	62.44	12.72	0.226	0.038
SFCNN (filtered)	66.54	7.087	59.32	12.79	0.140	−0.021
SFCNN (tuned)	54.62	4.648	57.92	6.093	0.199	0.192

	level-2-saccades		level-2-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
SFCNN (unfiltered)	146.7	53.69	119.3	48.92	0.322	0.069
SFCNN (filtered)	139.5	33.65	128.0	46.37	0.226	0.038
SFCNN (tuned)	109.0	6.454	99.38	20.84	0.434	0.157

Table 5.2: Results of the SpatialFilterCNN models. The unfiltered model was trained on unfiltered data, the filtered model on filtered data (as described in under *Filtering* of Section 4.1), and the tuned model was the best model found by the hyperparameter tuning process.

predicted movements often oscillate around the mean, meaning that the model has difficulty reproducing the lower frequency components of the target's motion.

Despite this, it is important to note that the frequency of the target movements varies significantly both within and across recordings. The model demonstrates an impressive ability to adapt to these variations, accurately predicting the moments when the target changes direction, even though it does not have access to past target positions. This highlights the model's ability to infer the correct timing for directional shifts based solely on the current window of EEG data. Furthermore, it should be emphasized that the participants included in the test set were not part of the training set. This means that not only were the recordings themselves novel to the model, but the participants were as well.

The results of the "best" SpatialFilterCNN model — as determined by the hyperparameter tuning process — are shown in Table 5.2, together with the results of the unfiltered and filtered models. The best (tuned) SpatialFilterCNN outperforms the mean baseline in terms of MED on all tasks except for the level-1 smooth task. However, as observed in Figure 5.3, the model has captured important aspects of the underlying structure of the level-1 smooth data, showing that it is doing more than simply predicting the mean. Compared to the webcam, all SpatialFilterCNN models perform worse in terms of MED on both level-2 tasks. On the level-1 tasks, surprisingly, all SpatialFilterCNN models outperform the webcam.

However, looking at the correlation metrics, the webcam is clearly superior. This is further supported when comparing Figures 5.3 and 5.1. To better assess the differences in performance on the level-1 saccades task between the best SpatialFilterCNN model and the webcam, we created a figure that

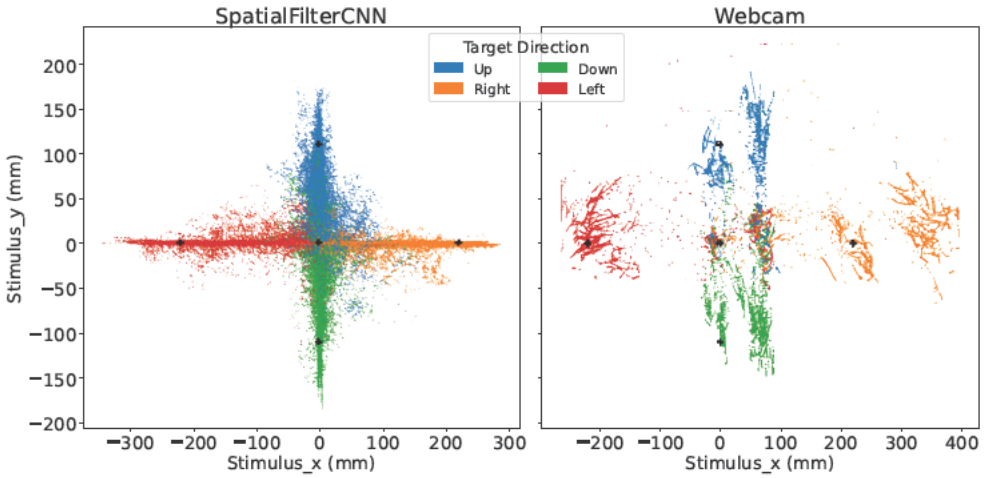

Figure 5.4: Comparison of the SpatialFilterCNN model and the webcam on the level-1 saccades task. Every prediction made on the test set is shown as a dot. The dots are colored based on the true target position. The four target positions are indicated by black crosses.

breaks down all predictions by separating them into four subplots based on the true target positions.

In this figure, shown in 5.4, we can see that the better performance of the SpatialFilterCNN model is due to the fact that the model has learned to confine its predicted positions to the x- or y-axis. This greatly reduces the prediction errors, but wont generalize well to targets outside of those lines. The webcam, on the other hand, often deviates from the coordinate axes and is partly biased to the right side. However, compared to the SpatialFilterCNN, the webcam's predictions are better separated from the centre.

5.3 BMOTE RESULTS

One of the first observations we made during our experiments with the BMOTE was the noticeable difference between the EEG data we collected using the Muse headset and the EOG data from [3]. A typical characteristic of both EEG and EOG data is the presence of baseline drift, which is the slow deviation from the baseline around which the signal initially oscillated. This drift can be caused by various factors, such as electrode movement, temperature changes, or changes in the impedance of the electrodes. However, as shown in Figure 5.5, the EEG data we collected using the Muse headset did not exhibit any baseline drift in contrast to the EOG data. Furthermore, the (baseline corrected) EOG data settles at different voltages for extended periods of time in line with the fixations performed by the participant, whereas the EEG data only seems to oscillate around the mean.

Given this discrepancy, we carefully examined the hardware and software involved to check if any onboard preprocessing, such as low-pass filtering or baseline drift correction, might be responsible for this difference. After reviewing the hardware specifications and the source code of the `muselsl`

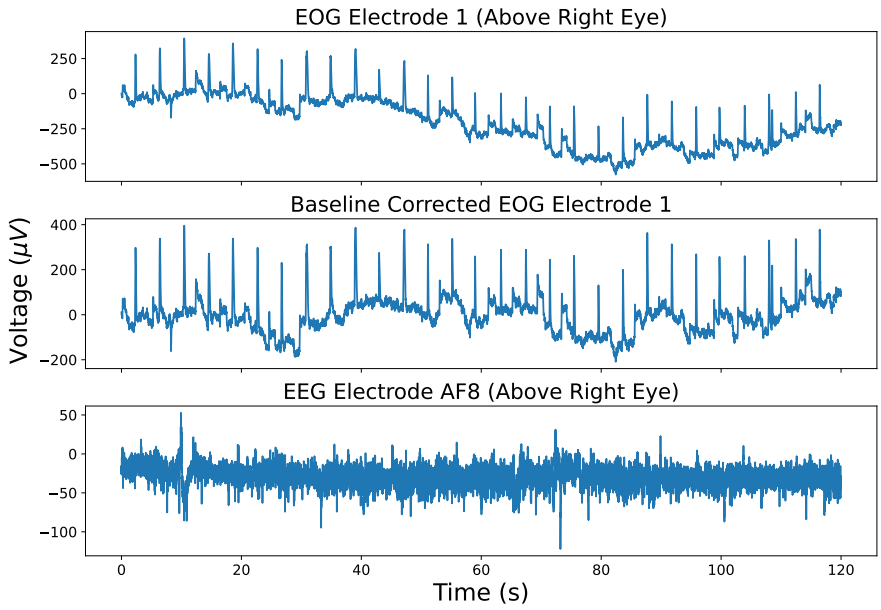

Figure 5.5: In the first panel, the EOG signal measured from subject 4 at the first electrode, the electrode above the right eye, is shown without any baseline correction. In the second panel, the same signal is shown with baseline correction. The third panel shows the EEG signal from recording "P045_01 level-2-saccades" at the electrode AF8, which is located on the forehead above the right eye on the Muse headband. The EEG signal is shown without any preprocessing.

package, which was used to interface with the Muse headset, we found no evidence of any preprocessing being applied by either the hardware or software.

This difference in the behavior of the EEG signal required extra thought when applying the BMOTE model to our data. In the original BMOTE paper, the authors employed several preprocessing steps, including:

- Bandpass filtering from 0 Hz to 30 Hz and applying a 50 Hz notch filter
- Blink removal based on template matching
- Baseline drift correction

Since our EEG signals did not show any baseline drift, we questioned whether applying an additional baseline correction step was necessary. Similarly, we were cautious about applying excessive filtering, as it might remove critical information from the signals.

