EEG-EyeTrack: A Benchmark for Time Series and Functional Data Analysis with Open Challenges and Baselines

Anonymous authors
Paper under double-blind review

Abstract

We present a new benchmark dataset for functional data analysis (FDA), focusing on the reconstruction of eye movements from EEG data. Our contribution is threefold: first, we introduce a challenging dataset collected with consumer-grade hardware under realistic conditions. Second, we propose open challenges and evaluation metrics tailored to FDA applications. Third, we establish baseline results for the primary regression task of reconstructing eye movements from EEG signals using functional neural networks. We report baseline results on both our new dataset and the established EEGEyeNet dataset, which was recorded with research-grade hardware.

1 Introduction

In the future, brain-computer interfaces (BCIs) might offer the possibility of restoring or augmenting sensory perception, such as allowing a blind person to see (Muqit et al., 2024), or even converting thoughts into text (Willett et al., 2023). Currently, electroencephalography (EEG), the technology behind modern BCIs, is used primarily to assist in the diagnosis of neurological diseases (Behzad & Behzad, 2021; Britton et al., 2016). Automated analysis of EEG data is an active area of research. However, there is still a long way to go before brain activity can be reliably analyzed, and used for the development of robust BCIs. A more feasible application of BCIs is EEG-based eye tracking Dietrich et al. (2017); Kastrati et al. (2021); Fuhl et al. (2023); Sun et al. (2023). Typically, electric activity related to eye movements is ignored or filtered out (Croft & Barry, 2000), but it might be used to reconstruct the gaze direction. A reliable and accurate tracking of eye movement opens up new possibilities for BCIs. EEG-based eye tracking has the advantage of requiring no additional hardware when brain activity is already being monitored with an EEG headset. Additionally, it remains effective even when the eyes are closed, such as during sleep. However, current EEG-based eye trackers are mainly based on EEG data recorded with expensive hardware in a laboratory setting. Open datasets recorded with consumer-level hardware outside a lab environment are scarce, yet crucial for the development of EEG-based eye trackers that work reliably in precisely this context.

Two promising fields of statistics for the interpretation of EEG data are time series and functional data analysis (FDA). FDA provides statistical tools for data that may be modelled as smooth functions (of time, space, frequency or any other reasonable argument), see Ramsay & Silverman (2005) and Kokoszka & Reimherr (2017) for foundational overviews. Recent extensions into deep learning give rise to functional neural networks (FNNs), which embed such smoothness directly into network layers (see Rossi et al., 2002; Rao & Reimherr, 2023b; Heinrichs et al., 2023, among others). It can be assumed that developments in these areas will enable robust and accurate eye tracking. However, baseline datasets for functional data are often either too simple or nearly impossible to analyze with modern methods. The aim of this work is two-fold. On the one hand, we introduce the *Consumer-Grade EEG-Based Eye Tracking Dataset* as a new, challenging benchmark for methods in the areas of time series analysis and FDA. For this purpose, open problems and evaluation metrics are established. On the other hand, we investigate the use of methods from FDA for the central problem of eye movement reconstruction from EEG data, which can be considered a scalar-on-function or function-on-function regression problem, and give first baseline results.

To obtain the baseline results, an existing model for eye movement reconstruction, as proposed by Fuhl et al. (2023), was used as a reference. Recently, neural networks with functional layers were specifically designed for the analysis of EEG data. These functional neural networks (FNNs) are expected to better model the smoothness of the data, and therefore require fewer parameters and be more robust to noise compared to other models. As FNNs have not been used for EEG-based eye tracking, we compare the models also on the already existing EEGEyeNet dataset.

The majority of FDA methods requires previous alignment of curves, which is referred to as *function registration*. However, this is only possible if meaningful features can be identified in the data or if events of interest are 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. However, due to the experimental design, the true eye movement data is known and can serve as a basis for curve alignment. This allows methods that require function registration to be applied to the dataset as well.

To summarize, our contribution is as follows:

- **Challenging dataset:** We introduce and publicly release the "Consumer-Grade EEG-Based Eye Tracking Dataset", a large-scale collection of EEG and eye-tracking recordings acquired with consumer-grade hardware under realistic, non-laboratory conditions.
- **Standardized FDA benchmark:** We formalize scalar-on-function regression on the public "Consumer-Grade EEG-Based Eye Tracking Dataset" with standardized evaluation metrics for reproducible FDA benchmarking.
- **Baseline performance:** We provide initial benchmarks by evaluating different functional and conventional models, on both the consumer-grade dataset and the research-grade EEGEyeNet, establishing reference numbers.

2 Related Work

Typically, when recording EEG data, one is interested in brain activity rather than artifacts from muscle or eye movements. The latter are either filtered out, with methods such as independent component analysis (ICA) or canonical correlation analysis (CCA), or completely ignored, especially in deep learning-based pipelines (Urigüen & Garcia-Zapirain, 2015). Recently, however, the reconstruction of eye movements from EEG recordings has been studied in the literature as an independent task. Dietrich et al. (2017) recorded short segments of EEG data, containing extreme eye movements (left, right, up, down), with an EEG headset with 14 wet electrodes. They classified the recordings with a variant of k-nearest neighbors, reaching an accuracy of 96%. Subsequently, Sun et al. (2023) recorded eye movements in a laboratory setting with 64 wet electrodes. The proposed algorithm, EEG-VET, was able to reconstruct *saccadic* (rapid) and smooth eye movements. Recently, the EEGEyeNet dataset was introduced as a benchmark dataset for the problem of reconstructing eye movement from EEG recordings (Kastrati et al., 2021). The dataset contains recordings from 356 subjects, comprising 38 hours of saccadic left-right eye movements, 7 hours and 52 minutes of saccadic movements on a grid, and 1 hour and 29 minutes of free eye movements. The EEG data was recorded in a laboratory setting with an EEG headset containing 128 electrodes. Based on the EEGEyeNet, Fuhl et al. (2023) proposed a neural network architecture, where the first layer should act as a (learnable) spatial filter. This model, referred to as *SpatialFilterCNN* in the remainder of this work, led to new state-of-the-art results for the EEGEyeNet dataset and is used as benchmark for subsequent experiments. Our newly introduced dataset is similar to EEGEyeNet, yet recorded with consumer-grade hardware outside a controlled laboratory setting, that shall allow the development of EEG-based eye trackers in real-world applications. The used EEG-headset had only four dry electrodes, and therefore a substantially lower signal-to-noise ratio. We will refer to our dataset as "Consumer-Grade EEG-based Eye Tracking" dataset.

Functional data analysis (FDA), a field of statistics, deals with smooth processes. If eye movements are recorded with a sufficiently high sampling frequency, the resulting data can be regarded as a smooth function of time. A variety of open datasets for different tasks are commonly used as benchmarks. These datasets

include the "Berkeley Growth Study" (Tuddenham, 1954), daily temperatures from Canadian weather stations and the "Handwriting" dataset (Ramsay & Silverman, 2005). For classification of functional data, the "Phoneme" and "Tecator" datasets are frequently used (Hastie et al., 1995; Thodberg, 2015). Especially for the latter two, very high accuracies ($> 91\%$ and 100%, respectively) have been achieved in the literature (Heinrichs et al., 2023). For future developments in FDA, new and challenging datasets are required. Based on our "Consumer-Grade EEG-based Eye Tracking" dataset, we formulate multiple open challenges in FDA in Section 4.1.

Most FDA methods require data to be *registered*, which means aligning the (functional) data to a common time axis, where each time point contains essentially the same information across all functions (Kneip & Gasser, 1992; Gasser & Kneip, 1995; Ramsay & Li, 1998). This does not only include classic methods, such as functional PCA (FPCA) and functional linear models (Shang, 2014; Cardot et al., 1999; Cuevas et al., 2002), but also modern methods for functional time series, such as tests on stationarity or white noise (Bücher et al., 2020; 2023).

Although curve registration is a common preprocessing step, it is not feasible in many applications, especially when sliding windows are considered. Therefore, different shift-invariant methods have been proposed recently, such as transform-invariant FPCA (Heinrichs, 2025). For classification and regression tasks, functional neural networks have been proposed and extensively studied (Rossi et al., 2002; Rossi & Conan-Guez, 2005; Rossi et al., 2005). Early functional neural networks essentially project the (infinite-dimensional) functions onto multivariate vectors in the first layer and use "standard" layers throughout the remainder. More recently, fully functional neural networks have been proposed for scalar-on-function and function-on-function regression (Rao & Reimherr, 2023a;b). Additionally, convolutional layers have been extended to functional data (Heinrichs et al., 2023). In the remainder, we will use the definitions of functional neurons from the latter reference, as it has been applied to EEG data in the past, and refer to it for explanations of the functional layers and their hyperparameters.

3 Functional Data Analysis

Functional data analysis (FDA) offers a framework for modeling signals, such as EEG recordings, as smooth functions $x(t)$ rather than high-dimensional vectors of discrete time points. With this approach, dependencies between neighboring points are explicitly taken into account. A first step in FDA, usually consists of smoothing the discrete observations to obtain smooth data. Afterwards, the functional data is often projected from an infinite-dimensional function space to a lower dimensional space by dimension reduction techniques, such as functional principal component analysis (FPCA), so that multivariate methods can be used for a subsequent analysis.

Through dimension reduction, important information might get lost, and it can be advantageous to work directly with functional data. Depending on the analysis' goal, it is often assumed that the observations are elements of $C([0,1])$, the Banach space of continuous functions on $[0,1]$, or elements of $L^2([0,1])$, the Hilbert space of square-integrable functions on $[0,1]$.

Let $H = L^2([0,1])$ denote the function space of the data. A central problem of FDA, is the task of regression, that links a functional input to a scalar or functional response. In *scalar-on-function* regression, a scalar outcome y is modeled as $y = F(x) + \epsilon$, where $F : H \to \mathbb{R}$ denotes a (not necessarily linear) functional from the function space H to $\mathbb{R}$. When F is assumed to be continuous and linear, we obtain the linear model

$$y = \alpha + \int \beta(t)x(t)\mathrm{d}t + \epsilon,$$

where $\beta(t)$ is a coefficient function. *Function-on-function* regression generalizes this to predict an entire output curve $y(t)$, e. g., a gaze trajectory. In this case, $y = F(x) + \epsilon$, where $F : H \to H$ denotes an operator. Again, if F is continuous and linear, we obtain the linear model

$$y(t) = \alpha(t) + \int \beta(t,s)x(s)\mathrm{d}s + \epsilon(t).$$

Functional neural networks (FNNs) allow modelling of non-linear relations between input and output. In these networks, discrete sums and convolutions are essentially replaced by their continuous counterparts. For input functions $x_1, \ldots, x_p$, defined on $[0, 1]$, a functional neuron might be defined by

$$y(t) = \sigma\left(w_0(t) + \sum_{i=1}^{p} w_i(t)x_i(t)\right),$$

for functional weights $w_0, \ldots, w_p$. Further, we may extend $x_1, \ldots, x_p$ to $[-b, 1 + b]$, for some bandwidth $b > 0$, by defining the functions as zero in $[-b, 0) \cup (1, 1 + b]$. Then, we can define a functional convolutional layer by

$$y(t) = \sigma\left(w_0(t) + \sum_{i=1}^{p} \int_{-b}^{b} w_i(s)x_i(t - s)\mathrm{d}s\right),$$

for functional weights $w_0 : [0, 1] \to \mathbb{R}$ and $w_1, \ldots, w_p : [-b, b] \to \mathbb{R}$, see Heinrichs et al. (2023). In the following, we consider smooth (continuous or differentiable) weights w. Due to the integration, the output becomes increasingly smooth. Generally, for differentiable weight functions $w_1, \ldots, w_p \in C^1([0, 1])$ and k-times differentiable inputs $x_1, \ldots, x_p \in C^k([0, 1])$, y will be $k + 1$ times differentiable. Moreover, for bounded functions $x_1, \ldots, x_p$, y is continuous and for continuous functions $x_1, \ldots, x_p$, y is differentiable. This observation is crucial for the use of functional neural networks, and explains how deep FNNs filter out local noise.

