Learning-Based Detection of Events in Eye-Tracking Data

An Investigation Into Small-Scale Models for Automotive Applications

Master’s thesis in Engineering Mathematics and Computational Science

Martin Due

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Learning-Based Detection of Events in Eye-Tracking Data

An Investigation Into Small-Scale Models for Automotive Applications

Martin Due

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2024

Learning-Based Detection of Events in Eye-Tracking Data
An Investigation Into Small-Scale Models for Automotive Applications
Martin Due

Supervisors:
Vincent Molin, Department of Mathematical Sciences
Raimondas Zemblys, Smart Eye AB

Examiner: Axel Ringh, Department of Mathematical Sciences,

Master's Thesis 2024
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Picture of human eye generated by Stable Diffusion.

Typeset in LaTeX
Gothenburg, Sweden 2024

Learning-Based Detection of Events in Eye-Tracking Data
An Investigation Into Small-Scale Models for Automotive Applications
Martin Due
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Detection of eye movement events in eye-tracking data is integral to various research fields and commercial applications. Traditionally, the detection has been accomplished either by hand or with threshold based detectors with the inherent drawback that thresholds levels and other parameters had to be hand picked for each scenario. In recent years machine learning methods have been employed for eye movement event detection that do away with that requirement. However, these models tend to be too large to run on limited resources in embedded applications, particularly automotive ones. This thesis focuses on creating models that are smaller but with retained performance. Five machine learning methods were evaluated and hyperparameters were tuned to create well performing small models. Furthermore, the usage of synthetic data for training was investigated, both as a supplement to real data and as a sole source of training data. The study found that a Multilayer Perceptron model (MLP) trained on a combination of real and synthetic data struck the best balance between size and performance. Additionally, results show that models trained purely on synthetic data also performed reasonably well. The findings of this thesis suggest that small, efficient models can effectively detect eye movement events, with potential applications in automotive contexts.

Keywords: Eye movements, Event detection, Fixations, Saccades, Machine Learning, Model Optimization.

Acknowledgements

I would like to thank my supervisors Raimondas Zemblys and Vincent Molin for their supervision and guidance during my thesis. Without your invaluable suggestions and insights this thesis could not have been completed. I would also like to thank my future colleagues at Smart Eye for helping me succeed and making me feel welcome from the very start of this thesis work.

Lastly, I would also like to thank my family and friends for always supporting and encouraging me. I am very grateful that you are a part of my life.

Martin Due, Gothenburg, September 2024

Nomenclature

Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CNN	Convolutional Neural Network
GT	Ground Truth
I-DT	Identification by Dispersion Threshold
IRF	Identification by Random Forest
I-VT	Identification by Velocity Threshold
kNN	k-Nearest Neighbour
MLP	Multi-Layer Perceptron
PCC	Pearson Correlation Coefficient
PSO	Post Saccadic Oscillation
RF	Random Forest
RMS	Root Mean Square
STD	Standard Deviation
SVM	Support Vector Machine

Contents

List of Figures

List of Tables

1

Introduction

Eye tracking refers to using technology to track the movements of the eye. In recent years its use has soared in many disciplines, particularly in research. The research disciplines that make use of eye-tracking include psychology, reading research, psycholinguistics and neurophysiology [1]. It also has numerous commercial applications in areas such as human-computer interaction [2], and drowsiness monitoring in vehicles [3].

There are multiple types of eye-trackers, but what they all have in common is that their output consists of information about where a user is looking. To successfully make conclusions about human behaviour from such data it has to be processed somehow. A common way to do that is to divide the data into segments that represent the different states that the eye can be in. These states are called oculumotor events and they include fixations, where the gaze pauses on a point of interest for some time, and saccades, where the eye rapidly moves to a different position. Although other types of events exist [4], in this thesis, we will limit ourselves to considering only these two. The detection of these events can either be done manually by a human, which is a very time consuming process, or it can be done automatically with software.

Automatic event detection has been a field of study for several decades, with some work being published as early as 1967. It involved placing a contact lens with a stalk on the eye of the recording subject, whose movements were recorded onto an analogue magnetic tape through the use of an intricate system. The magnetic tape was then digitized and analyzed on a computer [5]. The traditional event detection algorithms that were employed are based on thresholds. A significant drawback with such algorithms is that the user is required to manually set those thresholds for what constitutes as an event. Reasonable threshold levels tend to vary between datasets and individuals. Some approaches with adaptive thresholds have been presented [6], but they perform suboptimally on too noisy data [7].

With learning-based methods, such manual decisions are not necessary because, with adequate training data, the algorithm can learn to detect events even at moderate noise levels present in the data to be analyzed.

1.1 Background and Related Work

1.1.1 Traditional algorithms

Some of the first algorithms for automatic detection of eye-tracking events that are still commonly in use today were proposed in the year 2000 [8]. Here an explanation of two of those will be provided to give context for understanding more complex methods in modern event detection research.

The I-VT (Identification by Velocity Threshold) identifies eye movement events by analyzing the angular velocity of the gaze. A threshold for the speed of the eyes movement is set. Samples above that threshold are classified as saccade samples and samples below as fixations.

The I-DT (Identification by Dispersion Threshold) uses the fact that fixation points, due to their low velocity, tend to be grouped together. I-DT identifies fixations as groups of points that are within some maximum separation that is set by the user. Since fixations have a minimum duration, a corresponding threshold for this duration must also be established.The algorithm uses a sliding window and checks the dispersion of the points in the window. The dispersion in each window is calculated as $D = [\max(x) - \min(y)] + [\max(y) - \min(y)]$ where x and y are the gaze pitch and heading respectively. If the dispersion is below the maximum separation threshold the entire window represents as a fixation. Then the window is expanded by adding additional points to the right, until the dispersion goes above the threshold. This process continues with the window moving right until it reaches the end of the dataset that is being analyzed [8].

In 2022 a study found that these methods are outperformed by CNN and Random Forest based methods in all performance metrics that were evaluated except saccade recall [9].

1.1.2 Learning based methods

In recent years several learning based methods for eye movement event detection have been proposed. These methods use machine learning or statistical learning approaches where a model has to be trained to make predictions. That is in stark contrast to the previously discussed traditional algorithms that all rely on predefined rules or heuristics. Two studies that proposed learning based methods will be covered here.

The first one is known as IRF (Identification by Random Forest) [10]. It relies on feeding a set of pre-calculated features based on gaze direction data into a random forest model. The features are window based, so for each sample and feature a value is calculated based on the surrounding gaze direction samples. The main drawback of this model is that it requires a large amount of memory to run.

The second one is called gazeNet, and it uses a radically different approach compared to IRF. Instead of calculating a set of features for each sample, the raw gaze direction data is fed into a CNN, and eye movement event predictions are generated. This can be thought of as an end-to-end approach. [11]

Both these methods have in an independent review been found to generally outperform traditional methods [9].

1.1.3 Smart Eye

Smart Eye is a company that specialises in software and hardware that allow for insights into human behaviour. Their primary products are eye-tracking systems for Advanced Driving Assist Systems (ADAS) in cars, and a system targeted at behavioural research.

The system for behavioural research is called Smart Eye Pro, and it is used in a multitude of settings, ranging from aviation and aerospace to psychology and neuroscience.

1.2 Objective and Method

The purpose of this thesis is to create and evaluate models for the detection of events in eye-tracking data. A special emphasis is put on producing small models, in terms of memory usage, so that they can be run on low-performance embedded hardware in vehicles. The thesis will address a three questions related to this, the first of which is:

> *Can a data-driven model for the detection of events in eye-tracking data, with satisfactory performance, be created while maintaining a small model size?*

One of the main objectives of the thesis is to create a model that can be used to detect saccades and fixations. Although existing literature includes successful attempts for these tasks, they often do not explicitly state model size as a constraint. This thesis aims to replicate those results while focusing on minimizing the model's size.