To determine which preprocessing steps to retain or modify, we experimented with different levels of preprocessing. We started with only blink removal, using the blink detection model described in Section 4.1 with a threshold of 0.3, then tested blink removal combined with filtering, and finally, we applied the full set of preprocessing steps: blink removal, filtering, and baseline drift compensation. To evaluate the effectiveness of each approach, we examined the correlation between the voltage and gaze position, analyzing plots similar to Figure 4.8.

The results, however, were inconclusive. All preprocessing variations produced very similar outcomes. Given this lack of differentiation between the preprocessing methods, we decided to stick with the full preprocessing pipeline from the BMOTE paper to avoid introducing unnecessary deviations from the original approach.

To properly evaluate the BMOTE model, two additional steps were required:

1. We needed initial values for the electrode positions to begin the optimization process, as outlined in (2.5)
2. We needed to determine a suitable value for the constant $C = I/4\pi\sigma$, since the conductivity of the Muse headset electrodes was likely different from that of the electrodes used in [3]

To estimate the initial electrode positions P_{init}, we measured the placement of the Muse headset's electrodes relative to the center between the author's eyeballs while wearing the device. This was accomplished by taking both front-facing and side-facing images of the author's head zoomed-in from a distance to approximate an orthographic view. A reference object of known dimensions was placed next to the head in the images to provide a scale. From these images, we were able to estimate the electrode dimensions and positions, which were then averaged into a position per electrode. The final set for the TP9 (over left ear), AF7 (forehead left side), AF8 (forehead right side), and TP10 (over right ear) initial electrode positions in millimeters was (in this order)

$$P_{\text{init}} = \left\{ \begin{pmatrix} -83.19 \\ -87.33 \\ -0.02 \end{pmatrix}, \begin{pmatrix} 12.11 \\ -49.48 \\ 58.64 \end{pmatrix}, \begin{pmatrix} 12.11 \\ 49.48 \\ 58.64 \end{pmatrix}, \begin{pmatrix} -83.19 \\ 87.33 \\ -0.02 \end{pmatrix} \right\}.$$

In order to determine the value for the constant C, we conducted a search over a range of values, aiming to find the value for which the voltage levels of the model most closely matched the true voltage levels. For this, we calculated the power[2] of the voltage signal predicted by the model on the complete level-2 saccades training data and compared it to the power of the true signal measured by the electrodes.

The electrode positions of the model were initialized to P_{init} and estimated with formula (2.4) (instead of the original one (2.5)) also using the complete level-2 saccades training data. The average of the difference in power over all electrodes was calculated using the following formula

$$\frac{1}{N_e} \sum_{i=1}^{N_e} \left| \underbrace{\frac{1}{N} \sum_{k=1}^{N} V_i^{(\text{mod})}(\boldsymbol{\theta}_k)^2}_{\text{power of model voltage}} - \underbrace{\frac{1}{N} \sum_{k=1}^{N} V_i^{(\text{rec})}(\boldsymbol{\theta}_k)^2}_{\text{power of recorded voltage}} \right|, \tag{5.1}$$

2 The power of a signal is the sum of the squared values of the signal divided by the number of samples. It measures the amount of energy consumed per unit time.

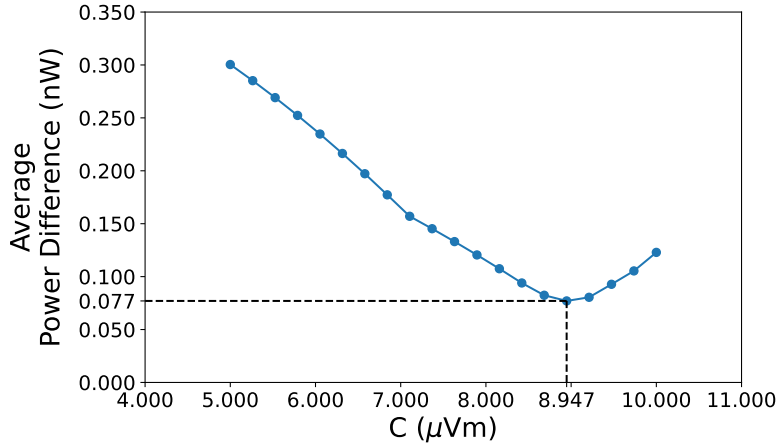

Figure 5.6: Tuning of the constant C for the BMOTE model. The optimal value for C is marked with dotted lines. Average power difference is given in "nano Watt".

where N_e is the number of electrodes, N is the number of samples, $V_i^{(\text{mod})}$ is the predicted voltage level of electrode i, $V_i^{(\text{rec})}$ is the recorded voltage level of electrode i, and $\boldsymbol{\theta}_k$ is the vector of vertical and horizontal gaze angle at time k. The difference in power was calculated for 20 values of C evenly spaced between $5\,\mu\text{V}\,\text{m}$ and $50\,\mu\text{V}\,\text{m}$. The results are shown in Figure 5.6, where the power difference is minimized for $C = 8.95$.

With the values for P_{init} and C determined, we were able to proceed with evaluating the BMOTE model on the test data. In the original BMOTE paper, the model is evaluated using electrode positions estimated from data of the same participant. However, in our case, we did not have additional (level-2) saccades data for the participants in the test set. This meant that the same level-2 saccades data used to estimate the electrode positions was also used to evaluate the model's performance on the level-2 saccades task. This approach potentially introduces an optimistic bias in the level-2 saccades evaluation, as the electrode positions were estimated using the same data they were tested on.

Ideally, we would have split the level-2 saccades data into two halves, one half for estimating the electrode positions, and the other half for evaluation. However, this was not feasible for two key reasons: first, we did not have enough level-2 saccades data per participant to make such a split, and second, splitting the data in this way would result in evaluating BMOTE on a different test set compared to the other models, making direct comparisons difficult. To mitigate this issue, we also evaluated the BMOTE model always using the electrode positions form the run with the optimal C value, *i.e.* electrode positions estimated from the full level-2 saccades training data.

Furthermore, we introduced a post-processing step where predictions outside a plausible range were discarded. This was necessary because the model often predicted extreme gaze angles close to or exceeding 90°, which, when converted to screen coordinates, resulted in extremely large values or even

	level-1-saccades		level-1-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
BMOTE*	164.86	158.30	140.31	150.14	0.022	−0.027
BMOTE (orig. formula)**	158.82	73.01	121.57	60.67	−0.089	−0.009
BMOTE (train estimated)***	159.34	184.84	145.35	181.28	−0.060	−0.049
Discarded: * 98.4%, ** 98.4%, *** 99.7%			Discarded: * 97.9%, ** 96.7%, *** 99.8%			

	level-2-saccades		level-2-smooth			
Model	MED	Precision	MED	Precision	corr_x	corr_y
BMOTE*	197.51	164.17	164.49	153.99	−0.007	0.017
BMOTE (orig. formula)**	194.91	76.99	143.73	74.25	0.059	0.078
BMOTE (train estimated)***	196.64	203.87	163.79	186.43	0.003	−0.002
Discarded: * 97.9%, ** 98.4%, *** 99.8%			Discarded: * 97.3%, ** 98.8%, *** 99.9%			

Table 5.3: Results of the BMOTE models. The results under BMOTE correspond to the models evaluated using the electrode positions estimated for every participant separately with formula (2.4). The results under BMOTE (orig. formula) correspond to the models evaluated using the electrode positions estimated for every participant separately with the original formula (2.5) from the paper. The results under BMOTE (train estimated) correspond to the models evaluated using the electrode positions estimated from the full level-2 saccades training data with formula (2.4). The percentage of discarded predictions is shown below each task.

infinity. To prevent these outliers from distorting the overall metrics, we removed all predictions that fell outside the known viewing boundary of −220 mm to 220 mm mm on the x-axis and −110 mm to 110 mm on the y-axis. We believe this adjustment is reasonable, as it could also be applied during inference in practical applications.