Beyond regression, FDA encompasses tasks such as classification, clustering curves by shape, and detecting change points in functional time series. In this paper, we focus empirically on scalar-on-function regression, while defining open challenges related to other FDA tasks.

4 Dataset and Open Challenges

We introduce the "Consumer-Grade EEG-based Eye Tracking" dataset, which is publicly available on Zenodo[1]. The dataset contains recordings from 116 sessions of 113 participants. Each session lasted approximately 6 minutes, yielding a total of 11 hours and 45 minutes.

The experiments consisted of a target moving on screen that participants were asked to follow closely. Data from three different modalities was recorded. First, EEG-data was measured at positions TP9, TP10, AF7, AF8 according to the international 10-10 system, where the electrode Fpz was used as reference. Second, the current gaze position, as estimated from a webcam-based eye tracker, was recorded. And finally, the target's position on screen was tracked.

Each session consisted of four stages. In the first two stages ("level-1" recordings), the target moved only horizontally and vertically on screen, while in the latter two stages ("level-2" recordings), the target moved in more directions, yielding more degrees of freedom. Each level had one stage, where the target changed its position abruptly, and another stage, where the target moved continuously across the screen. The former recordings are referred to as "saccades" while the latter as "smooth". Especially, recordings with smooth eye movements are of interest as benchmark for methods in functional data analysis.

Missing values in the data streams, that occurred due to hardware issues, were imputed through a suitable SARIMA model. For an explanation of this process, as well as details regarding the experiments and data acquisition, we refer to Appendix B. For the benchmark experiments, we used the preprocessing as described therein. More specifically, notch filters at 50 Hz and 60 Hz were used, and subsequently a bandpass filter between 0.5 Hz and 40 Hz. Note that the transition from the stopband to the passband of the bandpass filter allows some high-frequency noise to persist. This is mitigated by applying the notch filters beforehand. Additionally, we excluded recordings with known quality issues (recordings from participants 2, 4, 16-20, 62-67, 79 and the "level-1-saccades" recording from participant 50).

Across all 4 EEG channels and 407 recordings, missing samples due to hardware dropouts ranged from 0.0 % to 24.1 % per channel (median 6.3 %), with 27 recordings exceeding 10% of missing data (see Table 1). To

[1] Link to non-anonymous repository

preserve both non-stationary trends and the strong 50 Hz background noise, we imputed missing values with a SARIMA model with seasonal lag 5 and the remaining parameters tuned based on the Akaike Information Criterion.

Statistic	Value (%)
Minimum missing per channel	0.0
Median missing per channel	6.3
Maximum missing per channel	24.1
Recordings with >10% missing	27

Table 1: Summary of missing-data rates across EEG channels and recordings.

For details regarding participant demographics, recording environment, ethics/consent procedures, licensing conditions, and additional information, we refer to Appendix B.

In accordance with the guidelines of the German Research Foundation (DFG), no ethics vote was required for the existing study (German Research Foundation, 2023). More specifically, the DFG states that "A statement by an ethics committee is not usually required for studies involving electrophysiological leads (e.g. EEG, MEG, NIRS) if the above points do not apply.", where the "above points" refer to experiments involving high levels of stress, patients, (f)MRI or psychopharmacological studies; none of which applies here. In accordance with the GDPR, the EEG recordings were anonymized so that no conclusions can be drawn about the study participants.

However, the dataset used contains EEG signals that could potentially be misused for applications such as covert attention tracking or surveillance. Although the dataset itself does not include any personal identifiers, we caution against such misuse.

4.1 Open Challenges

Due to the experimental design, the dataset is well suited to serve as a benchmark for FDA methods. The main challenge of this dataset is the prediction of the target's position from the EEG signal. This can be formulated as a **function-on-function** or as a **scalar-on-function regression** problem, where in the first case the position over time and in the latter case the final position of the target should be predicted from the EEG data. Note that instead of the target's position, the gaze position, as estimated from a webcam-based eye tracker, might be used as a target variable as well.

The dataset can be used as a benchmark for FDA methods that require function registration, as well as for methods that do not require it. In level-1 experiments, the target starts in the center of the screen and moves up, down, left or right. The start of each movement can be used as a marker to divide an entire recording into multiple trials. These trials can be assumed to be aligned curves (or further registered). For methods that do not require registration, sliding windows can be generated from the entire recording.

For benchmark experiments, the originally proposed train-test split (90% training, 10% test) should be kept, and test data from other tasks/levels should not be used for training. For regression, both coordinates of the target, as defined in the columns `Stimulus_x` and `Stimulus_y` should be predicted. As a metric to measure the quality of predictions, we propose the Mean Euclidean Distance (MED) between the predicted and the ground truth stimulus position. The MED is defined as

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

where N_k denotes the number of samples in the k-th recording of the test data, x_i and y_i are the true, and $\hat{x}_i$ and $\hat{y}_i$ are the predicted target positions at time step i. The final score of a model is the MED over all

12 recordings in the test data, which is defined as

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

The MED is a standard metric in eye tracking, because of its easy and clear interpretation (Raghunath et al., 2012; Papoutsaki et al., 2018; Dalrymple et al., 2018). It measures the average distance between the ground truth stimulus position and its corresponding prediction. Other possible metrics would have been the 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.

When reporting the performance of a model, the MED for each task should be reported. This allows for a more nuanced comparison of the performance of different methods, as models could overfit to one task and perform poorly on others, making them unsuitable for general purpose eye tracking.

A schematic overview of the benchmarking pipeline is given in Figure 1 (a).

Besides the main challenge of (scalar-on-function) regression, we identified several additional challenges:

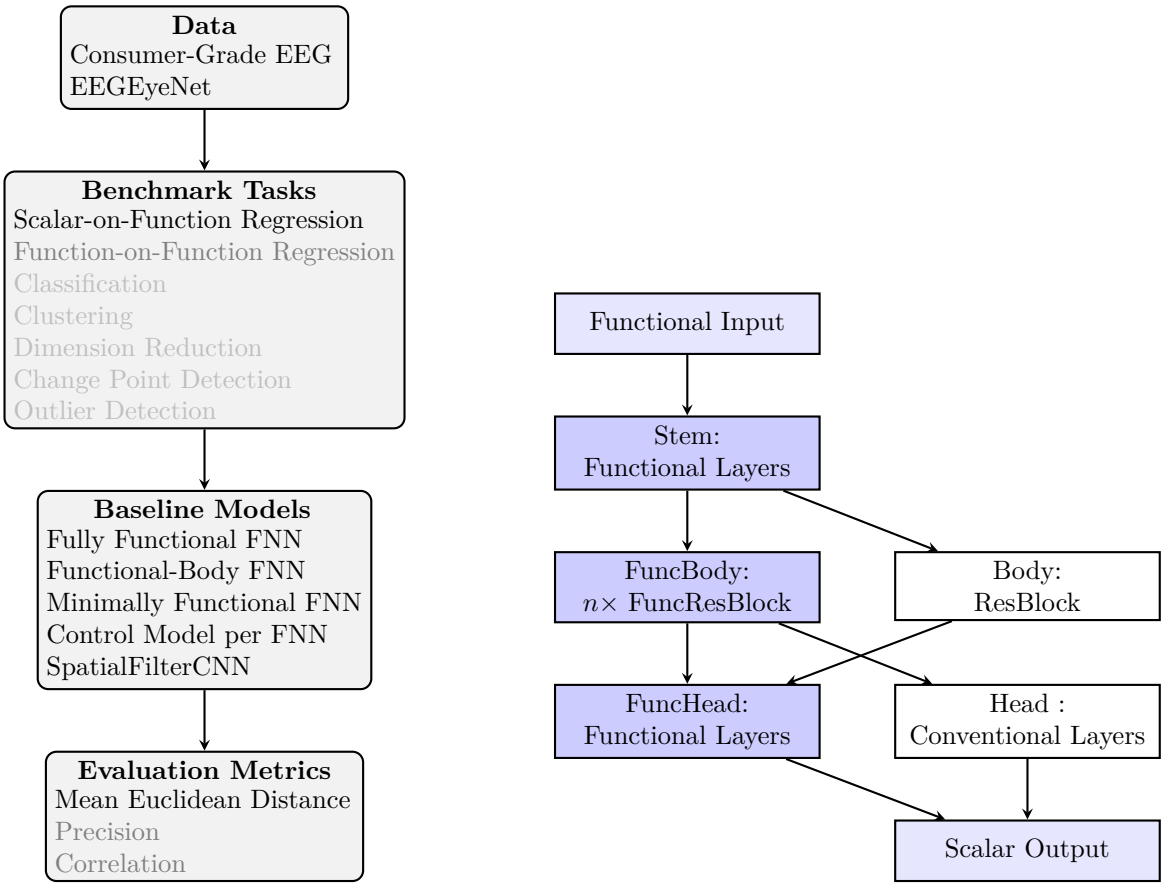
1. **Classification of Movements:** Classify EEG data based on eye movements, e. g., "horizontal" vs. "vertical", "saccades" vs. "smooth", or "up"/"down"/"left"/"right".
2. **Classification of Participants:** Classify participants into different groups with class labels generated from eye movements, e. g., as good or poor trackers, based on the difference between target and gaze position; or as fast or slow trackers, based on the lag between target and gaze position.
3. **Clustering:** Identify brain activity patterns from EEG signals.
4. **Dimension Reduction:** Reduce the dimension of the data while minimizing the reconstruction loss.
5. **Outlier detection:** Identify segments, or time points, with unusual data, e. g. due to missing values that have been replaced by zeros or other erroneous measurements.
6. **Change point detection:** Detect moments where gaze tracking or EEG activity shifts significantly, e. g., due to a loss of attention or transitions between movements and pauses.

5 Model 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 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 FNN, we replaced the standard convolutional layer with a functional convolutional layer, using a large resolution to mimic the large kernel size of conventional CNN. In addition, we placed a spatial filtering layer in front of this 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 8 of Appendix A.1.

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



(a) High-level benchmark pipeline.

(b) FNN architectures decomposed into Stem, Body, and Head. Functional layers shaded blue; input/output layers shaded in light blue; conventional layers left white.

Figure 1: (a) Benchmark pipeline from datasets through tasks to evaluation. (b) Functional neural network variants: every FNN has a stem (functional), a body (functional or conventional), and a head (functional or conventional).

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. Further, we set the kernel length of each filter to 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 9 of Appendix A.1.

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. An overview of the model architectures

is provided in Figure 1 (b), while additional information are provided in the appendix. An implementation of the models under MIT license is available online: link to non-anonymous repository

5.1 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 coincides with a standard convolutional layer. The functional residual block is displayed together with the standard residual block in Table 9 of Appendix A.1. 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: First, pooling operations with a stride break the smooth structure of the signals passing through the network. Second, using only functional layers at the head avoids the "parameter explosion" that would occur after the flattening operation.

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 displayed in Table 10 of Appendix A.1.

5.2 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. 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 ELU activation function, while the second layer has two neurons with a linear activation function, producing the final output. Table 11 of Appendix A.1 provides a detailed breakdown of the second FNN architecture.