In the context of machine learning the collection of relevant training data is something that large amounts of resources is devoted to. In some cases it is possible to artificially create synthetic data to supplement the real data, hence the second question that will be addressed is:

> *Can synthetic data be used in combination with real data to increase performance?*

If real world data collection can be avoided all together it would be even better. Therefore, it is of interest to investigate if a model with satisfactory performance can be created through training on synthetically generated data exclusively, leading to the third question:

> *Can purely synthetic data be used to train a model with sufficient performance?*

1.3 Scope

There are at least 8 different type of events that can occur in eye tracking data [12], but in this thesis only two of them are considered: fixations and saccades. The reasoning behind that delimitation is twofold. Firstly, those event types are by far the most commonly occuring in comparison to other event types. Secondly, detection of those event types in particular are especially important in automotive contexts. Detection of saccades can be used to infer different psychological states, such as drowsiness and intoxication, both of which are helpful to detect in automotive settings [13][14]. Since saccades and fixations are both the most prevalent event types and their detection has very meaningful applications the decision to focus solely on them in this thesis has been made.

1.4 Thesis Outline

The second chapter reviews the underlying theory necessary to understand the following chapters. It begins with a short look into the human eye and eye tracking technology. Thereafter supervised learning is discussed, first with a general mathematical formulation based on the expected value, and thereafter in terms of specific algorithms. Then different methods for evaluating classification results are considered.

The third chapter discusses the experimental methods used in the thesis. It begins with an overview of the different datasets that are used, and special emphasis is placed on the dataset that was collected and annotated specially for this thesis.

In the fourth chapter the results are presented. First the performance of different models and combinations of datasets are presented, and then a selection of the two best models is made. These two models are evaluated further, and some factors that impact model performance are investigated. Then some metrics derived from the classification results are presented. Lastly, some qualitative comparisons to the existing Smart Eye Pro classifier is made.

In the fifth and final chapter the key findings are summarized and interpreted, and conclusions are drawn. In the end an overview of potential future research is presented.

2
Theory

This chapter aims to introduce the necessary theory for this thesis. It starts with an overview of how the human eye moves and how modern eye tracking works. Then a description of the models used is given, and finally an introduction to the modern literature about evaluating results in the field of eye movement event detection.

2.1 Movement of the eye

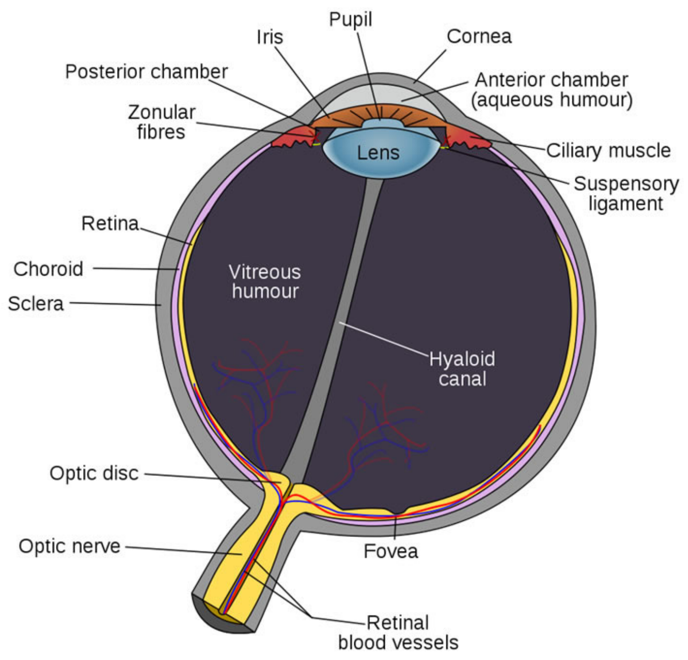


Figure 2.1: Schematic view of the human eye, with key parts labeled. The small region in the back of the eye called the fovea is of particular interest. For us to have a sharp view of something light has to hit the fovea. That is the reason behind the need for humans to move their gaze. Image belongs to the public domain and is taken from [15].

Due to the physiological structure of the eyes the human visual system does not process the entire visual input at the same resolution. In the back of the human eye there is a small area with a high amount of cone photorecptors known as the fovea. It spans about 5° of the visual field. At the center of the fovea there is an 1° area with thin and even more tightly packed cones. Light that gets projected in that narrow region of the eye is perceived by us with the highest resolution. Our objective in observing objects is therefore to direct the light towards the central area of the fovea, a task facilitated by the movement of both the head and, notably, the eyes [4].

Typically the gaze lingers around a point for 200 to 300 ms before it jumps to another one. This lingering is known as a **fixation** [1]. The fast jump to a different point is called a **saccade**. Saccades typically last 30 to 80 ms. After the saccade there is sometimes a bit of a transient, where the eye oscilliates around the next fixation point before settling in. That is called a Post Saccadic Oscillation (PSO) [16]. These can make it difficult to distinguish where a saccade ends and where a fixation begins.

2.1.1 Eye tracking

Video based eye trackers consist of two parts, infrared illumination and video camera(s). These systems can either be head mounted or, more commonly, statically mounted [17]. An example of how a statically mounted system could look is given in Figure 2.2.

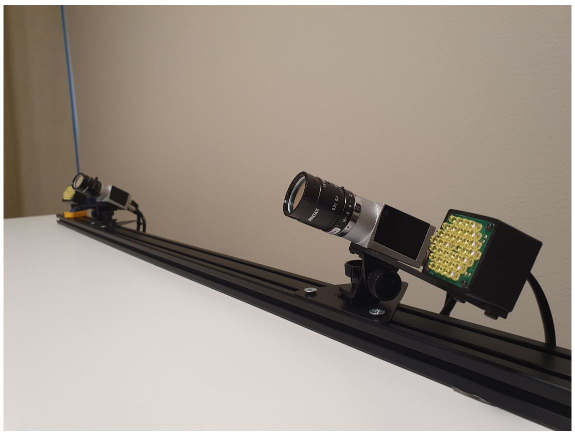


Figure 2.2: A statically mounted eye-tracking system with two cameras, and one infrared light for each camera. This system in particular is a Smart Eye Pro system [18].

2.2 Supervised learning

Supervised learning is a type of machine learning where the model is trained on a dataset with labels, which means that each training example comes with an associated label. More precisely, in this setting we are given a set of examples $\{(x_i, y_i)\}$, assumed to be approximately related by $y_i \approx f_{\text{true}}(x_i)$. The goal of supervised learning is to approximate f_{true} with a function or model that maps inputs to outputs such that it can be used to accurately predict the output for new, unseen inputs. It is possible to either let machine learning algorithms extract features from the input data, or to hand craft features based on expert knowledge. In this work the latter approach has been used.

2.2.1 General mathematical formulation

The following formulation is based on [19]. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ represent the training dataset, where $x_i \in \mathcal{X}$ is the input (feature) vector, $y_i \in \mathcal{Y}$ is the corresponding output (label). $\mathcal{X}$ and $\mathcal{Y}$ is the input and output space, respectively. The dataset $\mathcal{D}$ is assumed to be independently drawn from an unknown joint distribution $P(x, y)$ on $\mathcal{X} \times \mathcal{Y}$.

The goal is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that maps inputs to outputs. To evaluate the quality of the function f it is evaluated using a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ where $L(y, \hat{y})$ measures the difference between the true output and the predicted output, $\hat{y} = f(x)$.

The average loss is called risk, and is given by

$$R(f) = \mathbb{E}_{(x,y)\sim P(x,y)}[L(y, f(x))] = \int_{\mathcal{X}\times\mathcal{Y}} L\left(y, f(x)\right) dP(x, y). \tag{2.1}$$

When learning the optimal function f, this is what we want to minimize. The challenge stems from the fact that the probability measure $P(x, y)$ is unknown, and only the training data $\mathcal{D}$ is given. Hence the task of the learning algorithm is to minimize the risk without having the ability to evaluate it directly. Therefore the empirical risk can be minimized instead, which is an approximation of the risk based on the dataset $\mathcal{D}$:

$$R_{\text{emp}}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)). \tag{2.2}$$

The empirical risk can be thought of as a sample-based estimate of the risk.