The results of both experiments are shown in Table 5.3, including the number of discarded values. The experiment where electrode positions were estimated for every participant separately, was done once with the original formula for electrode positions estimation (2.5) and once with the alternative formula (2.4). We see that all configurations performed close to random, with nearly all of the predictions (always over 97%) being discarded during post-processing. This is likely in large part due to the assumption of point electrodes not being met by the Muse headband. Additionally, the EEG data we collected differed noticeably from the EOG data used in the original BMOTE paper [3]. While we anticipated this to impact the model's performance, we did not expect a complete failure in gaze prediction. Such a poor performance probably suggests that, at distances further from the eye, the voltage differences are no longer primarily governed by a simple battery model but are heavily influenced by other factors.

5.4 FNN RESULTS

After introducing the FNN architectures in Section 4.3, we now evaluate the performance of these models on our dataset. All models were trained for 30 epochs using the Adam optimizer, with a learning rate of 0.0008, a batch size of 384, and MSE as the loss function. We excluded blinks from the training data by filtering out windows, which included samples with a blink probability of at least 0.3. No additional filtering was applied to the data, even though filtered data had previously shown improved results for saccades tasks with the SpatialFilterCNN. This omission is acknowledged as a missed opportunity.[3]

To isolate the impact of functional layers on model performance, control models were trained for each architecture. These control models retained the overall structure of the FNNs but replaced the functional layers with equivalent "standard" layers. This allowed for a direct comparison between functional and standard architectures.

For the fully functional architecture, the functional residual blocks were replaced by standard residual blocks, while the two final functional dense layers were substituted with two convolutional layers followed by a global average pooling layer. The first convolutional layer contained 256 filters with an ELU activation function, and the second contained 2 filters with a linear (no) activation. Both layers had a kernel size of 12. This "substitution" closely mirrors the structure of the FNN, but allows the filters to adopt arbitrary (potentially non-smooth) forms, unlike the functional layers with inherent smoothness constraints.

In the architecture featuring a functional body, the control model is simply created by replacing the functional residual blocks with standard residual blocks. Finally, for the minimally functional architecture, where only the head contained a functional layer, we replaced the single functional dense pooling layer with a standard convolutional layer, a global average pooling layer, and an ELU activation. The convolutional layer had 512 filters, to match the 512 neurons of the functional dense layer, and a kernel size of 12. Global average pooling was used to obtain a "scalar" output as in the functional dense pooling layer. Once again, this configuration closely matched the original FNN, though the convolutional layer allowed for arbitrary filters in contrast to the smooth filters produced by the functional dense layer.

With these closely matched control models, we can attribute any systematic differences in performance to the functional layers themselves. The control models were trained under the exact same conditions as the functional models, namely for 30 epochs with the Adam optimizer, a learning rate of 0.0008, a batch size of 384, and MSE as the loss function.

The results of the FNN models, along with their control counterparts, are presented in Table 5.4. Across nearly every task and metric, the FNNs outperform the control models. One exception is observed in the level-1 smooth

3 Filtering was planned for the saccades tasks, but was forgotten during implementation. Unfortunately this was noticed too late to be corrected without the need for significant retraining.

	level-1-saccades		level-1-smooth			
Model	MED	Precision	MED	Precision	$corr_x$	$corr_y$
FullyFunc	**73.02**	1.908	61.18	0.898	0.220	**0.052**
FullyFunc (control)	73.41	**1.772**	**55.11**	**0.882**	**0.285**	0.011
FuncBody	**78.35**	**2.278**	**57.47**	**0.965**	**0.259**	0.027
FuncBody (control)	78.85	2.366	64.69	1.100	0.220	**0.046**
MinFunc	**64.72**	**1.865**	**56.18**	**0.901**	**0.168**	0.070
MinFunc (control)	71.13	1.920	58.30	1.007	0.154	**0.076**

	level-2-saccades		level-2-smooth			
Model	MED	Precision	MED	Precision	$corr_x$	$corr_y$
FullyFunc	**127.5**	**2.276**	**100.8**	**0.913**	**0.409**	**0.138**
FullyFunc (control)	128.8	2.391	104.4	1.092	0.364	0.104
FuncBody	**130.6**	2.842	**104.2**	**1.161**	**0.384**	0.097
FuncBody (control)	135.8	**2.747**	105.2	1.176	0.378	**0.146**
MinFunc	**129.7**	2.602	**101.5**	**1.181**	**0.411**	**0.159**
MinFunc (control)	132.2	**2.494**	108.2	1.336	0.348	0.087

Table 5.4: Results of the FNN models. The better result between the functional and control model is highlighted in bold, and the best result for each metric is underlined. The control models are named by appending "(control)" to the model name, which admittedly is not ideal, as for example the "FullyFunc (control)" model has no functional layers (except for one in the stem). However, this naming convention makes it easier to see the functional-control pairs.

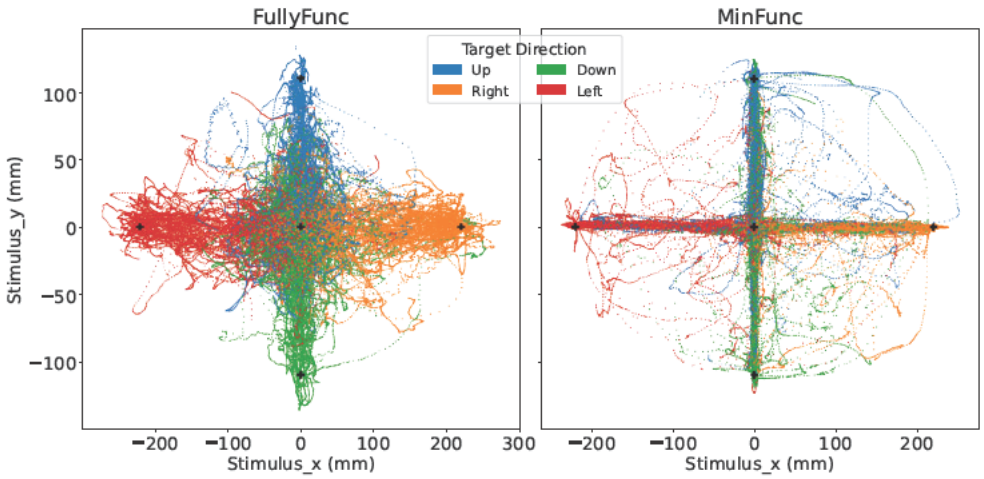

Figure 5.7: Comparison of the fully functional model and the minimally functional model on the level-1 saccades task. Every prediction made on the test set is shown as a dot. The dots are colored based on the true target position. The four target positions are indicated by black crosses.

tasks, where the fully functional model underperforms compared to its control in all metrics except corr_y, where both models have a value close to zero. Generally, the best performance in the corr_y metric is less "one-sided". Notably, when the control model shows better results in corr_y, both models exhibit values near zero.

Of the three architectures, the fully functional model performs the best on all tasks across most metrics, followed by the minimally functional model in most cases. One stark exception is the level-1 saccades task, where the minimally functional model significantly outperforms the fully functional model in the MED metric. However, upon closer inspection, we find that this difference mostly stems from the fact that the minimally functional model has learned to make predictions where one axis is set to zero. This is evident in the results shown in Figure 5.7. Upon closer inspection, one notices that the minimally functional model frequently confuses left with up and right with down, as evidenced by the large cluster of blue dots to the left and green dots to the right. Thus, judging only by the ability to separate view directions, the minimally functional model likely performs worse than the fully functional model.

When comparing the FNN results to those of the SpatialFilterCNN, we observe several noteworthy trends. First, Precision is significantly improved in the FNN models compared to the SpatialFilterCNN models. This means that the predictions of the FNN models are more consistent and "jump around" less than those of the SpatialFilterCNN models. This improvement, however, is also observed in the control models, suggesting that the general architecture is likely responsible for this enhancement rather than the functional layers themselves. Additionally, on the smooth tasks, the FNN models perform better or closely match the MED of the best SpatialFilterCNN model,

while on the saccades tasks, the best SpatialFilterCNN model outperforms all FNN models in the MED metric.