5.3 Minimally Functional Architecture

The third FNN architecture uses functional layers sparingly, with only a single functional dense layer in the head and the functional convolutional layer in the stem, that is part of all three architectures. The former 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 variety. 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 12 of Appendix A.1.

6 Experiments

6.1 Metrics

Three metrics were used to evaluate model performance in the experiments: the Mean Euclidean Distance (MED), as described in Section 4.1, the Pearson correlation coefficient between true and predicted trajectories, and the precision. 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.

Note that the term "precision", contrary to its use in classification, refers to a measure of variation of the predicted gaze position. More precisely, we define the precision as the MED between true and predicted direction of eye movement, i. e.,

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

where y_i and $\hat{y}_i$ denote the true and predicted gaze position at time step i.

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 these metrics.

While the MED serves as a primary measure of model performance, we introduced the Pearson correlation coefficient to complement the MED. It can, for example, detect models that predict constant values, such as the mean, as it will be (approximately) zero in this case.

Further, the Pearson correlation is translation and scale invariant. Models that predict generally the right direction, but not the correct position or scale, would be penalized by the MED, but will have a high correlation. This property makes the correlation especially valuable in early stages of model development, where capturing the direction is a positive sign of learning.

It is important to note, however, that while the 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 values.

6.2 Experimental Setup

The experiments were conducted on a Linux server, the hardware specifications of which are outlined in Table 2. The system features an AMD Ryzen Threadripper PRO 7995WX 96-Cores CPU running at up to 5.19 GHz, with 96 physical cores (192 threads). The server is equipped with 502 GiB (539 GB) of RAM. All experiments were run directly on the host operating system, which is Ubuntu 22.04.5 LTS. No GPUs were used in our setup.

All experiments were conducted using Python 3.10.12. Hyperparameter tuning was handled using *Optuna*, with the TPE sampler configured for efficient search through the hyperparameter space.

Component	Specification
CPU	AMD Ryzen Threadripper PRO 7995WX 96-Cores @ 5.19,GHz (96-Core/192 Threads)
RAM	502,GiB DDR4 SDRAM
OS	Ubuntu 22.04.5 LTS
Python	3.10.12

Table 2: Hardware specifications of the Linux server used for the experiments.

7 Results and Discussion

7.1 Consumer-Grade EEG Eye Tracking

In the following, we compare the EEG-based reconstructions of eye movements with different baselines. The first baseline predicts the target's position randomly, and is used to verify if the model learns anything at all. As a stronger baseline, we use the mean position over the entire training set, which corresponds to the center of the screen. While this baseline is still trivial, it helps to check if the model learned something non-trivial. Finally, we use the webcam-based predictions as a strong baseline. This baseline is generally expected to perform well, up to some delay between movement and predictions, and is used for a direct comparison between EEG- and webcam-based eye tracking.

We initially used the SpatialFilterCNN with its original hyperparameters, i. e., $N_S = 16$, $N_1 = 32$, $N_2 = 64$, `spatial_filtering` enabled, and `equally_sized` convolutional layers. 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. Further, we used both filtered and unfiltered data. Details on the architecture of the SpatialFilterCNN are provided in Appendix A.2.

The results suggested that filtered data improves predictions for the "saccades" paradigm, while unfiltered data is better suited for smooth eye movements. Additionally, we conducted hyperparameter tuning experiments, applying the previously identified optimal filtering – filtered data for "saccades" and unfiltered data for "smooth" tasks. Using Optuna, the hyperparameters were tuned by training 20 models for 30 epochs with different hyperparameter configurations, for each of the four tasks. The ranges of the tuned hyperparameters are provided in Table 14 in the Appendix.

As additional baseline models, we used (i) an LSTM with two LSTM layers of 128 units and a dense layer with 2 neurons, and (ii) functional PCA (FPCA) combined with linear regression. In the context of sliding windows, the latter approach is expected to yield trivial results, since FPCA requires curve registration.

We used recordings of participants 1, 20, 28, 42, 52, and 69 as validation data. These recordings were selected as a representative subset from the training data. After hyperparameter tuning, the final model was trained on the entire training dataset.

The functional neural networks were trained exactly as the SpatialFilterCNN, i. e., by using a batch size of 384 for 30 epochs, early stopping at a patience of 5 epochs, and the Adam optimizer with a tuned learning rate minimizing the mean-squared error. As before, we used Optuna to tune hyperparameters, as specified in Table 13.

The results of the level-1 and level-2 experiments are given in Tables 3 and 4, respectively. Additionally, exemplary predictions of different neural networks are displayed in Figure 2.

Overall, all models made better predictions than random guessing. For level-1 experiments, the MED of the mean baseline is comparable or even lower than that of the webcam-based predictions. This effect is mainly due to the fact that level-1 experiments only included horizontal and vertical movements and short pauses in the center of the screen between these movements. The "mean baseline", remained at the origin, and predicted (at least) one coordinate correctly as 0, while the webcam-based predictions vary across both axes. When taking the correlations corr_x and corr_y into account, we see that the webcam-based predictions contain useful information about the target's position, while the mean baseline does not. Among the reference

	level-1-saccades		**level-1-smooth**			
Model	MED	Precision	MED	Precision	corr_x	corr_y
Random	165.90	177.10	143.60	177.40	−0.01	0.00
Mean	83.58	**0.65**	**51.98**	**0.36**	0.00	0.00
Webcam	**81.89**	1.48	74.59	0.89	**0.69**	**0.35**
SFCNN	**54.62**	4.65	57.92	6.09	**0.20**	**0.19**
FPCA	89.71	2.25	50.23	0.75	0.04	0.05
LSTM	83.35	**0.64**	**43.42**	**0.37**	0.00	0.00
FullyFunc	61.82 (±1.41)	**1.71** (±0.05)	56.02 (±2.93)	1.18 (±0.30)	0.21 (±0.06)	0.01 (±0.03)
FuncBody	56.76 (±1.95)	1.86 (±0.18)	67.18 (±3.86)	**1.15** (±0.18)	0.17 (±0.04)	**0.03** (±0.03)
MinFunc	**55.82** (±1.06)	1.83 (±0.30)	**55.53** (±2.78)	1.25 (±0.48)	**0.23** (±0.01)	0.01 (±0.02)

Table 3: Results of the baseline and reference models for level-1 experiments. Functional neural networks were trained 5 times and average values are reported (standard deviation in parentheses). The best result for each metric in each category is marked in bold.

models, the combination of FPCA and linear regression led to (almost) trivial results that were only slightly better than the mean. The LSTM achieved good results for precision and, in smooth experiments, for the MED. However, considering the correlations, the predictions were only slightly better than the mean. Only the SpatialFilterCNN achieved meaningful results, with correlations around 0.2 and considerably low MED. The precision is relatively high, which indicates that successive predictions vary greatly and predicted trajectories are not continuous.

In general, the FNNs seem to fail predicting the target's y-coordinate, but outperform the SpatialFilterCNN in terms of corr_x. This is likely due to the experiment setup, since horizontal eye movements are more pronounced than vertical ones. For saccades experiments, the minimally functional and the functional body neural networks have an MED comparable to that of the SpatialFilterCNN. In case of smooth experiments, the MEDs of minimally and fully functional neural networks are slightly lower than that of the SpatialFilterCNN. Notably, when comparing the SpatialFilterCNNs with the FNNs, the precision of the latter is substantially smaller. This is due to the rather smooth trajectories that the FNNs predict, compared to the rough trajectories of the SpatialFilterCNNs. Note that small values of the precision, as defined in equation 1, are preferable.

For level-2 experiments, which introduce more degrees of freedom, the results look different. First, the "mean baseline" has a substantially higher MED, which stems from the fact that now two non-zero coordinates must be predicted. Overall, the webcam-based predictions yield the lowest MED and the highest correlations corr_x and corr_y. The combination of FPCA and linear regression as well as the LSTM have results similar to the mean, indicating that they collapsed to constant models. For saccades experiments, the SpatialFilterCNN has a lower MED, yet a higher precision than the FNNs. For smooth experiments, FuncBody and MinFunc yield results similar to the SFCNN, in terms of MED and corr_x. Though, their precision is substantially lower, indicating smoother trajectories of predictions.

Generally, the SpatialFilterCNN has the lowest prediction error in terms of the MED. For smooth movements, the FNNs yield similar (level-2) or higher (level-1) correlations along the x-axis compared to the SpatialFilterCNN. This suggests that the FNNs are better at learning movement directions, while the SpatialFilterCNN is better at learning the exact magnitude of movements. The substantially lower precision metric of the FNNs suggests that their predictions are smoother and more consistent compared to those of the SpatialFilterCNN.

7.1.1 Ablation Study

In addition, we performed an ablation study, to evaluate the use of functional layers. For this, we used control models for each architecture, where the functional layers were replaced by conventional layers. More

	level-2-saccades		**level-2-smooth**			
Model	MED	Precision	MED	Precision	corr_x	corr_y
Random	199.80	177.30	164.50	177.50	0.00	0.00
Mean	162.30	**0.59**	104.50	**0.42**	0.00	0.00
Webcam	**87.19**	1.47	**77.00**	1.00	**0.90**	**0.75**
SFCNN	**109.00**	6.45	**99.38**	20.84	**0.43**	**0.16**
FPCA	161.83	2.34	104.62	**0.57**	0.00	0.00
LSTM	162.85	**0.61**	104.14	**0.43**	0.25	−0.02
FullyFunc	123.47 (±1.70)	2.55 (±0.41)	111.17 (±0.91)	1.67 (±0.03)	0.36 (±0.03)	**0.13** (±0.04)
FuncBody	123.11 (±2.69)	3.25 (±0.95)	**99.33** (±0.67)	1.15 (±0.19)	**0.43** (±0.01)	0.11 (±0.02)
MinFunc	**120.93** (±0.97)	**2.00** (±0.22)	99.78 (±1.82)	**1.04** (±0.14)	**0.43** (±0.03)	0.08 (±0.06)

Table 4: Results of the baseline and reference models for level-2 experiments. Functional neural networks were trained 5 times and average values are reported (standard deviation in parentheses). The best result for each metric in each category is marked in bold.

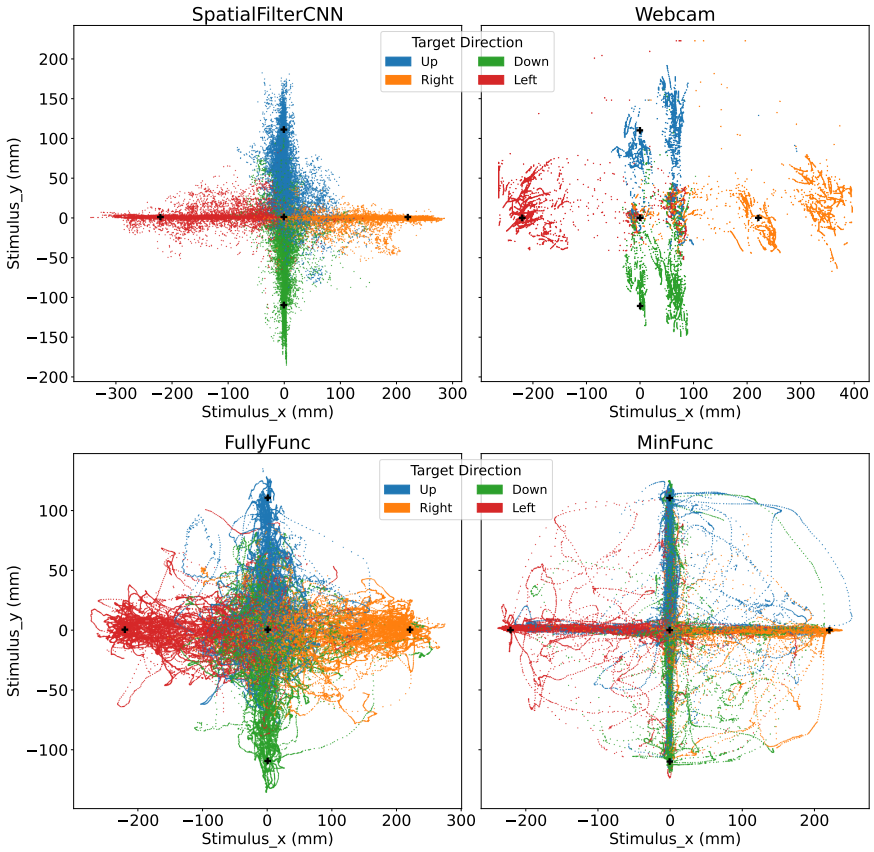


Figure 2: Comparison of various neural networks on the level-1 saccades task. Every prediction made on the test set is shown as a dot. The dots are colored based on the true target position. The four target positions are indicated by black crosses.