After these definitions we are now ready to state the supervised learning problem. It can be formulated as an optimization problem where the objective is to find the function f^* that minimizes the empirical risk

$$f^* = \arg\min_{f\in\mathcal{F}} R_{\text{emp}}(f), \tag{2.3}$$

where $\mathcal{F}$ represents the hypothesis space, i.e. the set of all possible functions that the learning algorithm can choose from. It turns out that this in general is a computationally very hard task. Only in specific simple cases it is possible to find an efficient algorithm or closed form solution for minimizing the empirical risk. The methods for supervised learning used in this thesis all represent implementations of the general procedure described above.

2.2.2 Binary classification

Binary classification refers to the process of assigning samples to one of two categories based on some attributes of those samples [20]. This is the type of learning problem that is the focus of this thesis.

2.2.2.1 Binary cross entropy and Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence is a fundamental concept in statistics used to measure the difference between two probability distributions. It can be used to quantify the dissimilarity of one probability distribution from another. As such, it can be used to evaluate classification results or to compare distributions. It is defined as

$$D_{\mathrm{KL}}(P \parallel Q) := \sum_k P(k) \log \left(\frac{P(k)}{Q(k)} \right), \tag{2.4}$$

so it is zero when the two distributions are equal everywhere, i.e. $P(k) = Q(k)\ \forall\ k$, since $\log(1) = 0$. Note that D_{KL} is asymmetric with regard to the two distributions. In particular, the contribution to D_{KL} is small from terms where $P(k)$ is close to zero but $Q(k)$ is large. In the context of comparing some ground truth distribution to a predicted distribution this asymmetry can be helpful. The asymmetry allows the KL-divergence to place more emphasis on regions where the predicted distribution assigns high probability, but the true distribution does not. This helps identify areas where the predictive model is confidently wrong.

According to [21], equation (2.4) can be rewritten as

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_k P(k) \log P(k) - \sum_k P(k) \log Q(k) = -\mathbb{H}(P) + \mathbb{H}(P, Q), \tag{2.5}$$

where $\mathbb{H}(P, Q)$ is called the cross entropy, and

$$\mathbb{H}(P, Q) := -\sum_k P(k) \log Q(k). \tag{2.6}$$

In the special case where there are only two classes this simplifies to

$$\text{Binary Cross Entropy} = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right], \tag{2.7}$$

where n is the number of samples, y_i is the label of sample i, and p_i is its corresponding probability. The binary cross entropy is also known as the log loss for binary classification.

2.2.3 Logistic regression

Logistic regression is a statistical method used for binomial classification tasks [22]. Given some feature vector ϕ it predicts the probability of an event e_1 occurring,

$$p(e_1|\phi) = y(\phi) = \sigma(\mathbf{w}_1^{\mathrm{T}} \phi + \mathbf{w}_0) \in (0, 1). \tag{2.8}$$

Here $\mathbf{w}$ is a vector of weights and σ is the sigmoid function,

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{2.9}$$

The loss function that is commonly used when finding the optimal weights is the log loss as in equation (2.7). To avoid overfitting the weights to specific data points in the training set an additional regularization term can be added to the loss function. One such regularization term is called L2-regularization or ridge regression. It constitutes of the sum of squares of all weights. Hence large weights are penalized, which leads to all weights being around the same size and close to zero. That decreases the risk of over reliance on a single feature, and thereby reduces overfitting [23].

2.2.3.1 Advantages and limitations

A key advantage of logistic regression is the small model size. For a M-dimensional feature space the model only has $\mathcal{O}(M)$ adjustable parameters, which are the weights for each feature [22]. Since this is a linear model it has a high degree of interpretability, in the sense that if the model fits the data well it is possible to read out the importance of each feature by looking at the relative sizes of the weights. A drawback of the method is that it is a linear model, which is unable to capture non-linear relationships in the data. Another factor to consider is that if the different input variables are correlated with each other, the effect of each on the regression model becomes less precise [24].

2.2.4 k-Nearest Neighbour (kNN)

The k-Nearest Neighbour algorithm is a fundamental and widely used supervised machine learning technique for classification and regression tasks. For each data point that is to be classified it assigns the class label of the majority of the k nearest neighbours. A visualization of this procedure for the case where there are two classes and k=3 is given in Figure 2.3.

To find the nearest neighbours a distance metric has to be used. In $\mathbb{R}^q$ it is common to use the Minkowski metric (also known as the p-norm), which is the Euclidean norm when $p = 2$. For a query point $\mathbf{x}'$ the distance to a training point $\mathbf{x_j}$ is

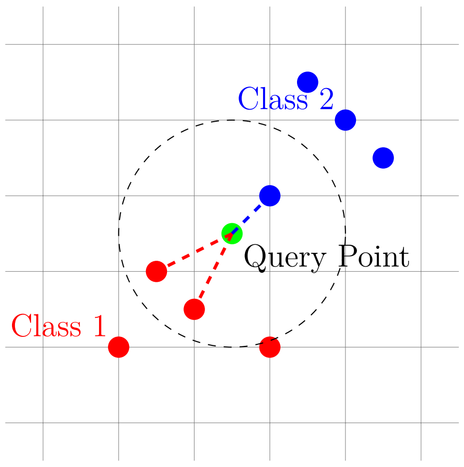
$$d(\mathbf{x}', \mathbf{x}_j) = \|\mathbf{x}' - \mathbf{x}_j\|_2 . \tag{2.10}$$

Then the points are sorted, such that

$$d\left(\mathbf{x}', \mathbf{x}_{(1)}\right) \le d\left(\mathbf{x}', \mathbf{x}_{(2)}\right) \le \cdots \le d\left(\mathbf{x}', \mathbf{x}_{(n-1)}\right) \le d\left(\mathbf{x}', \mathbf{x}_{(n)}\right). \tag{2.11}$$

Finally, a label $\hat{y}$ is assigned to $\mathbf{x}$ from the set of the C possible labels, $\{1, 2, \ldots, C\}$, based on the majority vote of among the labels of the k nearest neighbours $\mathbf{x}_i$ identified in (2.11):

$$\hat{y} = \text{argmax}_{c \in \{1,2,\ldots,C\}} \sum_{i=1}^{k} \mathbf{1}\left(y_{(i)} = c\right), \quad \mathbf{1}[a = b] = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{if } a \neq b. \end{cases} \tag{2.12}$$



Decision: Class 1 (based on majority vote of k=3)

Figure 2.3: Visualisation of the k-Nearest neighbour algorithm with k=3. Here two of the three closest points belong to class 1, so the query point gets classified as belonging to that class.

Here, $y_{(i)}$ denotes the label corresponding to the i-th nearest neighbour $\mathbf{x}_{(i)}$ to $\mathbf{x}'$, and $\hat{y}$ represents the predicted label for $\mathbf{x}'$.

2.2.4.1 Advantages and limitations

kNN models are capable of capturing non-linear relationships. Their flexibility can easily be adjusted by tuning the k value. A higher value creates a more robust and rigid model whereas a lower value leads to a more flexible model.

A significant drawback of kNN models is that they tend to be quite large since their size grows at the rate $\mathcal{O}(n)$ with the size of the training dataset. Furthermore, the inference time grows at the rate $\mathcal{O}(n\log(n))$ with the size of the training dataset since the step in (2.11) requires the list of distances to be sorted. These two problems are likely exacerbated when handling complex functions by the fact that those require a large amount of training data.

2.2.5 Support Vector Machines

A support vector machine is a powerful class of supervised learning algorithms used both for regression and classification. In a binomial classification setting a support vector machine attempts to find the optimal hyperplane such that the margin between the two classes is maximized. The margin is the distance between the hyperplane and the nearest points from each class [25].

2.2.5.1 Advantages and limitations

Linear SVMs are very simple and quite interpretable. The decision boundary is a hyperplane, and the magnitudes of the coefficients that describe this hyperplane give insights into which features are important. They are also very computationally efficient, so they scale well with large data sets and high dimensional data.