Importantly, the fully functional model outperforms any untuned (and unfiltered) SpatialFilterCNN across all tasks and metrics.[4] This opens up the possibility that with further tuning, the FNNs could potentially deliver even better overall performance than the SpatialFilterCNN, not just in smooth tasks but also in saccades tasks. Furthermore, the FNNs were not trained with any additional filtering, which was shown to improve the performance of the SpatialFilterCNN models on the saccades tasks.

5.5 FNN RESULTS ON EEGEYENET DATASET

To gain a more comprehensive understanding of the performance of functional models on EEG-based eye tracking data, we extended our evaluation to include the EEGEyeNet dataset. Unlike our dataset, which reflects what is currently possible with consumer-grade hardware, the EEGEyeNet dataset represents the high-end of data quality for EEG-based eye tracking, featuring laboratory-controlled conditions and research-grade EEG equipment with more electrodes and wet sensors. This comparison allows us to explore how FNNs perform under optimal data collection conditions, versus more practical, real-world scenarios.

The exact same architectures and parameters that were used on our dataset were applied to the EEGEyeNet dataset The training parameters were set to match the ones used by the SpatialFilterCNN. This included early stopping based on the validation loss, with patience of 20 epochs, the Adam optimizer with a learning rate of 0.0001 and a batch size of 64. Models were trained for a maximum of 50 epochs, with a window size of 500 (which is fixed by the EEGEyeNet dataset).

Following the standard practice from the EEGEyeNet paper [23], each model was trained and evaluated five times. The results, including the mean and standard deviation of the MED and MAE metrics, are summarized in Table 5.5. With the exception of the fully functional control model, all functional architectures outperformed the SpatialFilterCNN in both the MED and MAE metrics, resulting in new state-of-the-art results on the EEGEyeNet benchmark (at least to the best of our knowledge). The best-performing model was the MinFunc control model, which surpassed the SpatialFilterCNN by 2.6 mm in the MED and 1.8 mm in the MAE.

It is important to note that the architectures used in these experiments were not tuned specifically for the EEGEyeNet dataset, and it is likely that further improvements could be achieved with hyperparameter tuning. However, the same can be said for the SpatialFilterCNN, as its parameters were adopted directly from the work of [15], which did not indicate any kind of tuning.

Unlike the results on our dataset, where functional layers showed a clear benefit, the differences between the functional, and control models on the

4 With one exception being the corr_x metric on the level-1 smooth task

Model	MED	MAE
FullyFunc	**68.5** ± 1.0	**42.7** ± 0.6
FullyFunc (control)	69.9 ± 1.3	43.6 ± 0.8
FuncBody	**68.0** ± 0.8	42.4 ± 0.6
FuncBody (control)	68.1 ± 0.5	**42.3** ± 0.3
MinFunc	66.8 ± 0.5	41.5 ± 0.4
MinFunc (control)	**66.2** ± 0.8	**41.1** ± 0.5
SpatialFilterCNN	68.8 ± 1.4	42.9 ± 0.8

Table 5.5: Results of the FNN models on the EEGEyeNet dataset. The mean and standard deviation of the MED and MAE metrics are shown for each model. The better result between the functional and control model is highlighted in bold, and the best result for each metric is underlined. The results of the SpatialFilterCNN are included for comparison. The MED and MAE are reported in millimeters using a conversion factor of 0.5.

EEGEyeNet dataset were much smaller. In fact, the control models outperformed the functional models half of the time. Therefore, the advantage of functional layers in the constructed architectures is less clear when evaluated on the EEGEyeNet dataset.

Several factors could explain these results. One possibility is that functional models scale better with larger window sizes or handle noisy data better than the control counterparts. This could account for the more consistent improvements observed with the FNNs compared to the control models in our dataset, where the window size was larger, and the signal from the dry electrodes less clean. Alternatively, the differences observed between the functional and control models may simply be due to chance, such as variations caused by random weight initialization. To confidently assess the influence of functional layers, many more repeated experiments would be required. That being said, the results already suggest that the benefit of functional layers on the EEGEyeNet dataset, at least for the constructed architectures, is minimal at best.

Finally, we note that having surpassed the current state of the art with a model architecture that is mostly based on standard convolutional architectures, suggests that there is still considerable room for improvement in the field of EEG-based eye tracking.

6

DISCUSSION, OUTLOOK AND CONCLUSION

In this final chapter, we will revisit the key contributions of this thesis, summarizing the major findings and their implications. We begin by discussing the "Consumer EEG-ET dataset" a large-scale EEG and eye tracking dataset that we collected. Next, we reflect on the insights gained from experimenting with FNNs, and their performance on EEG-ET tasks. Finally, we consider the broader limitations of our work and outline potential directions for future research, ending with a final conclusion.

DISCUSSION AND IMPLICATIONS

One of the key contributions of this thesis is the introduction of a new, large-scale EEG eye tracking dataset collected using consumer-grade hardware. This dataset stands out as the largest known dataset of its kind, featuring over 11 hours and 45 minutes of continuous training data from 113 participants. It incorporates two different experimental paradigms: smooth pursuit and saccades tasks in two increasing levels of difficulty. Notably, this dataset addresses critical gaps present in other EEG-ET datasets, which either lack smooth movements — important for applications like the FDA — or rely on research-grade hardware, making them less applicable to real-world scenarios.

Because such a dataset was never collected before, it was unclear whether EEG data recorded with consumer-grade hardware — with as little as four electrodes — would be sufficient to predict gaze position in any meaningful way. Additionally, the relaxed recording conditions compared to related works, such as the absence of a chin rest, no special isolated room for recording, *etc.* introduced the potential for high noise levels that could obscure any usable signal. To address these concerns, we evaluated a range of models and compared their performance against simple baselines, such as mean and random predictions.

With the exception of the level-1 smooth task, where even the webcam failed to outperform the mean baseline, all baseline models could be successfully beaten by an evaluated model. Based on this and detailed investigations using various metrics and visualizations, we can confidently conclude that the Consumer EEG-ET dataset contains sufficient information to predict gaze positions across all tasks. Thus demonstrating that simple EEG-based eye tracking works even with consumer-grade hardware (with as few as four electrodes!), and in relaxed recording conditions, making a first step towards EEG-based eye tracking in real-world applications.

A Webcam is effectively the consumer-grade hardware for camera-based eye tracking. Having recorded the gaze position with it, we can thus make a

comparison between EEG-based and camera-based eye tracking using consumer-grade hardware. The webcam achieved accuracies ranging from 6.94° to 7.69° across the four tasks, which is significantly lower than the performance of high-end camera-based eye tracking systems that typically reach accuracies between 0.25° and 0.50° [35]. Despite this, the webcam generally outperformed the models trained on EEG data. Specifically, on the level-2 smooth and level-2 saccades tasks, the best model (SpatialFilterCNN) achieved accuracies of 9.29° and 9.96°, respectively. However, on both level-1 tasks, the FNNs and SpatialFilterCNN surpassed the webcam in terms of the MED metric.

This comparison suggests that while consumer EEG hardware is not yet competitive with consumer camera-based systems for general eye tracking tasks, it shows promise in more controlled scenarios where eye movement is restricted to one axis. In such cases, the EEG-based models performed favorably compared to the webcam. Nevertheless, it is important to acknowledge that a webcam-based model specifically trained for this restricted scenario would likely show improved results.

The second major contribution of this thesis is the exploration of FNNs for EEG-based eye tracking tasks. FNNs produced compelling results on both our dataset and the EEGEyeNet dataset. When compared to the SpatialFilterCNN, the FNNs demonstrated competitive or superior performance on smooth tasks, and benefited from more precise predictions on all tasks, with the fully functional model generally performing the best out of the three proposed architectures. Importantly, this was achieved without any tuning or data filtering, suggesting potential for even greater performance with further optimizations. Additionally, we set a new state-of-the-art benchmark on the EEGEyeNet dataset.