	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	**<u>1.772</u>**	**<u>55.11</u>**	**<u>0.882</u>**	**<u>0.285</u>**	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	**<u>64.72</u>**	**1.865**	**56.18**	**0.901**	**0.168**	0.070
MinFunc (control)	71.13	1.920	58.30	1.007	0.154	**<u>0.076</u>**

Table 5: Results of the functional and control models for level-1 experiments. The better result between functional and control model is highlighted in bold, and the best result for each metric is underlined.

	level-2-saccades		**level-2-smooth**			
Model	MED	Precision	MED	Precision	corr_x	corr_y
FullyFunc	**<u>127.5</u>**	**<u>2.276</u>**	**<u>100.8</u>**	**<u>0.913</u>**	**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**	**<u>0.411</u>**	**<u>0.159</u>**
MinFunc (control)	132.2	**2.494**	108.2	1.336	0.348	0.087

Table 6: Results of the functional and control models for level-2 experiments. The better result between functional and control model is highlighted in bold, and the best result for each metric is underlined.

specifically, for the fully functional architecture, the functional residual blocks were replaced by standard residual blocks and the two final functional dense layers were substituted by two convolutional layers followed by a global average pooling layer. The first convolutional layer contained 256 filters with ELU activation, and the second contained 2 filters with a linear (no) activation. Both layers had a kernel size of 12. For the functional body, the functional residual blocks were replaced by standard residual blocks. Finally, for the minimally functional neural network, 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, corresponding to the output of the functional dense pooling layer.

The results are displayed in Tables 5 and 6. For level-1 experiments, the functional versions seem preferable for FuncBody and MinFunc, whereas conventional layers seem preferable in case of the FullyFunc model. In fact, the FullyFunc control model led to the best results for 4 out of 6 metrics. For level-2 experiments, i. e., experiments with more degrees of freedom, the functional versions of all models led to better results. More specifically, the FullyFunc model has the highest MED and lowest precision, and the MinFunc model the highest correlations among all models.

The results indicate that functional layers lead to more reliable predictions, especially when the number of degrees of freedom is high.

7.2 EEGEyeNet

The previous comparison is based on data recorded by consumer-grade hardware. In addition, we compare FNNs with conventional CNNs based on the EEGEyeNet dataset, which was recorded under laboratory-controlled conditions and with research-grade EEG equipment with more electrodes and wet sensors. This allows an evaluation of FNNs under optimal conditions, compared to the real-world conditions from the previous section.

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)	**<u>66.2</u>** (±0.8)	**<u>41.1</u>** (±0.5)
EEGNet	77.3 (±0.3)	48.7 (±0.2)
SpatialFilterCNN	68.8 (±1.4)	42.9 (±0.8)

Table 7: Results of various models on the EEGEyeNet dataset. The mean (and standard deviation) of the MED and MAE 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 MED and MAE are reported in millimeters using a conversion factor of 0.5.

We used the same architectures and hyperparameters as before. Furthermore, we have added the EEGNet, a standard neural network for the analysis of EEG data, to the comparisons.

For training, the Adam optimizer with a learning rate of 0.0001 and a batch size of 64 was used. Further, we employed early stopping based on the validation loss, with patience of 20 epochs. All models were trained for a maximum of 50 epochs, with a window size of 500, as specified by Kastrati et al. (2021). In line with the literature, each model was trained 5 times. The results, including the mean and standard deviation of the MED and mean absolute error (MAE), are summarized in Table 7.

Except for the control model of the fully functional neural network, 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. 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. Unlike the results from Section 7.1, where functional layers showed a clear benefit, the differences between the functional, and control models on the 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.

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

8 Conclusion

We introduced the "Consumer-Grade EEG-based Eye Tracking" dataset as a new benchmark for methods in the field of functional data analysis. Additionally, we stated a number of open challenges (classification, clustering, dimension reduction, outlier and change point detection) related to the dataset. We further studied the use of functional neural networks to solve the main challenge associated with the dataset, reconstructing a target's position on the screen from raw EEG data. This task can be formulated as function-on-function or scalar-on-function regression problem. We provided benchmark results for the latter. While our focus was on the main challenge, the remaining tasks outlined, such as classification and change point detection, offer valuable opportunities for further exploration. Baseline results for these tasks are yet to be established and provide a promising direction for future research.

While the results on the EEGEyeNet dataset were less conclusive, the results from Section 7.1 suggest that functional neural networks have beneficial properties for the prediction of the target's position. The ablation study carried out, in which functional and conventional neural networks were compared, generally showed

better results for the respective functional version. For smooth movements, FNNs achieved similar MED, corr_x and corr_y and substantially lower precision compared to the tuned SpatialFilterCNN. Note that with the definition of "precision" from equation 1, small values are better. The comparably small values for the precision are probably due to the fact that the predictions of FNNs tend to be rather smooth, as illustrated by 2. Overall, the results suggest that FNNs are a powerful tool for predicting smooth targets, which makes them particularly promising for analyzing functional data.

In conclusion, the "Consumer-Grade EEG-based Eye Tracking" dataset allows a comparison of FDA methods across different open challenges. While FNNs have been found to be a helpful method for analyzing the dataset, it remains an open question how well other approaches work. It is expected that progress on this dataset will advance both EEG analysis and functional data analysis.

References

Alexandre Barachant, Dano Morrison, Hubert Banville, Jason Kowaleski, Uri Shaked, Sylvain Chevallier, and Juan Jesús Torre Tresols. muse-lsl, May 2019. URL `https://doi.org/10.5281/zenodo.3228861`.

Ramina Behzad and Aida Behzad. The role of EEG in the diagnosis and management of patients with sleep disorders. *Journal of Behavioral and Brain Science*, 11(10):257–266, 2021.

Jeffrey W. Britton, Lauren C. Frey, Jennifer L. Hopp, Pearce Korb, Mohamad Z. Koubeissi, William E. Lievens, Elia M. Pestana-Knight, and Erik K. St. Louis. Electroencephalography (EEG): An introductory text and atlas of normal and abnormal findings in adults, children, and infants, 2016.

Axel Bücher, Holger Dette, and Florian Heinrichs. Detecting deviations from second-order stationarity in locally stationary functional time series. *Annals of the Institute of Statistical Mathematics*, 72:1055–1094, 2020.

Axel Bücher, Holger Dette, and Florian Heinrichs. A portmanteau-type test for detecting serial correlation in locally stationary functional time series. *Statistical Inference for Stochastic Processes*, 26(2):255–278, 2023.

Hervé Cardot, Frédéric Ferraty, and Pascal Sarda. Functional linear model. *Statistics & Probability Letters*, 45(1):11–22, 1999.

R.J. Croft and R.J. Barry. Removal of ocular artifact from the EEG: a review. *Neurophysiologie Clinique/Clinical Neurophysiology*, 30(1):5–19, 2000. ISSN 0987-7053.

Antonio Cuevas, Manuel Febrero, and Ricardo Fraiman. Linear functional regression: the case of fixed design and functional response. *Canadian Journal of Statistics*, 30(2):285–300, 2002.

Kirsten A Dalrymple, Marie D Manner, Katherine A Harmelink, Elayne P Teska, and Jed T Elison. An examination of recording accuracy and precision from eye tracking data from toddlerhood to adulthood. *Frontiers in psychology*, 9:803, 2018.

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)*, pp. 227–232, 2017.

Benedikt V Ehinger, Katharina Groß, Inga Ibs, and Peter König. A new comprehensive eye-tracking test battery concurrently evaluating the pupil labs glasses and the eyelink 1000. *PeerJ*, 7:e7086, 2019.

Lucas Falch and Katrin Solveig Lohan. Webcam-based gaze estimation for computer screen interaction. *Frontiers in Robotics and AI*, 11, 2024. ISSN 2296-9144. doi: 10.3389/frobt.2024.1369566. URL `https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2024.1369566`.

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, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701504.

Theo Gasser and Alois Kneip. Searching for structure in curve samples. *Journal of the american statistical association*, 90(432):1179–1188, 1995.

German Research Foundation. Faqs on proposal funding process in humanities and social sciences. `https://www.dfg.de/en/research-funding/proposal-funding-process/faq/humanities-social-sciences`, 2023. Accessed: 2025-11-20.

Trevor Hastie, Andreas Buja, and Robert Tibshirani. Penalized discriminant analysis. *The Annals of Statistics*, 23(1):73–102, 1995.

M. Heck, V. Deutscher, and C. Becker. Webcam eye tracking for desktop and mobile devices: a systematic review. In *Proceedings of the 56th Hawaii International Conference on System Science*. Hawaii International Conference on System Science, 2023.

Florian Heinrichs. TI-FPCA: effective and interpretable dimensionality reduction with transform-invariant functional principal component analysis. In *International Conference on Optimization, Learning Algorithms and Applications*, pp. 142–157. Springer, 2025.

Florian Heinrichs, Mavin Heim, and Corinna Weber. Functional neural networks: Shift invariant models for functional data with applications to EEG classification. In *International Conference on Machine Learning*, pp. 12866–12881. PMLR, 2023.

Rob J Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27:1–22, 2008.

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 J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

Alois Kneip and Theo Gasser. Statistical tools to analyze data representing a sample of curves. *The Annals of Statistics*, pp. 1266–1305, 1992.

Piotr Kokoszka and Matthew Reimherr. *Introduction to functional data analysis*. Chapman and Hall/CRC, 2017.

Christian Kothe, Seyed Yahya Shirazi, Tristan Stenner, David Medine, Chadwick Boulay, Matthew I Grivich, Fiorenzo Artoni, Tim Mullen, Arnaud Delorme, and Scott Makeig. The lab streaming layer for synchronized multimodal recording. *Imaging Neuroscience*, 2025.

Logitech. Logitech streamcam technical specifications, 2025. URL `https://www.logitech.com/en-us/shop/p/streamcam`.

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. *Ophthalmology Science*, 4(5):100510, 2024.

Alexandra Papoutsaki, Aaron Gokaslan, James Tompkin, Yuze He, and Jeff Huang. The eye of the typer: a benchmark and analysis of gaze behavior during typing. In *Proceedings of the 2018 acm symposium on eye tracking research & applications*, pp. 1–9, 2018.

Vignesh Raghunath, Melissa O Braxton, Stephanie A Gagnon, Tad T Brunyé, Kimberly H Allison, Lisa M Reisch, Donald L Weaver, Joann G Elmore, and Linda G Shapiro. Mouse cursor movement and eye tracking data as an indicator of pathologists' attention when viewing digital whole slide images. *Journal of pathology informatics*, 3(1):43, 2012.