However, linear SVMs can only model linear relationships between features and classes. They may not perform well on features where the decision boundary is highly non-linear. To improve performance on complex datasets feature engineering can be used to make the data more linearly separable, however this requires additional effort.

2.2.6 Random Forest

A random forest is a widely used ensemble learning method for classification or regression. The random forest consists of multiple decision trees that are trained on different subsets of the features and training data [26]. At the end, predictions from all trees are aggregated by taking a majority vote. For brevity, the details are omitted but the interested reader is referred to [27] for a more comprehensive account.

2.2.6.1 Advantages and limitations

Random forests are able to learn complex and non-linear decision boundaries. They are also very stable with regard to outliers. Another benefit is that no feature scaling is required, since the method uses a rule based approach instead of distance calculations. The method can also provide information about the importance of each feature, which can give insights into the underlying patterns in the data.

A large drawback of random forests in contexts where the computational resources are limited is that the model can take up a lot of memory, depending on the number of trees and their depth.

2.2.7 Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron is a simple feed-forward neural network. It consists of multiple layers of interconnected neurons, including an input layer, one or more hidden layers and an output layer. Each neuron is connected to all neurons in the adjacent layers, forming a fully connected architecture. This structure is visualised in Figure 2.4. The connections between neurons are associated with weights, which are adjusted during the training process. The neurons in the MLP use non-linear activation functions, which allow the network to capture complex non-linear patterns in the data [22].

A MLP network defines a mapping $\mathbf{y} = f_\theta(\mathbf{x}; \theta)$. The network can be represented as a composition of functions, where each layer constitutes a function. Consider a network with layers $l = 1, 2, \ldots, L$. The function for the l-th layer is

$$f_{\theta_l}\left(\mathbf{a}^{(l-1)}\right) = \sigma^{(l)}\left(\mathbf{w}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{2.13}$$

where $\mathbf{a}^{(l-1)}$ is the output from the previous layer (or the input vector $\mathbf{x}$ for the first layer), $\mathbf{w}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases for the l-th layer, and $\sigma^{(l)}(\cdot)$ is the

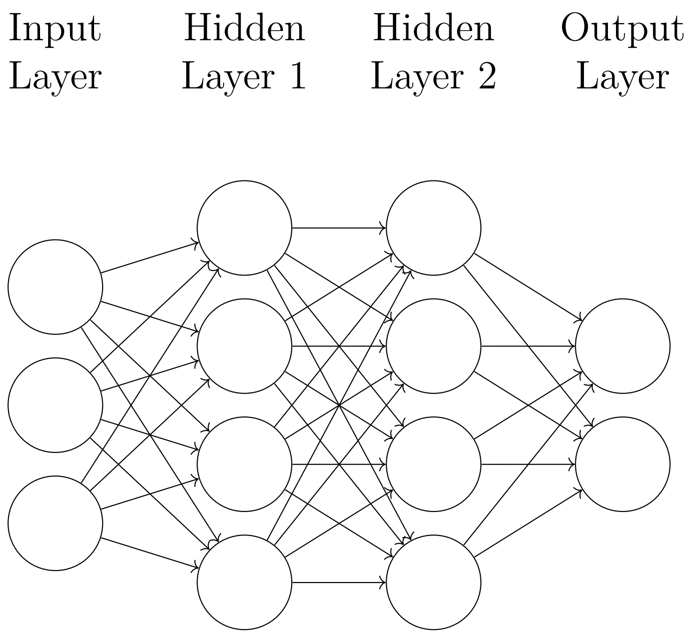


Figure 2.4: Network diagram for fully connected Multi-Layer Perceptron with three dimensional input, two hidden layers and two dimensional output. Each link has an associated weight parameter, and each neuron contains an activation function.[22]

activation function for the l-th layer. The entire function can then be written as

$$f_\theta(\mathbf{x}) = f_{\theta_L} \circ f_{\theta_{L-1}} \circ \cdots \circ f_{\theta_2} \circ f_{\theta_1}(\mathbf{x}). \tag{2.14}$$

For more details, see for instance the book [23].

2.2.7.1 Advantages and limitations

An advantage of MLPs is that they are capable of learning complex, nonlinear relationships in the data. A disadvantage is that the task of finding the optimal weights and biases is a highly non-convex optimization problem. To find good weights and biases the stochastic gradient descent method is utilized. After the model has been trained it offers a low level of interpretability.

2.3 Evaluating classification results

There exists multiple methods for evaluating eye movement event detectors, and which one is the best to use depends on the specific intent of the evaluation. For example, when comparing a large number of event detection algorithms, it would be appropriate to use a concise set of measures that can be clearly presented in a table or plot. Conversely, comparing labels from two human annotators may require a more detailed and qualitative description. Preferably, a set of metrics that are both concise and descriptive should be used. The most common approach to evaluate event detection performance involves comparing predicted labels to ground truth labels, which can be done with two fundamentally different techniques: either on a sample-by-sample basis, or by considering continuous sequences of samples with the same label, known as oculomotor events, for example saccades and fixations [28].

When choosing a suitable metric for evaluating an event detection algorithm, one might initially consider a sample-level evaluation, where the predicted label is compared to the ground truth on a sample-by-sample basis. However, this approach has significant drawbacks, particularly when dealing with unbalanced class distributions, which is often the case in eye-tracking data, where saccades often account for only a small portion of the total samples. Using a metric like sample-level accuracy could misleadingly suggest good performance if a classifier consistently predicts that every sample is a fixation. This issue can be mitigated to some extent by using metrics that account for class imbalance, such as ROC-AUC, F1-score, and G-Mean [29]. Figure 2.5 illustrates a case where sample-level evaluation fails, as all three of these metrics yield the same score despite drastically different perceived levels of performance.

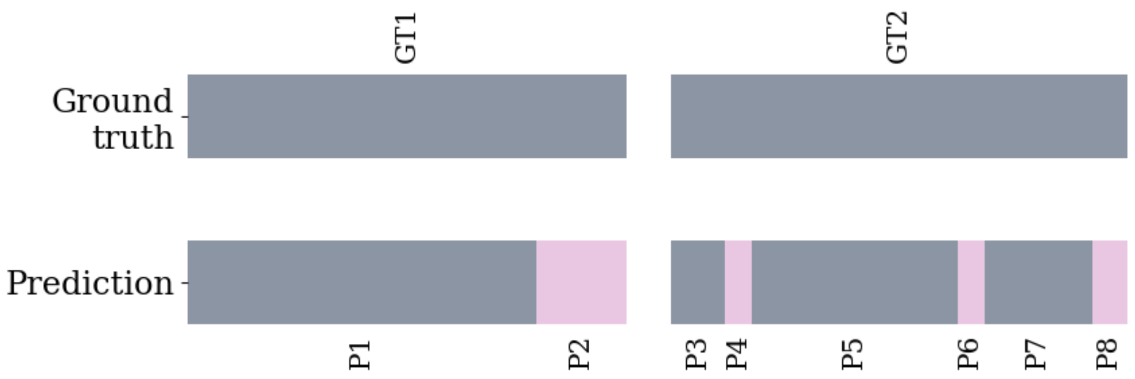


Figure 2.5: The grey areas represent fixations and the pink represent saccades. This figure illustrates two situations where the sample level accuracy would be exactly the same, but the practical performance differs substantially. On the left side there is one false saccade, whereas on the right side there are three. Figure recreated with small adjustments from [28] with permission.

2.3.1 Event matching

The other way to evaluate classification performance is to do it on an event level, where entire events are matched to each other based on some sort of matching algorithm. Then some metric can be applied to evaluate the aggregate level of correctness for those matchings. There are several strategies for matching events (an overview is provided in [28]), and a few will be briefly presented to provide context for the strategy chosen in this thesis.