However, these successes were not exclusive to the FNNs. Control models, in fact, achieved similar results, with a control model securing the best performance on the EEGEyeNet dataset. On our dataset, functional models generally outperformed their control counterparts on most tasks and across various metrics. On the EEGEyeNet dataset, however, the performance gap between the functional and control models was less distinct. This suggests that the functional layers may have had only a minor impact, with the overall architecture playing a more significant role in driving the results. While we observed hints of improved performance from the functional layers, further experimentation is needed to draw any definitive conclusions.

The fact that state-of-the-art performance was achieved with control models — based on relatively simple reference architectures — was surprising. This highlights the untapped potential in the field of EEG-based eye tracking, where more advanced architectures, such as those based on transformers, could yield even better results. Regarding functional layers, given their novelty, it remains unclear whether traditional CNN architectures are the most suitable for them. It is possible that different, more "functional-aware" architectures could allow these layers to demonstrate their full potential. This, however, remains an open question requiring further research.

The BMOTE performed quite poorly on our dataset, failing to make any meaningful predictions and performing close to random. In fact, when not discarding predictions outside the viewing boundary, the BMOTE model actually performed significantly worse than the random baseline. One key reason for this poor performance is likely the assumption of point electrodes, which is not well-suited to the Muse headband. However, comparing our EEG recordings to the EOG recordings from the original BMOTE paper, this is probably not the only limitation at play. Given that BMOTE is based on a physical model of the eye, its failure to generalize from EOG to EEG suggests that the voltage differences measured further from the eye are no longer governed by a simple battery model.Instead, they are heavily influenced by additional processes.

LIMITATIONS

Despite the promising results and contributions presented in this thesis, several limitations need to be acknowledged. These limitations highlight areas where the methodology or scope of the study could be refined as well as open questions that remain unresolved. Addressing these factors in future research could further strengthen the findings and expand upon the work conducted here.

One limitation of the dataset collection process is the absence of a high-end camera-based eye-tracking system for providing a reliable ground truth. Although we acquired a dedicated eye tracker, licensing issues prevented us from using it in our study. As a result, we relied on a standard webcam, which due to its limited accuracy is not suitable as a precise ground truth system but rather serves as a reference model. Consequently, the stimulus position itself was used as ground truth. While this approach provides a reasonable approximation, it does not account for cases where participants failed to follow the target accurately, or lagged behind the target due to reaction time delays. Despite these challenges, we aimed to mitigate such issues by designing the stimulus presentation to guide participants, ensuring they could easily anticipate the movement and timing of jumps of the target. Furthermore, the webcam, while lacking spatial precision, offers high temporal accuracy, which could potentially allow for reaction time correction, and the detection of major deviations from the expected gaze patterns.

Another limitation lies in the design of the level-1 smooth task. While it was intended to reflect smooth eye movements, the task was not entirely continuous, as the dot jumped back to the center after smoothly reaching the edge of the screen. A more ideal movement pattern would have involved a gradual slowdown at the screen's edge, followed by a smooth return to the center. As a result, performance on the level-1 smooth task may be less representative of continuous smooth eye movements, making it unsuitable as a proxy for model performance on more complex smooth tasks like the level-2 smooth task. Essentially, the dataset lacks a true entry-level smooth task. This also reduces its usefulness for FDA. However, the level-2 smooth

task, which offers more varied movements and twice the amount of data, remains available for such analyses.

An unexpected observation during the dataset collection was the absence of baseline drift in the EEG recordings obtained from the Muse headband. This is somewhat unusual, as baseline drift is typically expected in EEG signals. While this requires further investigation, it does not pose a significant problem, as a first step in most EEG processing is usually to remove the baseline drift by filtering out low-frequency components anyways. Additionally, we were able to demonstrate that, despite differences compared to traditional EEG recordings, meaningful eye-tracking predictions could still be made based on this dataset.

Due to limited time and compute resources, we were unable to perform hyperparameter tuning for the FNNs, leaving room for potential performance improvements. Moreover, the models were trained only on unfiltered data. Using filtered data might have further improved the results on the saccades task as we observed for the SpatialFilterCNN models. This presents a clear opportunity for future work to explore.

Additionally, while our findings suggest that FNNs generally outperformed control models on our dataset, these conclusions would benefit from more experimental runs, ideally with cross-validation. Increasing the number of repetitions would provide a more robust assessment of the impact of the functional layers and allow for more definitive conclusions.

Finally, we were unable to fully reproduce the results reported in the EEGEyeNet paper, as the original work did not clearly specify the metric or pixel-to-millimeter conversion used. Additionally, the provided code yielded results that differed significantly from those in the paper. While this creates some uncertainty around our quantitative findings on the EEGEyeNet dataset, our qualitative conclusion remains: based on the plots of true versus predicted values, FNNs appear to be either better than or at least competitive with the SpatialFilterCNN.

OUTLOOK AND FUTURE WORK

There are several directions for future work to address the limitations of this study. Hyperparameter tuning and training on filtered data, which were omitted due to time and resource constraints, could be straightforward to implement and may lead to immediate performance gains. Optimizing the data loading pipeline is another area of improvement. Faster loading would reduce training times and allow for more extensive experimental runs, helping in the assessment of the impact of functional layers. To further explore the potential of FNNs, future research could attempt to design more "functional-aware" architectures. For example, one missed opportunity in this thesis was training FNNs with functional outputs, which could potentially better leverage the unique capabilities of these architectures. Lastly, the discrepancies with the EEGEyeNet paper could likely be resolved by consulting the origi-

nal authors for clarification on their metrics and pixel-to-millimeter conversion.

To further enhance EEG-based eye tracking performance, several promising strategies could be explored. One key area is improving and expanding on ways the data is preprocessed. For instance, using webcam data we could correct for participants' reaction times, by temporally aligning the stimulus position with the gaze position recorded by the webcam. Dynamic Time Warping, which we began experimenting with, could for example be a viable approach for this task. Additionally, filtering out sections where participants failed to follow the target could further improve the quality of the training data, particularly when combined with temporally corrected targets.

Another avenue is experimenting with lagged target data, where instead of using the latest stimulus of each window as the training target, an earlier stimulus is used as the target. This approach would provide the model with "post-eye-movement" EEG data, which might contain more useful information about the eye's position at the lagged time as relevant voltage fluctuations might not immediately settle after the eye movement. However, while this method might improve model performance, it would also introduce some latency into the model's predictions, which in real-time applications can only be tolerated to a certain extent. Note, that this too would benefit from time corrected targets, as otherwise the lagged targets, in reality, would be lagged by different amounts.

Additionally, a simple way to increase the data available for training could be to train with data from all tasks and then fine-tune on a specific task.

Another potential improvement is training models to predict relative movement instead of absolute gaze position. Given that the voltages in our recordings tend to revert to the mean over time (which we observed as a missing baseline drift), information about the absolute position of the eye can gradually be lost in the signal. Predicting relative movement could help preserve the model's accuracy over longer time windows. Modern architectures capable of incorporating much larger windows of past data, such as transformers, could also address this challenge.

Another promising approach involves utilizing endogenous data, treating the problem more like a time series forecasting task. The methods presented in this thesis rely solely on exogenous data (EEG) to predict eye position, but incorporating past predictions of the stimulus position could provide useful context. Although implementing this would require the model to rely on its own predictions at test time — introducing some challenges — it could help the model follow slow eye movements and improve performance over extended periods.

Finally, exploring regularization techniques like early stopping or dropout could help avoid overfitting. While we briefly tested these strategies in a few initial experiments without seeing substantial gains, more thorough implementation could lead to improvements in future studies. This might be particularly important, as all trained models showed a tendency to overfit, performing significantly better on the training set than on the validation set.