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

James O Ramsay and Xiaochun Li. Curve registration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(2):351–363, 1998.

Aniruddha Rajendra Rao and Matthew Reimherr. Non-linear functional modeling using neural networks. *Journal of Computational and Graphical Statistics*, 32(4):1248–1257, 2023a.

Aniruddha Rajendra Rao and Matthew Reimherr. Modern non-linear function-on-function regression. *Statistics and Computing*, 33(6):130, 2023b.

Fabrice Rossi and Brieuc Conan-Guez. Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural networks*, 18(1):45–60, 2005.

Fabrice Rossi, Brieuc Conan-Guez, and François Fleuret. Functional data analysis with multi layer perceptrons. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pp. 2843–2848. IEEE, 2002.

Fabrice Rossi, Nicolas Delannay, Brieuc Conan-Guez, and Michel Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

Han Lin Shang. A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, 98(2):121–142, 2014.

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. *Brain and Behavior*, 13(10):e3205, 2023.

Hans Henrik Thodberg. Tecator meat sample dataset. statlib datasets archive, 2015.

Read D Tuddenham. Physical growth of california boys and girls from birth to eighteen years. *University of California publications in child development*, 1(2), 1954.

Jose Antonio Urigüen and Begoña Garcia-Zapirain. EEG artifact removal—state-of-the-art and guidelines. *Journal of neural engineering*, 12(3):031001, 2015.

Francis R Willett, Erin M Kunz, Chaofei Fan, Donald T Avansino, Guy H Wilson, Eun Young Choi, Foram Kamdar, Matthew F Glasser, Leigh R Hochberg, Shaul Druckmann, et al. A high-performance speech neuroprosthesis. *Nature*, 620(7976):1031–1036, 2023.

A Details on Model Architectures

A.1 Functional Neural Networks

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 8: The general structure of the stem, which is used by all three FNN architectures.

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 9: The structure of a standard ResBlock and a FuncResBlock.

Architecture #1: Fully Functional

Layer	Parameters	Output Shape
STEM	—	(batch_size, window_size/2, 64)
FuncResBlock	N_1 filters	(batch_size, window_size/2, 64)
FuncResBlock	N_2 filters	(batch_size, window_size/2, 96)
FuncResBlock	N_3 filters	(batch_size, window_size/2, 144)
FuncResBlock	N_4 filters	(batch_size, window_size/2, 216)
FuncDense	n_functions, basis_type, neurons: 256, activation: elu	(batch_size, window_size/2, 256)
FuncDense	n_functions, basis_type, neurons: 2, activation: linear, pooling: True	(batch_size, 2)

Table 10: Architecture of the "fully functional" neural network. Parameters without specified values are tuned.

Architecture #2: Functional Body

Layer	Parameters	Output Shape
STEM	—	(batch_size, window_size/2, 64)
FuncResBlock	N_1 filters	(batch_size, window_size/2, 64)
FuncResBlock	N_1 filters	(batch_size, window_size/2, 64)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/4, 64)
FuncResBlock	N_2 filters	(batch_size, window_size/4, 112)
FuncResBlock	N_2 filters	(batch_size, window_size/4, 112)
AvgPool	pool_size: 2, strides: 2	(batch_size, window_size/8, 112)
Flatten	—	(batch_size, window_size/8 $\times$ 112)
Dense	neurons: 64, activation: elu	(batch_size, 64)
Dense	neurons: 2, activation: linear	(batch_size, 2)

Table 11: Architecture of the "functional body" neural network. Parameters without specified values are tuned.

A.2 SpatialFilterCNN

A spatial filter, in contrast to temporal filters, is a filter that acts across different electrodes, combining signals at a single time point. Fuhl et al. (2023) proposed a neural network architecture, where the first layer shall act as a (learnable) spatial filter, for the EEGEyeNet data. We refer to this model as *SpatialFilterCNN*.

The model consists of a spatial filtering layer, two residual blocks and two fully connected layers at the output. The detailed architecture of the SpatialFilterCNN model is shown in Figure 3. In 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 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

Architecture #3: Minimally Functional

Layer	Parameters	Output Shape
STEM	—	$(\text{batch_size}, \text{window_size}/2, 64)$
ResBlock	N_1 filters	$(\text{batch_size}, \text{window_size}/2, 64)$
ResBlock	N_1 filters	$(\text{batch_size}, \text{window_size}/2, 64)$
AvgPool	pool_size: 2, strides: 2	$(\text{batch_size}, \text{window_size}/4, 64)$
ResBlock	N_2 filters	$(\text{batch_size}, \text{window_size}/4, 112)$
ResBlock	N_2 filters	$(\text{batch_size}, \text{window_size}/4, 112)$
AvgPool	pool_size: 2, strides: 2	$(\text{batch_size}, \text{window_size}/8, 112)$
FuncDense	n_functions, basis_type, neurons: 512, activation: elu, pooling: True	$(\text{batch_size}, 512)$
Dense	neurons: 512, activation: elu	$(\text{batch_size}, 512)$
Dense	neurons: 2, activation: linear	$(\text{batch_size}, 2)$

Table 12: Architecture of the "minimally functional" neural network. Parameters without specified values are tuned.

Hyperparameter	**Range / Setting**
Temporal filtering	Enabled or Disabled
Learning rate	10^{-5} to 10^{-1}
N_1 (number of filters)	4 to 64
N_2 (number of filters)	8 to 128
N_3* (number of filters)	16 to 256
N_4* (number of filters)	32 to 512
Basis type	Legendre or Fourier
Resolution	12 to 256
Number of basis functions	6 to 128

Table 13: Hyperparameter ranges of the functional neural networks. Hyperparameters marked with "*" correspond to the fully functional neural network.

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. 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 convolutional 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. 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. 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 hyperparameters of the SpatialFilterCNN are given in Table 14.

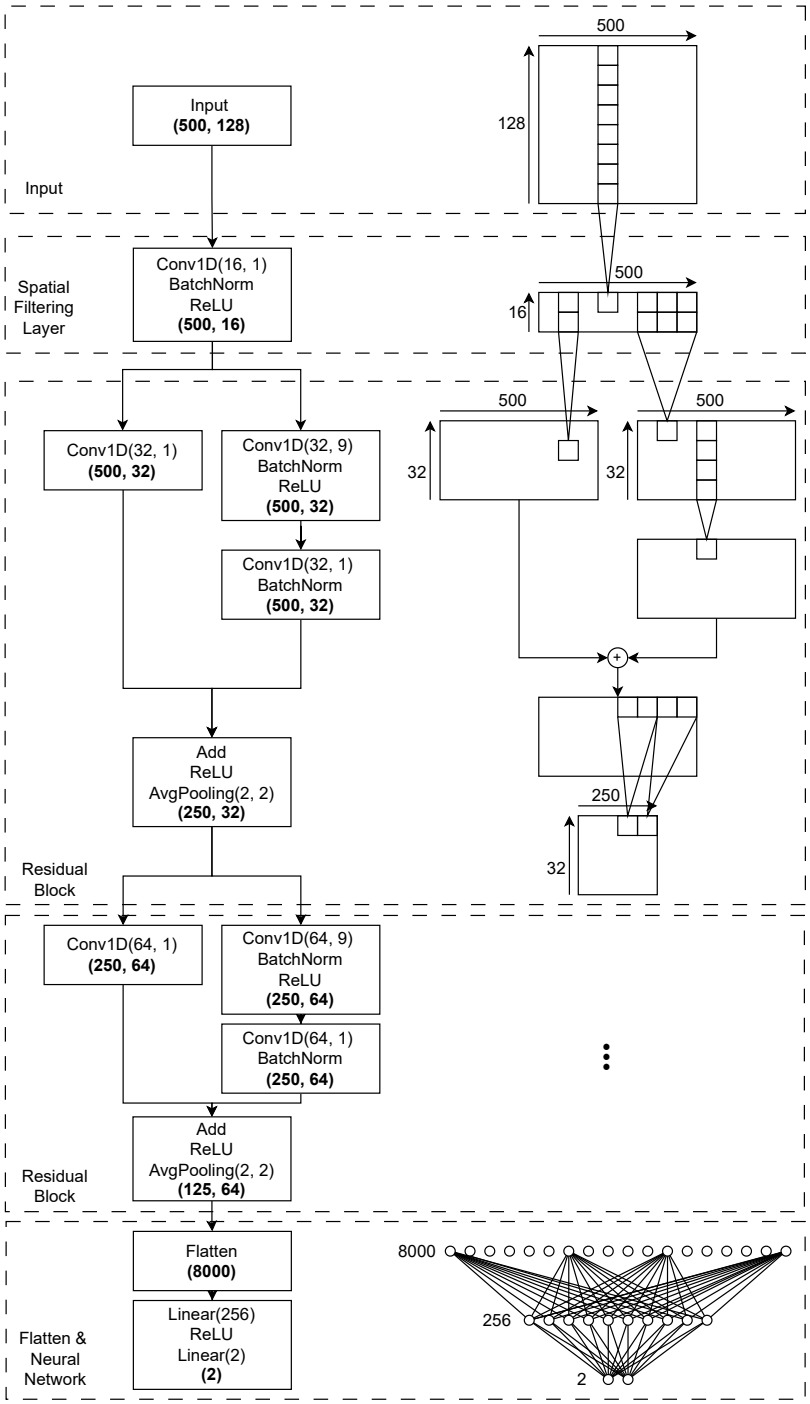


Figure 3: The architecture of the SpatialFilterCNN model, which on a high level consists of the input, a spatial filtering layer, two residual blocks and a fully connected neural network at the output. Conv1D(n, m) denotes a one-dimensional convolutional layer with n filters and a kernel size of m, AvgPooling(n, m) average pooling with pool size n and stride m and Linear(n) a layer of n fully connected neurons.

Hyperparameter	Range / Setting
Window size	128 to 1024 samples
Learning rate	10^{-5} to 10^{-1}
Spatial filtering	Enabled or Disabled
Equally sized kernels	Enabled or Disabled
N_S (number of spatial filters)	4 to 64
N_1 (filters in first residual block)	8 to 128
N_2 (filters in second residual block)	16 to 256

Table 14: Hyperparameter ranges of the SpatialFilterCNN.

B Data Description

B.1 Experiments

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 target 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 target closely.

B.1.1 Participants

A total of 113 participants (91 males and 22 females, 100 right-handed and 12 left-handed, 1 ambidextrous) took part in this study. Demographic data, including the age, gender, handedness, vision correction, neurological disorders, and color blindness of the participants, is summarized in Table 15, the age distribution is shown in Figure 4. The participants were informed about the purpose of the study, the data that would be collected, the duration of the study, and that any collected data would be anonymized. All participants took part in this study voluntarily and received no compensation.

In accordance with the guidelines of the German Research Foundation (DFG), no ethics vote was required for the existing study (German Research Foundation, 2023). More specifically, the DFG states that "A statement by an ethics committee is not usually required for studies involving electrophysiological leads (e.g. EEG, MEG, NIRS) if the above points do not apply.", where the "above points" refer to experiments involving high levels of stress, patients, (f)MRI or psychopharmacological studies; none of which applies here. In accordance with the GDPR, the EEG recordings were anonymized so that no conclusions can be drawn about the study participants.

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. Additionally, participants confirmed that they were taking part in the study voluntarily and were not receiving any compensation. The second document was the demographic questionnaire, where the participants provided information about their age, gender, handedness, vision correction, neurological disorders, and color blindness.

The participant pool shows a noticeable demographic imbalance, with 81% male and 88% right-handed individuals (Table 15). While these proportions reflect the voluntary nature of recruitment within the given university course setting, they may introduce bias in downstream analyses, if ocular artifacts or EEG signal characteristics vary systematically by gender or handedness. This limitation should be considered when interpreting results derived from the dataset.