In 2016, Majority voting was proposed by [30], where performance is evaluated by checking, for each ground truth event, whether over 50 % of the samples in the predicted output have a certain class label, and assigning that class to the event. This is a simple approach, but just like the sample-level method it tends to give inflated scores. Effectively it is the same as a sample-level approach, with the difference being that correctness is decided for each ground truth event rather than for each sample [28]. In the two cases presented in Figure 2.5 the majority voting method gives the same score for both, since one ground truth event would be matched with one event in the prediction sequence, even though the two cases show a large qualitative difference.

Another approach is called the Maximum overlap matching, first introduced in 2019 [11]. For every ground truth event all overlapping events in the prediction are considered, and a match is made with the one that has maximum overlap. Unlike in Majority voting (where each ground truth event is just assigned a predicted label), this approach explicitly matches events, which allows for further evaluation with the metrics in Section 2.3.2. In certain situations however, the Maximum overlap method is prone to make matches that could be interpreted as mistakes. The classes in eye-movement event detection are inherently unbalanced with regard to their duration. Saccades are typically much shorter than fixations. So for a 30 ms saccade a 20 ms overlap with a fixation should intuitively be worth 'less' than a 10 ms overlap with another saccade. In Figure 2.6 there is such a case, where the P8 saccade is matched with the GT5 fixation rather than the (intuitively correct) GT6 saccade.

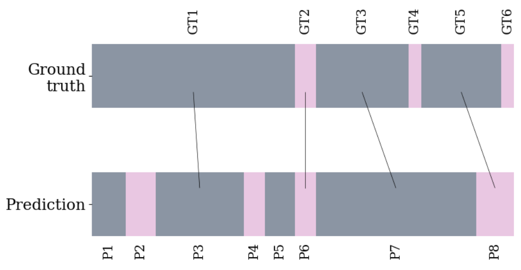


Figure 2.6: Example of the Maximum Overlap matching algorithm. Grey areas are fixations and pink areas are saccades. The black lines between predicted and ground truth events represent event matches. The matching GT5↔P8 is sub-optimal. Figure recreated with adjustments from [28] with permission.

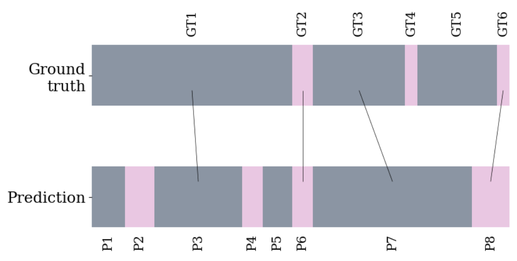


Figure 2.7: Example of the Maximum Intersection-Over-Union matching algorithm. Grey areas are fixations and pink areas are saccades. The black lines show between predicted and ground truth events represent event matches. Figure recreated with adjustments from [28] with permission.

Luckily, there is another matching algorithm, introduced by [31] that handles these cases in a better way. It is called the Maximum intersection-over-union (IoU) matching. For a ground truth event A and a predicted event B it is calculated as

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{|A \cap B|}{|A \cup B|}. \tag{2.15}$$

The IoU values are calculated for all combinations of overlapping events, and then those pairs are placed in a sorted list. The list is then iterated through from pairs with the largest value to the smallest. If neither one of the events in the candidate pair are already matched a match is made, otherwise the candidate pair is skipped.

Since this method divides the overlap with the total combined area it is more likely to match events with similar sizes to each other. In Figure 2.7 the same ground truth and predictions as in Figure 2.6 are shown, but with the matchings made by the Maximum intersection-over-union algorithm rather than the Maximum overlap algorithm.

These matching algorithms do not provide matches for all events. The events in the ground truth in Figure 2.6 and 2.7 without a black line going to them are undetected, and the events in the prediction without a black line going to them are either false positive or false negative events.

Here only three matching algorithms have been discussed. There are other ones, but according to [28] none of them can reach the same levels of usability and versatility as the Maximum intersection-over-union algorithm. Hence, this is the one that will be used in this thesis.

2.3.2 Evaluation metrics

Once matches between the ground truth and prediction sequences have been established the detection power can be quantified through the use of various metrics.

The simplest, yet most informative method is to use a confusion matrix. It not only gives information about the proportion of misclassified events, but also in what manner they have been misclassified. It shows if an event has been misclassified or missed completely. The drawback of using a confusion matrix for evaluation is that it is not particularly concise. If there are N classes it consists of N^2 values. Instead it is often better to use a metric that somehow distills the confusion matrix into a single value. There are many such metrics, and [28] evaluates the following: accuracy, balanced accuracy, precision, sensitivity, specificity, F1-score, Jacard index, Cohen's Kappa, Matthew's Correlation Coefficient (MCC), ROC AUC, and Length normalized Levenshtein distance. They conclude that the Matthew's Correlation Coefficient is the best metric for evaluating the performance of eye movement event detectors, and therefore it is the one which will be used. It is defined as

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.16}$$

where TP is true positive, TN is true negative, FP is false positive and FN is false negative.

The range of the MCC value is $[-1, 1]$, where 1 is the highest score and -1 the lowest. A MCC score of 0 means that the performance is equivalent with random guessing. A negative MCC score implies that the model's predictions are negatively correlated with the actual outcome, so that it performs worse than random guessing. If the MCC score is 1 all predictions are correct, and conversely if they are all incorrect the score is -1.

2.4 Savitsky-Golay filtering

A Savitsky-Golay filter (commonly referred to as a savgol filter) is a well-known type of filter for digital signal processing. It smooths the signal by fitting a low-degree polynomial to groups of adjacent data points with the linear least squares method, and then using the central point of the fitted polynomial as the new value. The key advantage of the Savitsky-Golay filter in comparison with simpler methods such as window based averaging is that it preserves features such as peaks and valleys in the original signal, while smoothing the data, and thereby improving the signal to noise ratio.

Mathematically, the procedure can be described as follows: given a set of datapoints (x_i, y_i), a polynomial $P(x)$ of degree r is fitted to a window of $2m+1$ points centered at x_0:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_r x^r. \tag{2.17}$$

The coefficients $a_0, a_1, \ldots, a_r$ are found by minimizing the least squares error:

$$\min_{P \in \mathcal{P}_r} \sum_{i=-m}^{m} \left(y_i - P(x_i)\right)^2. \tag{2.18}$$

After the polynomial has been fitted the value at the middle of the window is taken as the new smoothed value. This procedure is repeated for all points in the dataset. The filter has a two main parameters that have to be chosen. The first is the selection of the window size $2m + 1$, where a larger window size results in better noise reduction at the risk of losing important details in the signal. The second is the order r of the fitted polynomial, where a higher value allows for the capturing of more complex signal shapes, but also increases the risk of overfitting.

Simultaneous differentiation and filtering of the signal is also possible, simply by differentiating the polynomial $P(x)$ the desired amount of times after it has been fitted, hence for the first derivative we have

$$P'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \cdots + r a_r x^{r-1}. \tag{2.19}$$

Just as in the undifferentiated case the smoothed value is found by evaluating the polynomial at the center of the window [32].

2.5 Two parameter noise analysis

In 2020 a method to characterize noise in eye-tracking data was proposed, utilizing two parameters [33]. One parameter describes the magnitude of the noise and the other one quantifies its shape or behaviour. They are both based on standard deviation and sample-to-sample root mean square values, which in turn are defined as:

$$\mathrm{STD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left((x_i - \bar{x})^2 + (y_i - \bar{y})^2\right)} = \sqrt{\mathrm{STD}_x^2 + \mathrm{STD}_y^2} \tag{2.20}$$

and

$$\text{RMS-S2S} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}\left((x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2\right)} \tag{2.21}$$

respectively. The RMS-S2S provides insight into the step size of the signal between consecutive samples, and the STD value indicates the overall variability or dispersion of the signals value around the mean. These two metrics can be combined to form a new one that weighs both aspects of noise magnitude equally, which will be referred to as signal magnitude;

$$\text{signal magnitude} = \sqrt{\text{RMS-S2S}^2 + \text{STD}^2}. \tag{2.22}$$

The parameter that describes the shape of the noise is called signal type, and is defined as

$$\text{signal type} = \frac{\text{RMS-S2S}}{\text{STD}}. \tag{2.23}$$

It can be thought of as the ratio between displacement of successive samples and the dispersion or spatial extent of the segment that includes those samples. When this ratio is high it means that the distance between successive gaze positions covers a substantial portion of the segments spatial range, which is typical for high frequency noise. In contrast, when the ratio is low the gaze position changes smoothly, which is characteristic of low frequency noise. The quantity is dimensionless, since it is a ratio between two spatial measures with the same units.