CONCLUSION

In this thesis, we set out to build a large-scale EEG-ET dataset using consumer-grade hardware and assess the feasibility of EEG-based eye tracking under those conditions. The result is the largest known dataset of its kind. Through extensive analysis, we demonstrated that simple EEG-based eye tracking is possible even with consumer-grade hardware in relaxed recording conditions, a promising step for future applications.

Additionally, we explored the performance of models based on functional data analysis, specifically FNNs, for EEG-based eye tracking tasks. While the FNN architectures showed some promise, achieving state-of-the-art performance on the EEGEyeNet dataset, the impact of functional layers remains inconclusive and requires more extensive experimentation.

Overall, we showed that EEG-based eye tracking with consumer-grade hardware is feasible and provided a new dataset to continue work in this area. It is now up to future research to explore the limits of this technology and advance it further. There are many avenues for further research, as outlined above, and we strongly believe that the current state of EEG-based eye tracking, largely reliant on standard techniques, is only scratching the surface of what is possible. More focused efforts could push the technology significantly forward, revealing its full potential.

Part II

APPENDIX

A

SUPPORTING DOCUMENTS

A.1 CONSENT FORM

The following pages contain the consent form that was signed by each participant before the study. It is available in both English and German.

General Participant Information and Consent

Title of the Study: Creation of a Consumer EEG-ET Dataset

1. Research Project

Welcome to our study “Creation of a Consumer EEG-ET Dataset”! We appreciate your interest in this study.

The goal of this study, conducted as part of a master's thesis, is to create a dataset that includes EEG and eye-tracking data (EEG-ET). This dataset will be used to train models that can reconstruct eye movements based on EEG data (measured with consumer hardware).

The EEG data will be collected using the Muse S Gen 2 headset, a widely used consumer EEG headset with 4 dry electrodes. To capture eye movements, the positions of the targets on the screen will be recorded, and a webcam will also be used to track eye movements. No images or video will be saved, only the eye position derived from the live video stream.

Study Procedure

Participation in this study will take approximately 10 minutes in total and consists of the following parts: introduction, practice, and recording. Data will be collected for a total of 6 minutes.

1. **Introduction:** During the introduction, relevant demographic data will be collected, the camera-based eye tracker will be calibrated, you will be fitted with the EEG headset, which will be adjusted to your head, and the signal quality will be checked. You will also receive important instructions to follow during the recordings.
2. **Practice:** Before each recording, a short practice session will be conducted to prepare you for the respective task.
3. **Recording:** During the actual data collection, EEG and eye-tracking data will be recorded in four different situations:
    - Level 1 Smooth:
        - You will follow a point with your eyes that repeatedly moves steadily from the center to the left, right, up or down. Duration: approx. 1 minute.
    - Level 1 Saccades:
        - You will follow a point with your eyes that repeatedly jumps from the center to the left, right, up or down. Duration: approx. 1 minute.
    - Level 2 Smooth:
        - You will follow a point that steadily traverses various paths on the screen. Duration: approx. 2 minutes.
    - Level 2 Saccades: You will follow a point that jumps around on a grid. Duration: approx. 2 minutes.

If you have any questions, please contact the experimenter.

2. Voluntary Participation and Anonymity

Participation in the study is voluntary. You can withdraw your consent to participate in this study at any time and without providing any reasons, without any disadvantages to you. Data collection is completely anonymized, meaning your name will not be requested at any point. Your responses and results will be stored under a participant number that cannot be traced back to you.

3. Compenstation

No financial compensation is offered for participation in this study. Please be aware that your participation is voluntary and unpaid. We greatly appreciate your commitment and support for our research and sincerely thank you for your participation.

4. Contact Information

Tiago, Vasconcelos Afonso
Tel.: +49 173 4161278
E-Mail: tiago.vasconcelosafonso@stud.h-da.de

5. Consent to Participate in the Study

I hereby voluntarily consent to participate in the study “Creation of a Consumer EEG-ET Dataset.” I have been adequately informed and had the opportunity to ask questions.

I have received a copy of the information and consent form.

_____

Place, Date

_____

Name of the Participant in Block Letters

_____

Signature of the Participant

Allgemeine Teilnehmerinformation und Einwilligung

Titel der Studie: Erstellung eines Consumer EEG-ET Datensatzes

1. Forschungsvorhaben

Herzlich willkommen bei unserer Studie „Erstellung eines Consumer EEG-ET Datensatzes“! Wir danken Ihnen für Ihr Interesse an dieser Studie.

Ziel dieser Studie ist es im Rahmen einer Masterarbeit, einen Datensatz zu erstellen, der EEG- und Eye-Tracking-Daten (EEG-ET) umfasst, der verwendet werden kann, um Modelle zu trainieren, welche die Augenbewegung anhand von EEG-Daten (gemessen mit Consumer Hardware) rekonstruieren können.

Die EEG-Daten werden mithilfe des Muse S Gen 2 Headsets erfasst, einem weit verbreiteten Consumer EEG-Headset mit 4 Trockenelektroden. Zur Erfassung der Augenbewegungen werden einerseits die Positionen der Targets auf dem Bildschirm aufgezeichnet und andererseits wird die Augenbewegung durch eine Webcam aufgezeichnet. Dabei wird lediglich die aus dem Video bestimmte Augenposition gespeichert. Es werden keinerlei Bild- oder Videodaten gespeichert.

Ablauf der Studie

Die Teilnahme an dieser Studie dauert insgesamt etwa 10 Minuten und setzt sich aus folgenden Teilen zusammen: Einführung, Übung und Aufnahme. Dabei werden insgesamt 6 Minuten lang Daten aufgenommen.

1. **Einführung:** In der Einführung werden relevante demografische Daten erfasst, der kamerabasierte Eye Tracker kalibriert, Sie bekommen das EEG-Headset aufgesetzt, dieses wird an Ihren Kopf eingestellt und die Signalqualität überprüft. Außerdem erhalten Sie wichtige Hinweise, die während den Aufnahmen beachtet werden sollen.
2. **Übung:** Vor jeder Aufnahme wird eine kurze Übung durchgeführt, um Sie auf die jeweilige Aufgabe vorzubereiten.
3. **Aufnahme:** Während der eigentlichen Datenerhebung werden EEG- und Eye-Tracking-Daten in vier verschiedenen Situationen aufgezeichnet:

   Level 1 stetig:

   Sie verfolgen mit Ihrem Blick einen Punkt, der sich mehrfach von der Mitte stetig nach links, rechts, oben oder unten bewegt. Dauer: ca. 1 Minute.

   Level 1 Sakkaden:

   Sie verfolgen mit Ihrem Blick einen Punkt, der wiederholt von der Mitte nach links, rechts, oben oder unten springt. Dauer: ca. 1 Minute.

   Level 2 stetig:

   Sie verfolgen einen Punkt, der verschiedene Pfade auf dem Bildschirm stetig durchquert. Dauer: ca. 2 Minuten.

   Level 2 Sakkaden:

   Sie verfolgen einen Punkt, der auf einem Raster umherspringt. Dauer: ca. 2 Minuten.

Sollten Sie noch Fragen haben, wenden Sie sich damit bitte an den Versuchsleiter.

2. Freiwilligkeit und Anonymität

Die Teilnahme an der Studie ist freiwillig. Sie können jederzeit und ohne Angabe von Gründen Ihre Einwilligung zur Teilnahme an dieser Studie widerrufen, ohne dass Ihnen daraus Nachteile entstehen. Die Erhebung der Daten erfolgt vollständig anonymisiert, d. h. an keiner Stelle wird Ihr Name erfragt. Ihre Antworten und Ergebnisse werden unter einer Probandennummer gespeichert, welche nicht auf Sie zurückzuführen ist.

3. Vergütung

Für die Teilnahme an dieser Studie wird **keine** finanzielle Vergütung angeboten. Wir möchten Sie darauf hinweisen, dass Ihre Teilnahme auf freiwilliger Basis erfolgt und nicht entlohnt wird. Wir schätzen Ihr Engagement und Ihre Unterstützung für unsere Forschung sehr und danken Ihnen herzlich für Ihre Teilnahme.