B.1.2 Setup

The participants were seated in front of a 24 inch desktop monitor with a resolution of 2560 px×1440 px and a 60 Hz refresh rate, placed 60 cm away from the participants and raised high enough such that the center of

Demographic	No. of Participant	Percentage
Gender (male/female/diverse)	91 / 22 / 0	81% / 19% / 0%
Handedness (left/right/ambidextrous)	12 / 100 / 1	11% / 88% / 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 15: Demographic data of the participants.

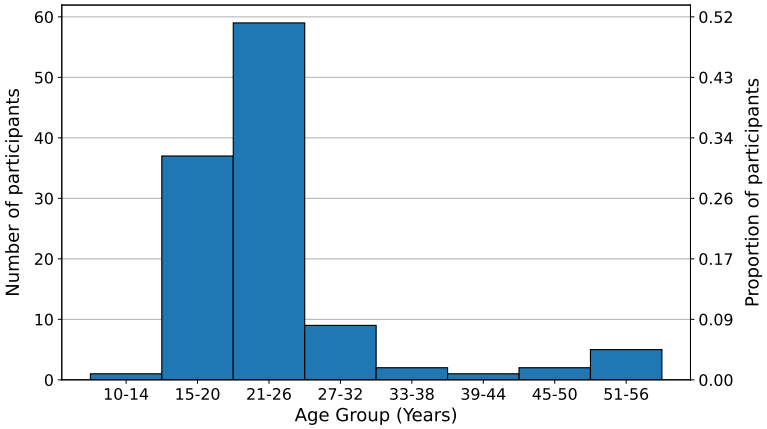


Figure 4: Histogram of the age distribution of the participants. The left y-axis shows the number of participants in each age group, the right y-axis shows the proportion of participants in each age group.

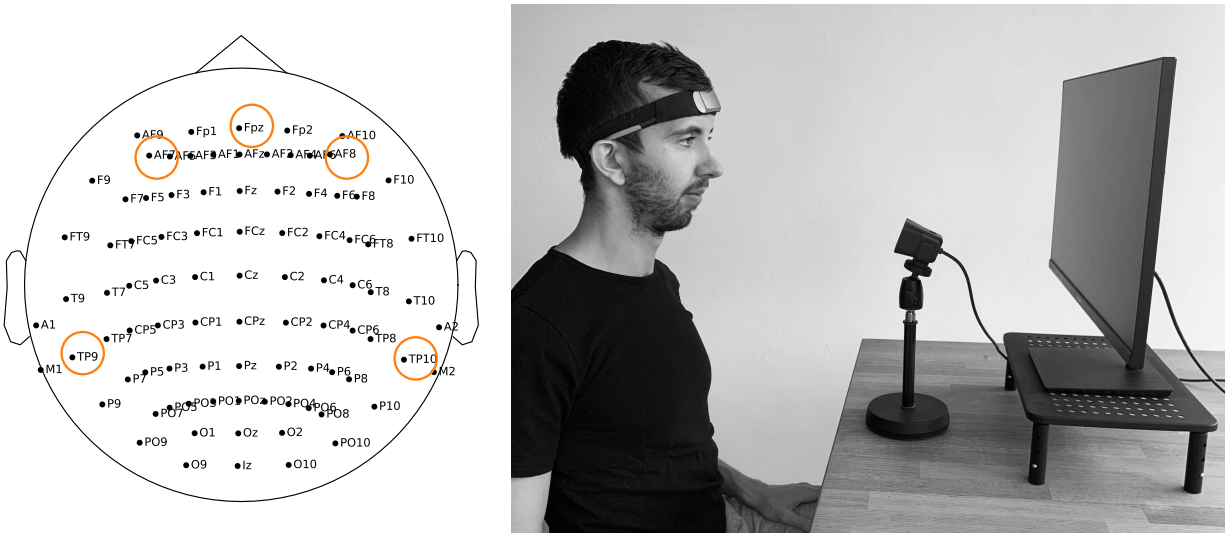


Figure 5: Left: Positions of electrodes of the Muse S 2 Headband in the 10-20 system. Right: Recording setup of the experiments. Consent was obtained for publication of the individual's facial image.

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 on the right of Figure 5.

B.1.3 Hardware

The webcam used was the Logitech StreamCam, which records in Full HD (1920 px×1080 px), at 60 frames per second. The monitor used was a DELL P2416D with a 2560x1440 resolution, 59.95Hz refresh rate, and 8-bit RGB Standard Dynamic Range. The EEG data was recorded using the Muse S 2 Headband (RRID: SCR_014418), a consumer-grade EEG-Headset with five dry electrodes TP9, TP10, AF7, AF8 and Fpz, where the latter is used as reference electrode. The headband records at a sampling rate of 256 Hz. The electrode placement in the standard 10-20 system is shown on the left of Figure 5.

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.

No additional calibration beyond the default manufacturer procedures was performed for either device type. The Logitech StreamCam operated with factory-set optical parameters such as focal length (3.7 mm), aperture size (f/2.0) and field of view (78° diagonal) Logitech (2025). Lighting conditions during sessions were not measured in lux but corresponded to typical indoor classroom illumination. For the Muse S 2 headband, electrode impedance thresholds were not recorded. Signal quality was assessed visually using the provided software. Detailed hardware calibration parameters were not systematically documented, reflecting the practical constraints and real-world focus of this consumer-grade dataset.

B.1.4 Software

Data from three different sources was recorded: the position of the target, the gaze position, and the EEG data. The resulting three data streams were synchronized using the Lab Streaming Layer Protocol (LSL). LSL is an open-source framework widely used for multimodal time-synchronized physiological signal acquisition Kothe et al. (2025). The LSL streams were recorded using the LabRecorder software, which properly resolves the synchronization information of multiple streams. The gaze position was estimated from the webcam video using the GazePointer software.

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 16: Duration of the experimental paradigms.

B.1.5 Experiment Procedure

Most sessions took place during 90 minute practice groups of different artificial intelligence related courses at the Darmstadt University of Applied Sciences. The participants were seated in the back of the room in which the practice group took place. 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 participant's 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 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 16 and was chosen to balance sufficient data collection per condition with participant comfort and to minimize fatigue effects observed during pilot trials. 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-based eye tracking techniques.

In a "smooth" experiment, the target moved across the screen in a continuous manner. The subject followed the target 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 target using a series of quick eye movements called "saccades". Here, the target 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 target. This way, the position of the target 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.

B.1.6 Stimulus Presentation

The target was presented to the participants using a custom stimuli presentation software ("Presentation App") written in Python using the Qt framework. The bounding box in all tasks was 440 mm x 220 mm and the target was presented as a circle with a diameter of 10 mm.

Saccade Experiments. The grid point positions in the level-1 and level-2 "saccades" experiments are shown in Figure 6 (left). 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 target jumped to any of the grid points every 1.5 seconds.

Smooth Experiments. The path along which the target moved in the "smooth" experiments was created using a parameterized curve. More specifically, for a given $t \in [0, 1]$, the x and y position of the target in

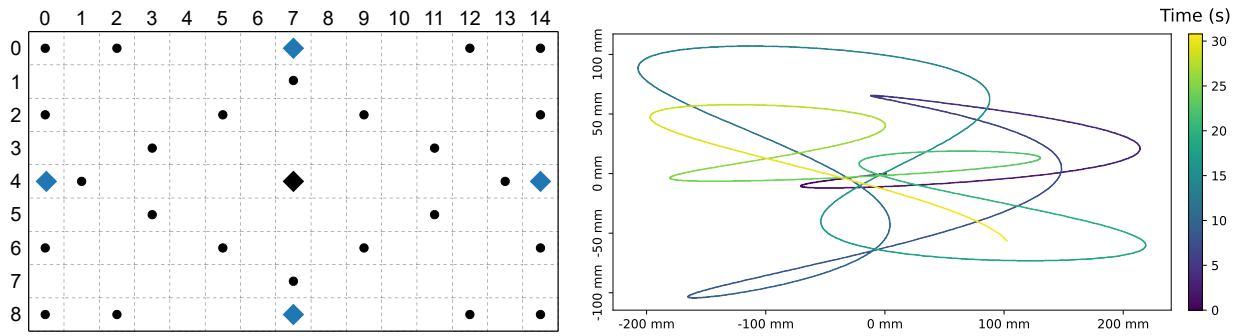


Figure 6: Left: Grid points in the "saccades" experiments. Displayed as blue diamonds (level-1 experiment), black dots (level-2 experiment) and black diamond (both). Right: Exemplary trajectory of the target that was presented to the participants. It was created using the parameterized function equation 2.

millimeters relative to the center of the screen was given by:

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

where w and h are the width and height of the bounding box in millimeters, i. e., $w = 440\,\text{mm}$ and $h = 220\,\text{mm}$. In particular, the x-coordinate describes the horizontal and the y-coordinate the vertical position of the target. The time in seconds the target takes to traverse the complete curve can be controlled by the parameter T. The value of t is incremented 120 times per second, in steps of size $\frac{1}{120}/T$, resulting in a smooth movement.

For all level-1 experiments $a = b = c = d = 0$, $e, f \in \{-1, 0, 1\}$. This results in paths where the target moves continuously from the center to one of the edges of the specified bounding box and back. For these experiments, $T = 1$, for vertical movements and $T = 2$ for horizontal movements. This choice resulted in the target taking 1 second for vertical and 2 seconds for horizontal movements. The different times were chosen to account for the different height and width of the bounding box.

For level-2 experiments a, b, c, d were randomly selected from the interval $[-50, 50]$, $e = f = 0$ and $T = 28.5$, so that the target takes 28.5 seconds to traverse an entire curve. The random selection of parameters results in more complex paths on which the target moves. An example of a path created using this method is displayed in Figure 6 (right). Multiple curves were shown in each experiment. Before the target started to move along the next curve, it waited for 2 seconds in the center of the screen. The value of T was selected so that completing four curves and the three intervening pauses took a total of 2 minutes.

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 level-2 smooth experiments, four curves were shown which were created by randomly picking values for a, b, c, d from the interval $[-50, 50]$. In total, 18 different of these curves were created.

In level-1 saccades experiments, the target jumped 30 times to one of the four directions (left, right, up, down) and back to the center, resulting in 60 saccades in total. To choose the 30 directions for the jumps, 8 random permutations of the four directions were created, and the last two directions from the final permutation were discarded. For the level-2 saccades experiments, 80 positions were selected by creating 4 random permutations of the 25 possible positions. The last 20 positions from the final sequence were then discarded.

Several design choices were made to help the participants anticipate the target's movement, to minimize the distance between the position of the target and the participant's gaze. The presentation differed between the "smooth" and "saccades" paradigms. In the "smooth" experiments, in addition to the yellow circle moving across the screen as the target, a line was displayed to show the expected direction of the target's movement.