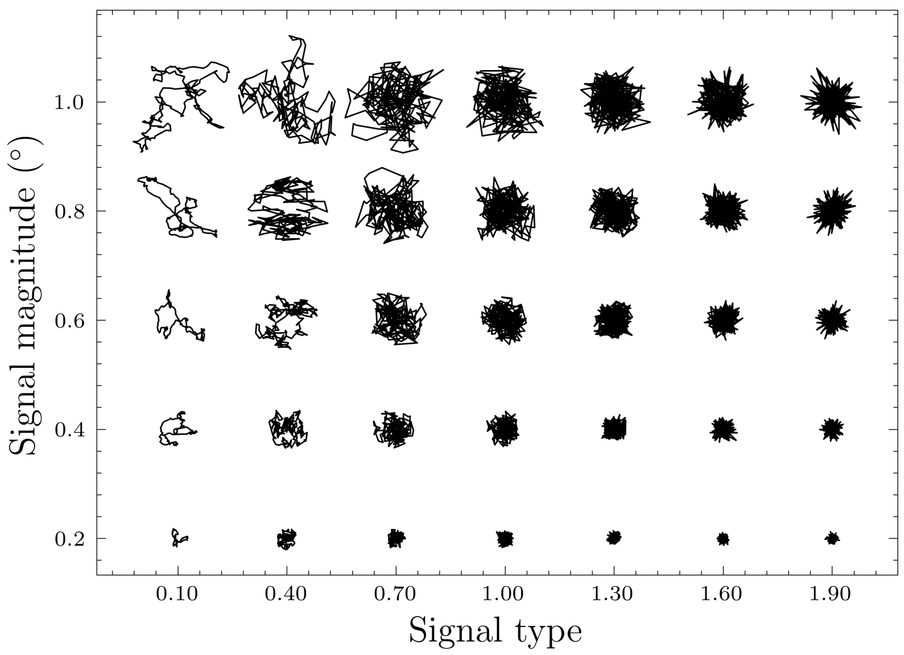


Figure 2.8: Generated two dimensional signals with different types and magnitudes of noise. On the left side are sequences with smooth, low frequency noise, and on the right side are sequences with rugged, high frequency noise. Each example is 200 samples long. The magnitude of the noise increases along the vertical axis. The figure is recreated from [33].

Figure 2.8 shows some examples of two dimensional noise with different signal types and magnitude. These two parameters can be used both to describe the properties of noise signals, but also to generate new noise that can be applied to synthesized signals, or as augmentation for existing ones. The interested reader is referred to [33] for a detailed account of how any two dimensional noise can be generated given the desired type and magnitude values.

3
Methods

This chapter begins with an overview of the datasets that have been used in the thesis, and a description of the process that was undertaken to record a new dataset for this thesis. Then an account of the feature extraction procedure is given. The final section delves into the model training, inference, post processing, evaluation and hyperparameter optimization.

3.1 Overview of datasets

Three different datasets were considered: one collected specifically for this thesis, one publicly available dataset from a previous study, and one that was synthetically generated. Validation and testing were conducted on the self-collected data.

3.1.1 Self-collected dataset

To verify that the created prediction models perform well on the type of data where the models eventually will be used, a rather significant amount of training, validation and test data was collected from 20 volunteers, using the open source software PsychoPy [34] and a two camera Smart Eye Pro eye-tracking system. In the coming chapter this dataset will be referred to by the name SER.

3.1.1.1 Physical setup

An experimental setup was built up in a room. It consisted of a projector, a table, two Smart Eye Pro cameras and an adjustable chair. The *Infocus IN118HDxc* projector created a 215 by 105 centimeter screen with a 1920×1080 resolution on the wall. As can be seen in Figure 3.1 it was mounted in the ceiling of the room, to allow it to project over the top of the seated test subject. The walls in the room were a bright grey color and all light sources in the room other than the 3200 lm projector were turned off during recordings. The participant was seated in a chair that could be adjusted backwards and forwards to make it possible to place the face of every subject in the focus range of the cameras.

3.1.1.2 Procedure

To find subjects to record employees at Smart Eye were asked to participate. We attempted to find subjects with a good variation of factors such as genders and

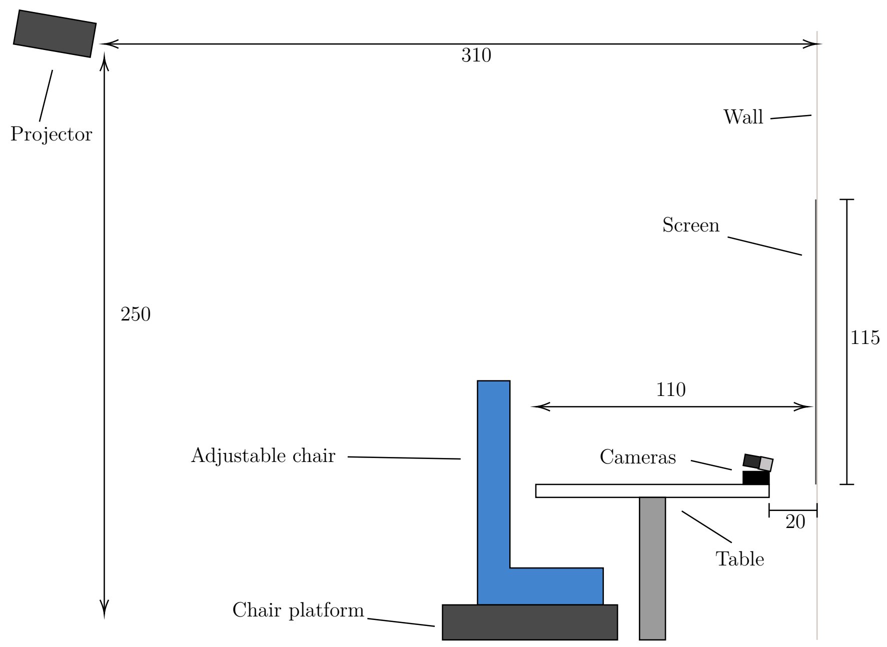


Figure 3.1: The recording setup viewed from the side. The length units are in centimeters and the schematic is roughly to scale. To achieve a rectangular image despite the projector being tilted at an angle, the vertical keystoning feature of the projector was used.

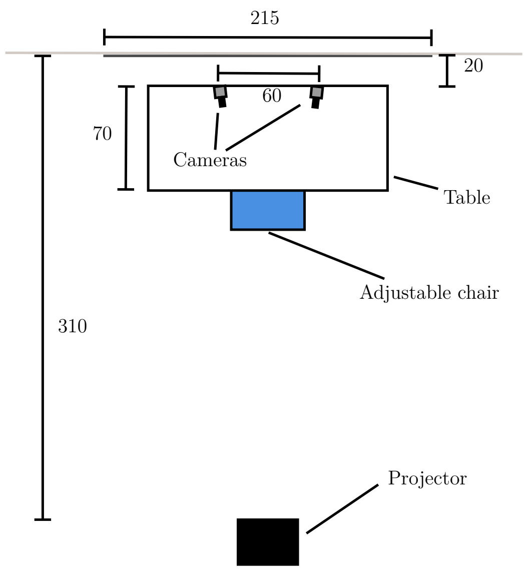


Figure 3.2: Top down view of the experiment setup. The length units are in centimeters and the schematic is roughly to scale.

ethnicity, but since the employees predominantly are European males the pool of subjects is slightly skewed, which can be seen in Table 3.1.