4. Kontaktdaten

Tiago, Vasconcelos Afonso
Tel.: +49 173 4161278
E-Mail: tiago.vasconcelosafonso@stud.h-da.de

5. Einwilligung an der Teilnahme der Studie

Hiermit willige ich freiwillig die Teilnahme an der Studie „Erstellung eines Consumer EEG-ET Datensatzes“ ein. Ich bin ausreichend informiert worden und hatte die Möglichkeit, Fragen zu stellen.

Eine Kopie der Aufklärung und der Einwilligung habe ich erhalten.

_____

Ort, Datum

_____

Name des Einwilligenden in Druckbuchstaben

_____

Unterschrift des Einwilligenden

A.2 DEMOGRAPHIC DATA QUESTIONNAIRE

The following pages contain the demographic data questionnaire that was filled out by each participant before the study. It is available in both English and German.

Participant Number (DO NOT FILL IN): _____

Demographic Data

Age: _____

Gender (male/female/diverse): _____

Handedness (left/right/ambidextrous): _____

Vision Correction (Yes/No): _____

Neurological Disorder (Yes/No): _____

Color Blind (Yes/No): _____

Probandennummer (BITTE NICHT AUSFÜLLEN): _____

Demografische Daten

Alter: _____

Geschlecht (männlich/weiblich/divers): _____

Händigkeit (Links-/Rechts-/Beidhändig): _____

Sehhilfe (Ja/Nein): _____

Neurologische Erkrankung (Ja/Nein): _____

Farbenblind (Ja/Nein): _____

BIBLIOGRAPHY

[1] FA Alturki, K AlSharabi, AM Abdurraqeeb, and M Aljalal. "EEG Signal Analysis for Diagnosing Neurological Disorders Using Discrete Wavelet Transform and Intelligent Techniques." In: *Sensors (Basel)* 20.9 (2020), p. 2505. DOI: 10.3390/s20092505.

[2] Nathaniel Barbara, Tracey A. Camilleri, and Kenneth P. Camilleri. "EOG-Based Gaze Angle Estimation Using a Battery Model of the Eye." In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019, pp. 6918–6921. DOI: 10.1109/EMBC.2019.8856323.

[3] Nathaniel Barbara, Tracey A. Camilleri, and Kenneth P. Camilleri. "Real-time continuous EOG-based gaze angle estimation with baseline drift compensation under stationary head conditions." In: *Biomedical Signal Processing and Control* 86 (2023), p. 105282. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2023.105282. URL: https://www.sciencedirect.com/science/article/pii/S1746809423007152.

[4] Ramina Behzad and Aida Behzad. "The Role of EEG in the Diagnosis and Management of Patients with Sleep Disorders." In: *Journal of Behavioral and Brain Science* 11.10 (2021), pp. 257–266. DOI: 10.4236/jbbs.2021.1110021.

[5] Inessa Bekerman, Paul Gottlieb, and Michael Vaiman. "Variations in eyeball diameters of the healthy adults." English. In: *Journal of Ophthalmology* 2014 (2014). PMCID: PMC4238270, p. 503645. ISSN: 2090-004X. DOI: 10.1155/2014/503645. URL: https://www.hindawi.com/journals/joph/2014/503645/.

[6] J.W. Britton, L.C. Frey, J.L. Hopp, et al. *Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults, Children, and Infants*. Ed. by E.K. St. Louis and L.C. Frey. Chicago: American Epilepsy Society, 2016. Chap. EEG in the Epilepsies. URL: https://www.ncbi.nlm.nih.gov/books/NBK390347/.

[7] Malcolm Brown, Michael Marmor, Vaegan, Eberhard Zrenner, Mitchell Brigell, and Michael Bach. "ISCEV Standard for Clinical Electro-oculography (EOG) 2006." In: *Documenta Ophthalmologica* 113.3 (2006), pp. 205–212. ISSN: 1573-2622. DOI: 10.1007/s10633-006-9030-0. URL: https://doi.org/10.1007/s10633-006-9030-0.

[8] Wen-Sheng Chang, Wei-Kuang Liang, Dong-Han Li, Neil G. Muggleton, Prasad Balachandran, Norden E. Huang, and Chi-Hung Juan. "The association between working memory precision and the nonlinear dynamics of frontal and parieto-occipital EEG activity." In: *Scientific Reports* 13.1 (2023), p. 14252. ISSN: 2045-2322. DOI: 10.1038/s41598-023-41358-0. URL: https://doi.org/10.1038/s41598-023-41358-0.

[9] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[10] R.J. Croft and R.J. Barry. "Removal of ocular artifact from the EEG: a review." In: *Neurophysiologie Clinique/Clinical Neurophysiology* 30.1 (2000), pp. 5–19. ISSN: 0987-7053. DOI: https://doi.org/10.1016/S0987-7053(00)00055-1. URL: https://www.sciencedirect.com/science/article/pii/S0987705300000551.

[11] Szymon Deja. *The comparison of accuracy and precision of eye tracking: GazeFlow vs. SMI RED 250*. Tech. rep. Version 1.1. Accessed: 2024-07-30. Kraków: SIMPLY USER, User Experience Lab, 2013. URL: https://gazerecorder.com/Raport/.

[12] Marc Philipp Dietrich, Götz Winterfeldt, and Sebastian von Mammen. "Towards EEG-based eye-tracking for interaction design in head-mounted devices." In: *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. 2017, pp. 227–232. DOI: 10.1109/ICCE-Berlin.2017.8210634.

[13] Chady El Moucary, Abdallah Kassem, Dominick Rizk, Rodrigue Rizk, Sawan Sawan, and Walid Zakhem. "A low-cost full-scale auto eye-tracking system for mobility-impaired patients." In: *AEU - International Journal of Electronics and Communications* 174 (2024), p. 155023. ISSN: 1434-8411. DOI: https://doi.org/10.1016/j.aeue.2023.155023. URL: https://www.sciencedirect.com/science/article/pii/S1434841123004971.

[14] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. "InceptionTime: Finding AlexNet for Time Series Classification." In: *CoRR* abs/1909.04939 (2019). arXiv: 1909.04939. URL: http://arxiv.org/abs/1909.04939.

[15] Wolfgang Fuhl, Susanne Zabel, Theresa Harbig, Julia-Astrid Moldt, Teresa Festl Wietek, Anne Herrmann-Werner, and Kay Nieselt. "One step closer to EEG based eye tracking." In: *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications*. ETRA '23. Tubingen, Germany: Association for Computing Machinery, 2023. ISBN: 9798400701504. DOI: 10.1145/3588015.3588423. URL: https://doi.org/10.1145/3588015.3588423.

[16] CM Gheorghe, VL Purcărea, and IR Gheorghe. "Using eye-tracking technology in Neuromarketing." In: *Rom J Ophthalmol* 67.1 (2023), pp. 2–6. DOI: 10.22336/rjo.2023.2.

[17] IS Gopal and GG Haddad. "Automatic detection of eye movements in REM sleep using the electrooculogram." In: *Am J Physiol* 241.3 (1981), R217–21. DOI: 10.1152/ajpregu.1981.241.3.R217.

[18] Claire C. Gordon et al. *2012 Anthropometric Survey of U.S. Army Personnel: Methods and Summary Statistics*. Technical Report ADA611869. Final report, Oct 2010-Apr 2012. Natick, MA: Army Natick Soldier Research, Development and Engineering Center, 2014. URL: https://apps.dtic.mil/sti/citations/ADA611869.

[19] 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." In: *Reviews of Modern Physics* 65.2 (1993), pp. 413–497. DOI: 10.1103/RevModPhys.65.413.

[20] Florian Heinrichs, Mavin Heim, and Corinna Weber. "Functional Neural Networks: Shift invariant models for functional data with applications to EEG classification." In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 12866–12881. URL: https://proceedings.mlr.press/v202/heinrichs23a.html.