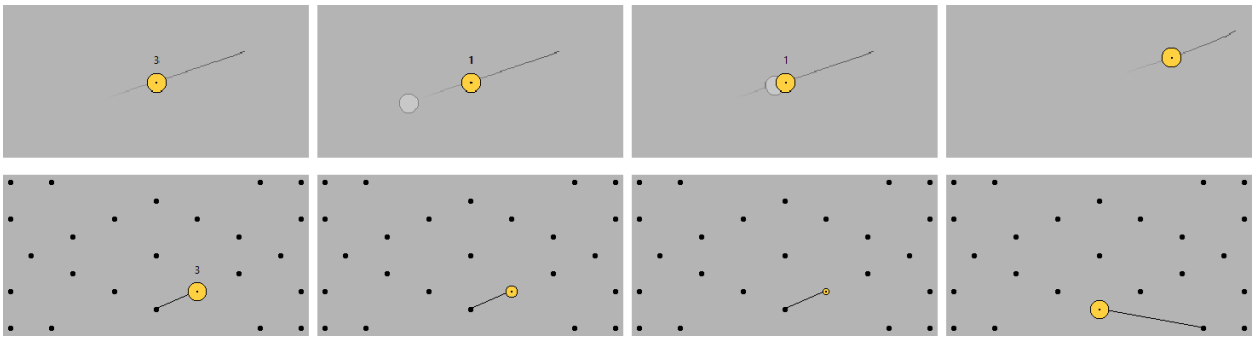


Figure 7: Stimuli presentation. Top: Presentation of the stimulus in the level-2 smooth experiment (countdown at the start - the "ghost" circle starts approaching - the "ghost" circle disappears - a line indicates the movement direction). Bottom: Presentation of the stimulus in the level-2 saccades experiment (countdown at the start - the target shrinks - the target shrinks again - the target jumps).

Additionally, before the target started moving, a "ghost" circle appeared, indicating the start and speed of the movement. See Figure 7 for an example of the stimuli presentation in a "smooth" experiment.

In the "saccades" experiments, a line connected the current position of the target with its next position, i.e., the position to which the target jumped. To indicate the moment when the jump occurred, the circle got scaled down 3 times, completely disappearing the last time. The moment the circle disappeared was the moment the jump happened. This is shown in the lower part of Figure 7.

B.1.7 Data Preprocessing

The suggested preprocessing pipeline consists of two steps, (i) imputation of missing values, and (ii) band-pass filtering. While the first step is computationally expensive, the second is efficient. To facilitate use of the data set, pre-processed data with imputed missing values is provided alongside the raw data and code to filter it.

B.1.8 Missing Value Imputation

Missing values were recorded as zeros, which could also occur as valid values. Thus, 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, a 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 Hyndman & Khandakar (2008), 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 because SARIMA's seasonal component could explicitly model periodic noise patterns inherent in EEG recordings from our setup, such as the particularly strong 50 Hz power line signal, which, at a sampling rate of 256 Hz, corresponds to a period of approximately 5 samples ($256/50 = 5.12 \approx 5$). Figure 8 illustrates the effect of the missing value imputation on the data.

To quantitatively assess the effectiveness and impact of missing value imputation, we calculated several summary statistics across all electrodes and recordings. On average, the percentage of missing values per channel was 6.6% ($\pm$ 8.0%). The large standard deviation indicates that the distribution is skewed, with most

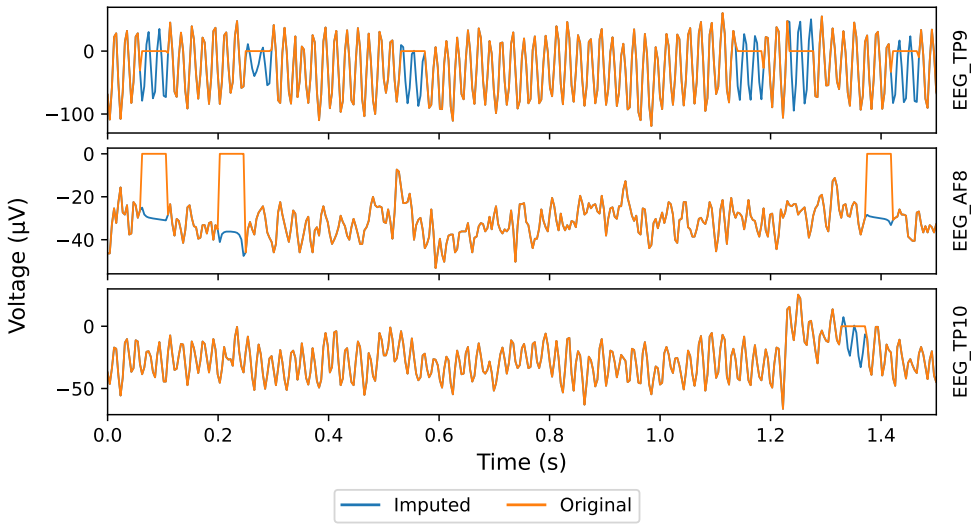


Figure 8: Exemplary imputation of missing values for recording "P045_01 level-1-saccades". Electrode AF7 did not contain missing values in this segment and is not displayed.

Electrode	Imputed Values (%)	Windows With Imputation (%)
TP9	14.96 (± 7.15)	87.16 (± 31.25)
AF7	0.22 (± 4.38)	0.32 (± 4.95)
AF8	8.79 (± 5.82)	78.46 (± 32.85)
TP10	2.52 (± 4.63)	36.72 (± 22.04)
Overall	6.62 (± 8.03)	50.66 (± 43.01)

Table 17: Mean (± SD) percentage of imputed samples and windows with imputed values by electrode.

recordings below average and few recordings with substantially more missing values. This effect is partially explained by differences between electrodes, see Table 17. EEG data were segmented into 1-second windows (256 samples), resulting in an average of 50.7% (± 43.0%) of windows containing at least one missing value. Again, there were major differences between electrodes.

Direct comparison of signal variability between windows with and without imputed values is limited by substantial differences across recordings and by many recordings containing only one window type. As such, pooled or per-recording statistics may not accurately reflect the local impact of imputation.

B.1.9 Filtering

As EEG data is highly prone to noise from various sources, filtering is a common preprocessing step in its analysis. Multiple frequency filters were applied subsequently. 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, such as muscle activity and, baseline drift. All filters were applied forwards and backwards to avoid phase distortion.

The filters were implemented using second-order sections (SOS) instead of directly applying the difference equation, as this was found to be more stable during testing. In Figure 9 the effect of applying the filters on the data is shown. It is evident that the vanilla filter causes the signal to diverge over time, while the SOS-filter keeps the signal stable. Both filters used the combination of 60 Hz and 50 Hz notch and 0.5 Hz to 40 Hz bandpass filter. However, the vanilla filter used an 8th order bandpass filter and applied all filters

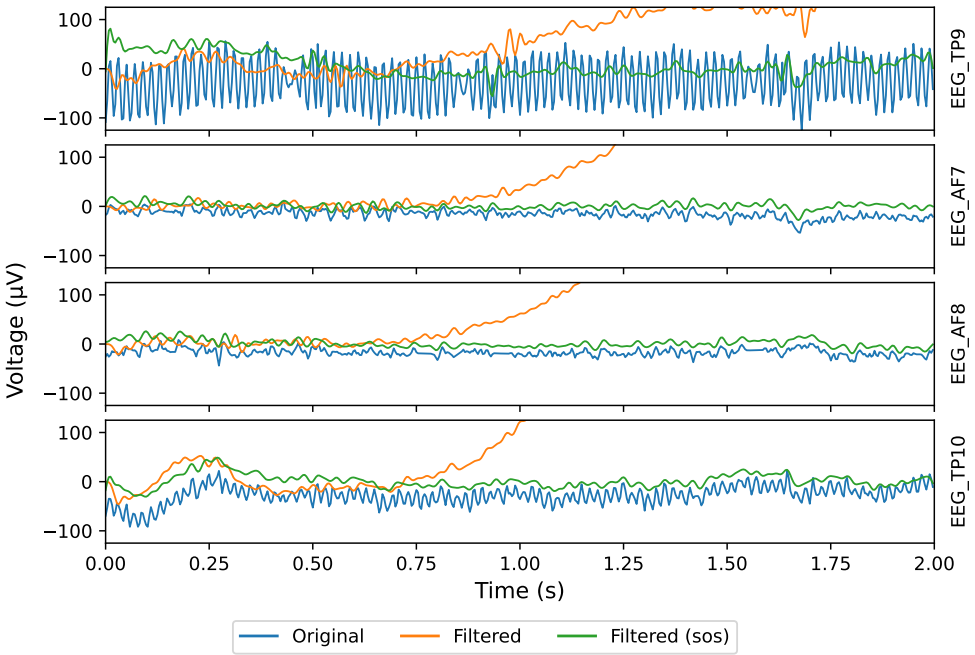


Figure 9: Frequency filters applied to the recording "P045_01 level-2-saccades", where missing values were imputed as described in the above section.

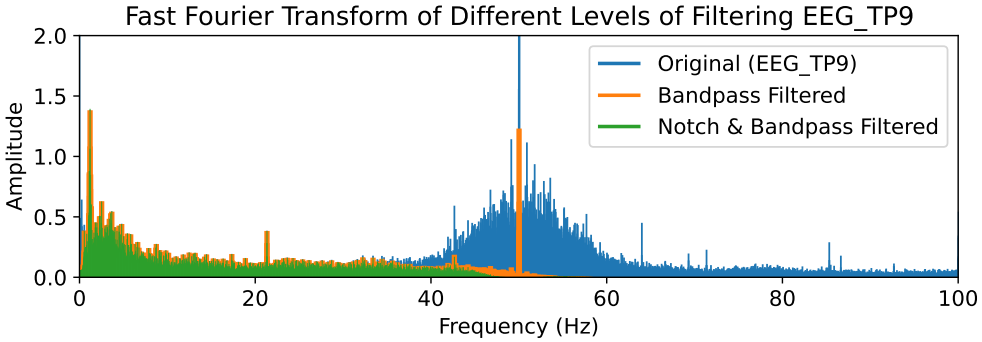


Figure 10: Bandpass filter applied to electrode TP9 of recording "P045_01 level-2-smooth".

only forwards. The SOS-filter used a 4th order bandpass filter and applied all filters both forwards and backwards, doubling the order of the filter and correcting phase shift.

Even though the bandpass filter limits the data to the frequency range from 0.5 Hz to 40 Hz, which should exclude the 50 Hz and 60 Hz noise, we have found that applying the bandpass alone was insufficient to remove the noise entirely. This can be seen in Figure 10 where the noise from the power lines is still present in the data after applying the bandpass filter.

B.2 Data Records

The "Consumer-Grade Electroencephalography and Eye-Tracking Dataset" is openly available on Zenodo (non-anonymous link)

In total, the dataset contains 11 hours and 45 minutes of EEG and eye tracking data. The total time recorded per paradigm and the duration in one session can be seen in Table 16.