After a subject had been recruited I explained that their task was to follow a target with their gaze for approximately one minute. This was done twice, once with their head fixated with a chin rest (as in Figure 3.6) and once where they could move their head freely. Random counterbalancing was performed by flipping a coin to determine the order of the recordings with and without the chin rest.

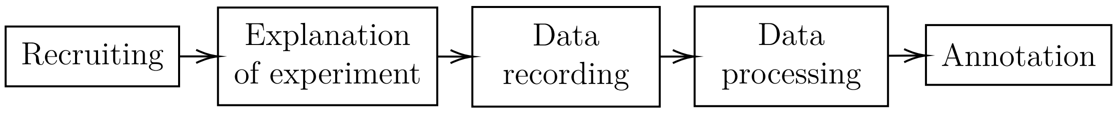


Figure 3.3: Diagram showing the overall experiment procedure.

The participants were tasked with following a viewing target displayed on the wall with their gaze. PsychoPy was used to create the viewing target and control its' movement. The viewing target itself, which can be seen in Figure 3.5, consisted of a red dot at the center of a circle with a cross inscribed. It has been proven that such a shape is optimal for increasing fixational stability [35]. This target jumped around to 30 random positions on the screen with a minimum gaze angle between consecutive points set to 2°. The target was displayed at each point for a randomly sampled duration between one and three seconds. A constant duration was avoided because it might have allowed the subjects to anticipate when the target would jump, potentially influencing their saccades. Ten consecutive positions were sampled from positions covering just the center 25 % of the screen whereas the other twenty were chosen from positions on the entire screen. This was done to increase the number of short saccades in the collected dataset. Which one of those two groups of stimuli that was displayed first was randomised for each trial, to rule out any potential impact from the ordering.

Each subject underwent the recording procedure twice. Once with their head fixated with a chin rest and once where they were able to move their head freely. These are known as lizard and owl gazes respectively [36]. This was done to allow for more variation in the dataset, and to provide both simpler and more difficult cases to test

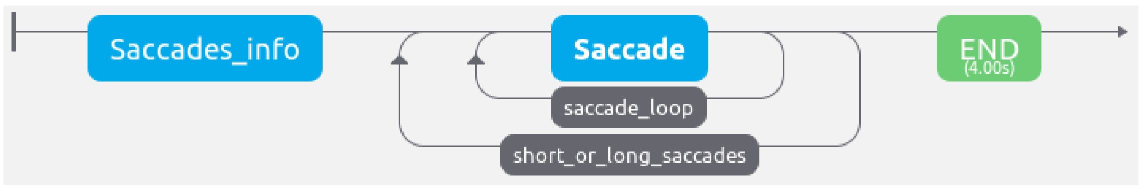


Figure 3.4: The experiment layout as shown in the PsychoPy Experiment Builder. Time moves from left to right in the chart. First an information screen (represented by the Saccades_info block) is displayed which says what the task is, and asks the participant to give a verbal cue once they are ready to start. Then the part where the saccades occur and data is recorded takes place (in the Saccade block). Finally an end screen is displayed, thanking the participant for their time (represented by the block labeled END).

Category	**Count**	**Percentage**
Gender		
Male	15	75 %
Female	5	25 %
Glasses wearer		
Yes	5	25 %
No	15	75 %
Age Range		
20-25	1	5 %
26-30	6	30 %
31-35	5	25 %
36-40	2	10 %
41-45	4	20 %
46-50	2	10 %

Table 3.1: Aggregated characteristics of subjects in the study.

and validate the algorithms on. The eye-tracking generally works better when the head is stationary.

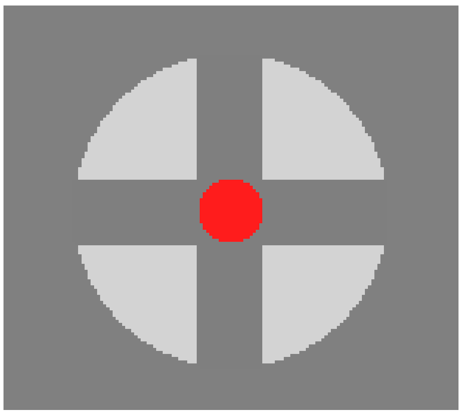


Figure 3.5: The target that the participants had to focus their gaze on. On the screen it appeared with a total diameter of 10 cm. The diameter of the red circle in the middle was 2.5 cm.

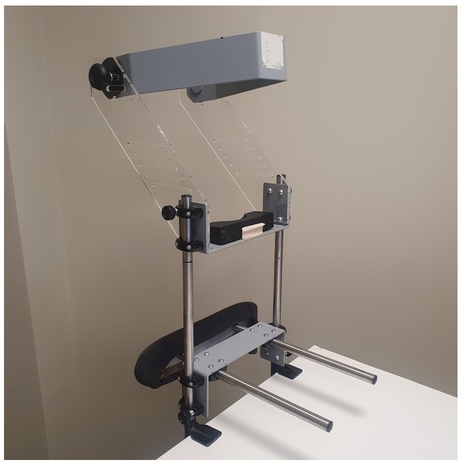


Figure 3.6: The chin rest that was used to fixate the heads for one recording of each participant.

3.1.1.3 Data annotation

The data was annotated using a tool that enabled simultaneous viewing of the recorded video and the eye-tracking data. The eye-tracking data was presented to the annotator in a normalized format, where the range of the gaze position signals spanned from -1 to 1. The video was used to mark the start and endpoints of events, while the gaze and heading signals from the eye-tracking data served as guides to identify roughly where saccades occured in the video. Some saccades, not visible in the signal view, were detectable in the video feed and were therefore annotated. The graphical user interface of the annotation tool is displayed in Figure 3.7. It took circa 45 minutes to annotate one minute of data, so for each recorded subject 1,5 hours were devoted to annotation. In total the annotation time amounted to 30 h.

In some rare cases it was not possible to see a saccade in the video where there was supposed to have been one. That led to some fixations being labeled as 5 s to 6 s, which is unreasonable considering that the fixation target moved at least every third second. The likely cause for those cases is that the gaze target moved a short distance, and the focus of the cameras might have been slightly poorly set.

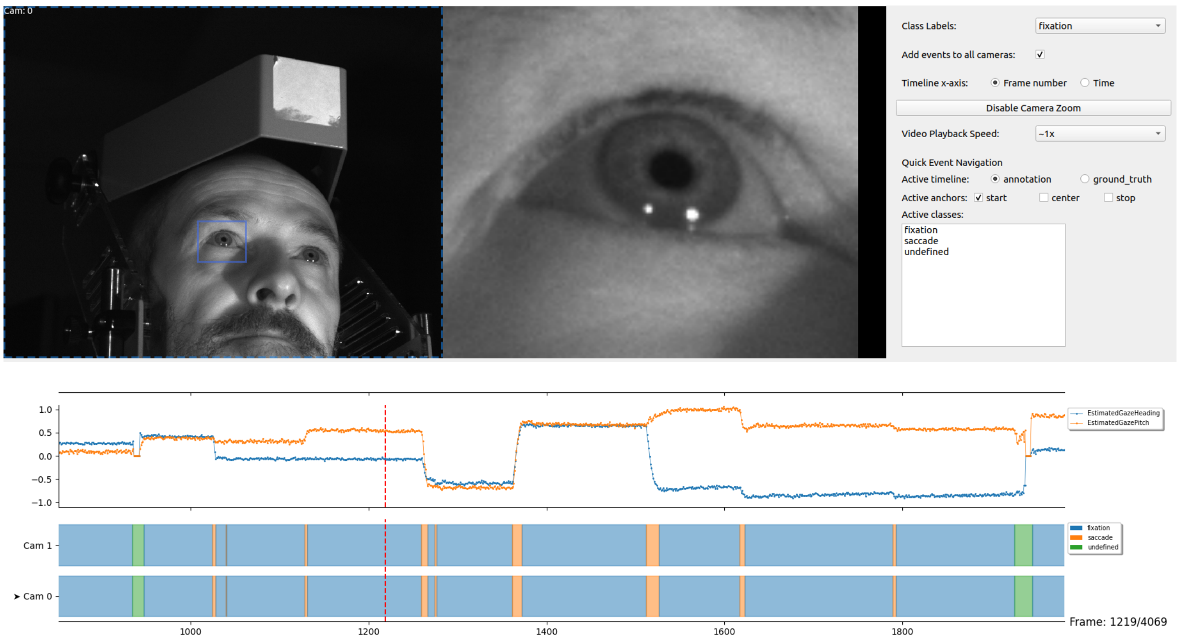

Class Labels: fixation
Add events to all cameras: [✓]
Timeline x-axis: [✓ Frame number] Time
Video Playback Speed: ~1x
Quick Event Navigation
Active timeline: [✓ annotation] ground_truth
Active anchors: [✓ start] center stop
Active classes: fixation, saccade, undefined
Frame: 1219/4069

Figure 3.7: The graphical user interface of the event annotation tool, that allows for annotation of events in the data. In the bottom section the labels can be seen, and in the middle are normalised plots of the heading and pitch of the gaze. In the upper section there is a video feed of the participant and a zoomed in view of one of their eyes.