[21] David F. Hight, Henrik A. Kaiser, John W. Sleigh, and Michael S. Avidan. "Continuing professional development module: An updated introduction to electroencephalogram-based brain monitoring during intended general anesthesia." In: *Canadian Journal of Anesthesia* 67.12 (2020), pp. 1858–1878. DOI: 10.1007/s12630-020-01820-3.

[22] Rob J. Hyndman and Yeasmin Khandakar. "Automatic Time Series Forecasting: The forecast Package for R." In: *Journal of Statistical Software* 27.3 (2008), pp. 1–22. DOI: 10.18637/jss.v027.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v027i03.

[23] Ard Kastrati, Martyna Plomecka, Damian Pascual Ortiz, Lukas Wolf, Victor Gillioz, Roger Wattenhofer, and Nicolas Langer. "EEGEyeNet: a Simultaneous Electroencephalography and Eye-tracking Dataset and Benchmark for Eye Movement Prediction." In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. Curran, 2021. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper files/paper/2021/file/a3c65c2974270fd093ee8a9bf8ae7d0b-Paper-round1.pdf.

[24] Pallavi Kaushik, Amir Moye, Marieke van Vugt, and Partha Pratim Roy. "Decoding the cognitive states of attention and distraction in a real-life setting using EEG." In: *Scientific Reports* 12.1 (2022), p. 20649. ISSN: 2045-2322. DOI: 10.1038/s41598-022-24417-w. URL: https://doi.org/10.1038/s41598-022-24417-w.

[25] 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." In: *Journal of Neural Engineering* 15.5 (2018), p. 056013. DOI: 10.1088/1741-2552/aace8c. URL: https://dx.doi.org/10.1088/1741-2552/aace8c.

[26] Seo-Young Lee, Won-Joo Kim, Jae Moon Kim, Juhan Kim, Soochul Park, and on behalf of the Korean Society of Clinical Neurophysiology Education Committee. "Electroencephalography for the diagnosis of brain death." In: *Annals of Clinical Neurophysiology* 19.2 (2017), pp. 118–124. DOI: 10.14253/acn.2017.19.2.118.

[27] Wei Tuck Lee, Humaira Nisar, Aamir S. Malik, and Kim Ho Yeap. "A brain computer interface for smart home control." In: *2013 IEEE International Symposium on Consumer Electronics (ISCE)*. 2013, pp. 35–36. DOI: 10.1109/ISCE.2013.6570240.

[28] ML Mele and S Federici. "Gaze and eye-tracking solutions for psychological research." In: *Cogn Process* 13 Suppl 1 (2012), S261–5. DOI: 10.1007/s10339-012-0499-z.

[29] Mahiul Muhammed Khan Muqit, Yannick Le Mer, Lisa Olmos de Koo, Frank G. Holz, Jose A. Sahel, and Daniel Palanker. "Prosthetic Visual Acuity with the PRIMA Subretinal Microchip in Patients with Atrophic Age-Related Macular Degeneration at 4 Years Follow-up." In: *Ophthalmology Science* 4.5 (2024), p. 100510. ISSN: 2666-9145. DOI: https://doi.org/10.1016/j.xops.2024.100510. URL: https://www.sciencedirect.com/science/article/pii/S2666914524000460.

[30] S Nagel and M Spüler. "World's fastest brain-computer interface: Combining EEG2Code with deep learning." In: *PLoS One* 14.9 (2019), e0221909. DOI: 10.1371/journal.pone.0221909.

[31] Khune Satt Nyein Chan, C. Srisurangkul, N. Depaiwa, and S. Pangkreung. "Detection of driver drowsiness from EEG signals using wearable brain sensing headband." In: *Journal of Research and Applications in Mechanical Engineering* 9.2 (2021), JRAME–21. URL: https://ph01.tci-thaijo.org/index.php/jrame/article/view/243597.

[32] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. 2nd ed. Springer Series in Statistics. Published: 08 June 2005 (Hardcover), 10 November 2010 (Softcover), 28 June 2006 (eBook). Springer New York, NY, 2005, pp. XIX, 429. ISBN: 978-0-387-40080-8. DOI: 10.1007/b98888.

[33] Aniruddha Rajendra Rao and Matthew Reimherr. "Modern non-linear function-on-function regression." In: *Statistics and Computing* 33.6 (2023). ISSN: 1573-1375. DOI: 10.1007/s11222-023-10299-z. URL: http://dx.doi.org/10.1007/s11222-023-10299-z.

[34] F. Rossi, B. Conan-Guez, and F. Fleuret. "Functional data analysis with multi layer perceptrons." In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. Vol. 3. 2002, 2843–2848 vol.3. DOI: 10.1109/IJCNN.2002.1007599.

[35] SR Research. *EyeLink 1000 Plus Specifications*. Accessed: 2024-07-30. 2017. URL: https://www.sr-research.com/wp-content/uploads/2017/11/eyelink-1000-plus-specifications.pdf.

[36] McCall E. Sarrett, Bob McMurray, and Efthymia C. Kapnoula. "Dynamic EEG analysis during language comprehension reveals interactive cascades between perceptual processing and sentential expectations." In: *Brain and Language* 211 (2020), p. 104875. ISSN: 0093-934X. DOI: https://doi.org/10.1016/j.bandl.2020.104875. URL: https://www.sciencedirect.com/science/article/pii/S0093934X20301346.

[37] Kayo Shinomiya, Hiroshi Shiota, Yaeko Ohgi, Nobuyuki Itsuki, Ray Tabesh, Masashi Yamada, and Masanori Kubo. "Analysis of the Characteristics of Electrooculogram Applied a Battery Model to the Eyeball." In: *2006 International Conference on Biomedical and Pharmaceutical Engineering*. 2006, pp. 428–431.

[38] SL Shishkin, YO Nuzhdin, EP Svirin, AG Trofimov, AA Fedorova, BL Kozyrskiy, and BM Velichkovsky. "EEG Negativity in Fixations Used for Gaze-Based Control: Toward Converting Intentions into Actions with an Eye-Brain-Computer Interface." In: *Front Neurosci* 10 (2016), p. 528. DOI: 10.3389/fnins.2016.00528.

[39] Afonso C. Silva and Hellmut Merkle. "Hardware considerations for functional magnetic resonance imaging." In: *Concepts in Magnetic Resonance Part A* 16A.1 (2003), pp. 35–49. DOI: https://doi.org/10.1002/cmr.a.10052. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cmr.a.10052. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cmr.a.10052.

[40] Rui Sun, Andy S. K. Cheng, Cynthia Chan, Janet Hsiao, Adam J. Privitera, Junling Gao, Ching-hang Fong, Ruoxi Ding, and Akaysha C. Tang. "Tracking gaze position from EEG: Exploring the possibility of an EEG-based virtual eye-tracker." In: *Brain and Behavior* 13.10 (2023), e3205. DOI: https://doi.org/10.1002/brb3.3205. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/brb3.3205.

[41] Tobii Technology. *Accuracy and Precision Test Method for Remote Eye Trackers*. Tech. rep. Version 1.1. Test Specification Version: 2.1.1, February 7, 2011. Tobii Technology, 2012. URL: https://go.tobii.com/Tobii-accuracy-and-precision-test-method.

[42] Tobii. *Tobii Eye Tracker 5*. https://gaming.tobii.com/product/eye-tracker-5/. Accessed: 2024-07-30. 2024. URL: https://gaming.tobii.com/product/eye-tracker-5/.

[43] Florida Atlantic University. *Tech Fee Proposal Building up an eye-tracking research laboratory*. 2013. URL: https://techfee.fau.edu/approvedproposals/Generate.cfm?pid=44 (visited on 07/10/2024).

[44] Wikipedia contributors. *Accuracy and precision — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Accuracy and precision&oldid=1246807706. [Online; accessed 8-October-2024]. 2024.

[45] Francis R. Willett et al. "A high-performance speech neuroprosthesis." In: *Nature* 620.7976 (2023), pp. 1031–1036. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06377-x. URL: https://doi.org/10.1038/s41586-023-06377-x.