The dataset is available as a single denormalized `CSV` file, which contains all the data collected during the sessions. The `CSV` file has a size of 1.42 GB and the following columns: `Participant_no`, `Task`, `Session_no`, `Timestamp`, `EEG_TP9`, `EEG_AF7`, `EEG_AF8`, `EEG_TP10`, `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", which contain one file per recording. When machine learning models are created to predict the true eye position from the EEG data, only the training data should be used for fitting the model, while the test data should be used to evaluate the fitted model. The filename always follows the 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-saccades
│   ├── 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-smooth
│   └── ...
├── level-2-saccades
│   └── ...
└── level-2-smooth
    └── ...
```

In the "csv" folder, the files are `CSV` files with columns `Timestamp`, `EEG_TP9`, `EEG_AF7`, `EEG_AF8`, `EEG_TP10`, `Gaze_x`, `Gaze_y`, `Stimulus_x`, `Stimulus_y`. A detailed description of the columns is provided in Table 18.

The "xdf" folder contains `XDF` files with the raw recorded data. Those files contain additional metadata of the multiple data streams. Users interested in more detailed, time-series structured data may find the `XDF` files useful.

The stimuli presentation software sends the current program status via a LSL stream. In particular, this includes a start and end marker. The first line of the `CSV` file is the first measurement after the start marker. Each line corresponds to a further EEG value. Since other signals (gaze position, program state, position of the target) do not change with the same frequency as the EEG data is recorded, these values always correspond to the most recent ones at this point in time. The last line of the `CSV` file is the last measurement before the end marker. In addition, the absolute positions in px from the `XDF` file are converted to positions relative to the center of the bounding box in mm.

Both the `CSV` file containing the complete dataset, and the "csv" and "xdf" folders are available on Zenodo.

B.3 Technical Validation

In this section, we describe the technical validation of the data collected using a consumer-grade BCI headset and a webcam-based eye tracker. Given the nature of the experiments, there was a consistent trade-off between obtaining high-quality data and maintaining a realistic, non-laboratory setting. While the consumer-level hardware offered the advantage of a more natural and accessible data collection environment, it also posed challenges related to signal accuracy and reliability. This section outlines the steps taken to assess

Source	Column Name	Data Type	Description
LSL	Timestamp	float	Timestamp from synchronized data streams
EEG	EEG_TP9	float	EEG signal from electrode TP9
	EEG_AF7	float	EEG signal from electrode AF7
	EEG_AF8	float	EEG signal from electrode AF8
	EEG_TP10	float	EEG signal from electrode TP10
Eye Tracker	Gaze_x	float	(estimated) x coordinate of the gaze on screen
	Gaze_y	float	(estimated) y coordinate of the gaze on screen
Presentation App	Stimulus_x	float	x coordinate of the target on screen
	Stimulus_y	float	y coordinate of the target on screen

Table 18: Data structure description.

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 headgear

Table 19: List of recordings with known quality issues..

the quality of the data and address potential issues inherent in using such devices outside of controlled lab conditions.

B.3.1 Known Quality Issues

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 no 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 might be excluded from subsequent experiments.

In total, 14 out of 116 recorded sessions (12.1%) were flagged as having known quality issues (Table 19). The most common problems were faulty hardware leading to abnormal electrode readings (7.0%) and Bluetooth interference in EEG transmission affecting multiple consecutive sessions (4.3%). The affected recordings represent approximately 64 minutes, or about 9.3% of the total recording time. While some recordings may still contain usable segments, users should treat them with caution and consider excluding them or applying additional preprocessing depending on their analysis goals.

B.3.2 EEG Data

The aim of the experiments was to create a representative dataset of EEG data recorded with consumer-grade hardware in a realistic environment. For the EEG data, we had to carefully decide when the signal-to-noise ratio became too low, even for this purpose. At the beginning of each recording, the signal quality was assessed visually. The EEG data was visualized using a viewer from the Python library `muselsl` Barachant et al. (2019), which applies a bandpass filter between 1 and 40 Hz by default. The filtered signal was then checked to ensure it was primarily below 30 µV. This naïve heuristic helped identify recordings whose signal-to-noise ratio was too low even for a consumer-level BCI headset. A more systematic approach was avoided in order to collect representative and realistic EEG signals.

The data contains missing values due to incorrect transmission, which were stored as zeros. In the "Methods" section, we explain the approach that was used for the imputation of missing values.

B.3.3 Stimulus Presentation

Several measures were implemented, to ensure that participants could effectively follow the target on the screen during the experiment. As described in the "Methods" section, the direction and timing of the target's movement could be easily predicted due to visual cues. The target was displayed as a circle that got scaled down three times before jumping to the next position. This gradual reduction in size should help the participants to better estimate the time of the next jump. Further visual cues allowed participants to anticipate the next location or direction of movement, such as a "ghost" dot and a line hinting in the latter direction, cf. Figure 7.

In addition to the visual cues, the participants tried out the various tasks in a trial run before the recording. This ensured that participants were familiar with the target's behavior and could follow it better.

These measures were crucial for maintaining the integrity of the data, while finding a balance between realistic conditions and the need for accurate eye tracking. By making the target visually clear, predictable, and easy to follow, we aimed to optimize the participant's ability to complete the task successfully, thus improving the reliability of the recorded data.

B.3.4 Eye Tracking Data

The gaze position, as obtained from the eye tracking software, should not be mistaken for the "ground truth" of eye movements. In this study, the exact on-screen coordinates of the moving target were recorded with pixel-level precision. Given that participants were instructed and visually cued to follow this target closely throughout each trial, these target positions provide a practical proxy for true gaze location in subsequent analyses. Accordingly, when comparing EEG-based gaze predictions with webcam-based estimates, both should ideally be evaluated against the recorded target trajectory rather than directly against one another.

The data from the camera-based tracker was captured using a consumer-grade webcam and eye tracking software (GazePointer), which, although effective in non-laboratory settings, introduces certain inaccuracies and limitations. Its spatial accuracy depends on factors such as lighting conditions, camera angle, and calibration quality. External influences like head movement or changing illumination can further degrade performance. Published evaluations report an angular error between 1.4° and 1.9° Heck et al. (2023); Falch & Lohan (2024), which at our viewing distance of approximately 60 cm corresponds to about 15-20 mm on screen. By contrast, research-grade systems such as Tobii or EyeLink typically achieve <0.9° accuracy (< 9.5 mm at 60 cm) Dalrymple et al. (2018); Ehinger et al. (2019).

In addition to spatial imprecision, small temporal lags between actual eye movements and their detection can arise from both software processing delays and hardware limitations. While generally minor, these lags may introduce noticeable discrepancies during rapid saccades or other fast eye movements. These constraints are inherent to experiments based on consumer-grade hardware/software combinations, and must be considered when interpreting model performance using this dataset.

B.3.5 Correlation Between Target and Gaze

The target positions correspond exactly to the location of the target on the screen, while the gaze positions are estimated by the eye tracking software. It is anticipated that the gaze will lag behind the target. To validate this relationship, we calculated the cross-correlation functions between the x and y coordinates of the gaze and target, respectively (see, e. g., Figure 11). Subsequently, we calculated the lag of maximal cross-correlation for each recording. The averages of these lags across all recordings are reported in Table 20.

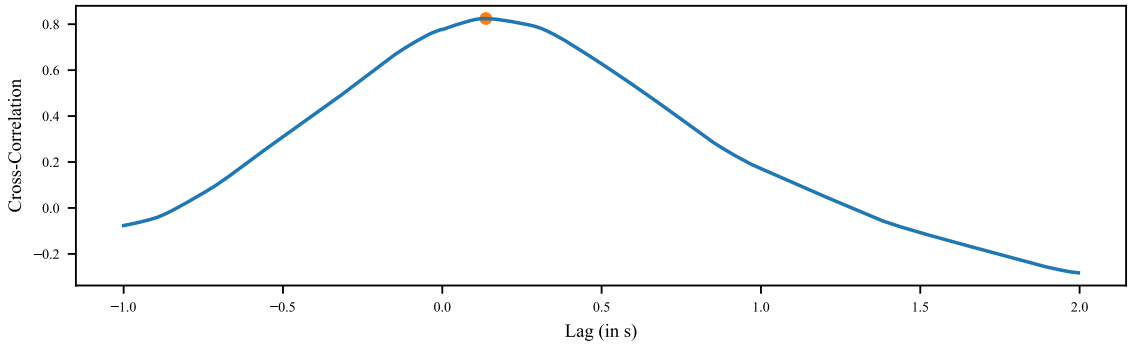


Figure 11: Cross-correlation function between y coordinates of gaze and target for the level-1 "saccades" recording of participant 21.

Experimental Paradigm	Avg. lag (x)	Avg. lag (y)
level-1-saccades	0.18 s (± 0.13 s)	0.23 s (± 0.13 s)
level-1-smooth	0.36 s (± 0.20 s)	0.38 s (± 0.20 s)
level-2-saccades	0.16 s (± 0.20 s)	0.19 s (± 0.18 s)
level-2-smooth	0.26 s (± 0.16 s)	0.32 s (± 0.18 s)

Table 20: Average lag between target and gaze movement across all recordings, determined via cross-correlation function between stimulus and gaze coordinates per axis. Errors represent standard deviations across recordings' lag estimates within each paradigm.

B.4 Usage Notes

When working with the dataset, there are several aspects to take into account. EEG data is highly noisy even when recorded in a laboratory. The signals measured with consumer-grade hardware have a low signal-to-noise ratio. Therefore, it is recommended to preprocess the data first, or use the provided preprocessed data. Exemplary code to load the preprocessed data can be found in the GitHub repository (non-anonymous link).

To get started with the dataset, we recommend using `CSV` files. For users who are familiar with the `XDF` format, we recommend using the latter, as it contains additional information and metadata.

When evaluating EEG-based gaze predictions against webcam-based estimates, we recommend using the recorded target position as proxy for the ground truth, since webcam-derived gaze positions are subject to spatial inaccuracies (between 1.4° and 1.9°, corresponding to 15-20 mm at 60 cm).

When the dataset is used to train and evaluate machine learning models, only the training data should be used for model training, model selection and hyperparameter tuning. The test data should only be used for a final evaluation, and, in particular, not for model selection or hyperparameter tuning. For the latter tasks, the training data might be split into "training" and "validation" data.

It should be noted that, although demographic information such as gender and handedness was collected during the study, these attributes have been removed from the publicly released dataset to ensure participant anonymity. The participant pool is imbalanced (81% male; 88% right-handed), which could theoretically influence EEG-based eye tracking model performance if such factors affect ocular artifacts or gaze behavior. While users cannot directly control for these variables, the provided train-test split preserves the same underlying demographic ratios across both sets, helping to mitigate systematic bias between training and evaluation phases.

A subset of the dataset (12.1% of all sessions, ≈ 9.3% of total recording time) is flagged for potential quality concerns such as Bluetooth interference or faulty electrodes. These files are listed in Table 19. Depending

on their tolerance for noise and missing data, users may wish to exclude these sessions entirely or apply additional preprocessing before model training.

Generally, we assume that methods from time series analysis are suitable for interpreting the data. The "smooth" experiments were specifically designed so that the dataset can serve as a benchmark in functional data analysis, and we expect that the corresponding methods are particularly suited for these recordings.

B.5 Data Availability

The "Consumer-Grade Electroencephalography and Eye-Tracking Dataset" is openly available from Zenodo: Non-anonymous link (Zenodo record XXXXXXXX, DOI: xx.xxxx/zenodo.XXXXXXXX). The repository contains: (1) a single denormalized CSV file (1.42 GB) with all recordings and (2) separate "csv" and "xdf" folders organized by paradigm (level-1-saccades, level-1-smooth, level-2-saccades, level-2-smooth) with train/test subfolders and per-recording files (`PXXX_YY.csv` or `PXXX_YY.xdf`). The XDF files contain the raw multi-stream data and metadata. The CSV files provide time-aligned samples. The complete dataset comprises 11 hours 45 minutes of EEG and eye-tracking data.

B.6 Code Availability

The code used for the recordings is written in Python and is publicly available on GitHub (Non-anonymous link).The code was developed with Python (3.11), Matplotlib (3.9.2), NumPy (1.26.4), Pandas (2.2.3), Pooch (1.8.2), SciPy (1.14.1), Statsmodels (0.14.4), and respective dependencies.

The repository contains the following key components:

- `stimuli-presentation-app`: Code for the application used to record the data.
- `analyse_data.py`: Script to calculate the lag between gaze and stimulus through the cross-correlation function, as described in the "Technical Validation" section.
- `impute_missing_values.py`: Script to impute missing values, as described in the "Methods" section.
- `xdf_to_csv.py`: Script to convert the raw data from XDF to CSV format.

Most importantly, the repository includes the `load_data` module, which provides functions to load and filter the data. The `load_data` functions leverage the `pooch` library in the backend to automatically fetch the required dataset from Zenodo, when it is called for the first time.