3.1.1.4 Data splitting

The recorded data was split into three groups, test data, train data and validation data as described in Table 3.2. The training set was used for training and data exploration, the validation set was used for tuning of hyperparameters, and the test dataset was solely used to test the models. Splitting was done on a subject level, so both recordings from each subject were placed in the same split. If that had not been the case recordings from the same subject could have ended up in both the training and test sets, which would severely limit what conclusions could be drawn from the results. With this approach the algorithm can be tested on completely unseen data, i.e. data from a different subject.

As can be seen in Table 3.2 the largest split is the test split, at about 50 % of the recorded data. In other studies a split of 80 / 20 % is commonly used [37]. Here a larger portion of test data was selected because other datasets will also be used for training, so in the end the size of the test set will be reasonable.

Data Split	**Time (minutes)**	**Percentage of Total Time**	**# Saccades**
Test	22.7	49.8%	927
Train	11.7	25.6%	345
Validation	11.2	24.5%	421
Total	**45.6**	**100%**	**1693**

Table 3.2: Amount of recorded data in each split. Synthetic data and data from other sources is also used in the thesis, so the distribution in this table is not representative of the overall distribution.

3.1.2 Synthetic data

Synthetic data was generated by iteratively producing an alternating sequence of fixations and saccades. In the coming chapters this dataset will be referred to by the name SYN. Let $z \sim P$ denote a random variable with distribution P, and let $\mathrm{Unif}(a, b)$ denote the uniform distribution over the integers $\{a, a+1, ..., b\}$. Additionally, let $\mathbf{x}$ and $\mathbf{y}$ be vectors with cardinality C where the generated heading and pitch values will be stored, and let the index i denote the current position in the vectors $\mathbf{x}$ and $\mathbf{y}$ where new values will be inserted. Given a sampling frequency f and a target maximum gaze angle α, the following steps were performed:

1. **Iterate** over the steps (a) and (b) until the end of $\mathbf{x}$ and $\mathbf{y}$ has been reached:
   (a) Sample a **fixation** duration, $\tau_{\text{fix}} \sim \mathrm{Unif}\left(\lfloor a \cdot f \rfloor, \lfloor b \cdot f \rfloor\right)$, corresponding to a fixation duration between a and b seconds. Add τ_{fix} to i: $i \leftarrow i + \tau_{\text{fix}}$. Fill the entries of $\mathbf{x}$ and $\mathbf{y}$ with the last populated values until the index i is reached.

   (b) Sample two **saccade** durations, one for pitch and one for heading,

$$\begin{aligned}\tau_{\text{sacc}_x} &\sim \mathrm{Unif}\left(\lfloor c \cdot f \rfloor, \lfloor d \cdot f \rfloor\right)\\ \tau_{\text{sacc}_y} &\sim \mathrm{Unif}\left(\lfloor c \cdot f \rfloor, \lfloor d \cdot f \rfloor\right),\end{aligned}$$

   corresponding to saccade durations between c and d seconds. The saccade amplitudes s_x and s_y are calculated as $s_x = \frac{\tau_{\text{sacc}_x} \cdot v}{f}$ and $s_y = \frac{\tau_{\text{sacc}_y} \cdot v}{f}$, where v represents the average velocity of the saccade in degrees per second.

   Let $\delta_x, \delta_y \in \{+1, -1\}$ be direction variables.

   - Let x be the last populated value in $\mathbf{x}$. If $x \notin [-\alpha, \alpha]$, choose the direction of δ_x to bring x closer to the allowed range.
   - Otherwise, if $x \in [-\alpha, \alpha]$, flip the sign of δ_x with a probability of 0.5.

   Do the same for $\mathbf{y}$ and δ_y.

Insert the saccadic movements in x- and y-directions into $\mathbf{x}$ and $\mathbf{y}$ respectively using scaling by the function $g(t) = \sigma(5(2t-1))$, where σ is the sigmoid function. The updated $\mathbf{x}$ and $\mathbf{y}$ vectors are expressed as:

$$\mathbf{x} = \left(x_1, ..., x_i, x_i + \delta_x \cdot g\left(\frac{1}{\tau_{\text{sacc}_\text{x}}}\right) s_x, \ldots, x + \delta_x \cdot g\left(\frac{\tau_{\text{sacc}_\text{x}}}{\tau_{\text{sacc}_\text{x}}}\right) s_x, *, *, \ldots\right)$$
$$\mathbf{y} = \left(y_1, ..., y_i, y_i + \delta_y \cdot g\left(\frac{1}{\tau_{\text{sacc}_\text{y}}}\right) s_y, \ldots, y + \delta_y \cdot g\left(\frac{\tau_{\text{sacc}_\text{y}}}{\tau_{\text{sacc}_\text{y}}}\right) s_y, *, *, \ldots\right)$$

where $*$ denotes undefined values that will be populated later. The function $g(t)$ has the properties $g(0) \approx 0$ and $g(1) \approx 1$.

2. **Add noise** to the vectors $\mathbf{x}$ and $\mathbf{y}$ after completing the iterations. The noise was generated as described in [33], with a noise type value of 1.24 and magnitude of 1.5°. The definitions of noise type and magnitude are provided in Section 2.5.

The values for α and f used were 35° and 60 Hz respectively. The fixation durations went from $a = 1\,\text{s}$ to $b = 3\,\text{s}$, and the saccade durations ranged from $c = 16\,\text{ms}$ to $d = 160\,\text{ms}$.

The saccades were modelled as sigmoid functions, which corresponds well with how the human eye actually rotates during a saccade, with an acceleration phase, a middle phase with constant velocity and finally a deceleration phase [38].

To find a suitable value for the average saccade velocity, v, two different sources were considered, and the recorded dataset was analyzed. In 1988 a relation between saccade amplitude in degrees and duration in milliseconds was proposed:

$$\text{duration} = 2.2 \cdot \text{amplitude} + 21 \text{ [39].} \tag{3.1}$$

Later in the same year another study considered centrifugal saccades (where the eye rotates away from a neutral position) and centripetal saccades (where the eye rotates toward a neutral position) separately [40]. They reported the following relationships:

$$\text{centripetal duration} = 2.5 \cdot \text{amplitude} + 27 \tag{3.2}$$
$$\text{centrifugal duration} = 3.9 \cdot \text{amplitude} + 13. \tag{3.3}$$

which both are slightly slower than what was initially proposed. In Figure 3.8 the amplitudes and durations of saccades in the self-collected dataset are presented. A least squares fit to those data points without offset is also included, as well as the previously mentioned relationships. There is a large spread in the recorded data, which none of the plotted lines can capture. All of the plotted lines are quite similar, however in the end it was decided to use the least squares fit as the basis for the synthetic saccade average velocity, primarily because it was desired for the synthetic data to be as similar as possible to the recorded data that will be used for testing and validation. It amounts to the relationship duration(ms) = $4.02 \cdot$ amplitude(°), which is the same as 249 °/s.

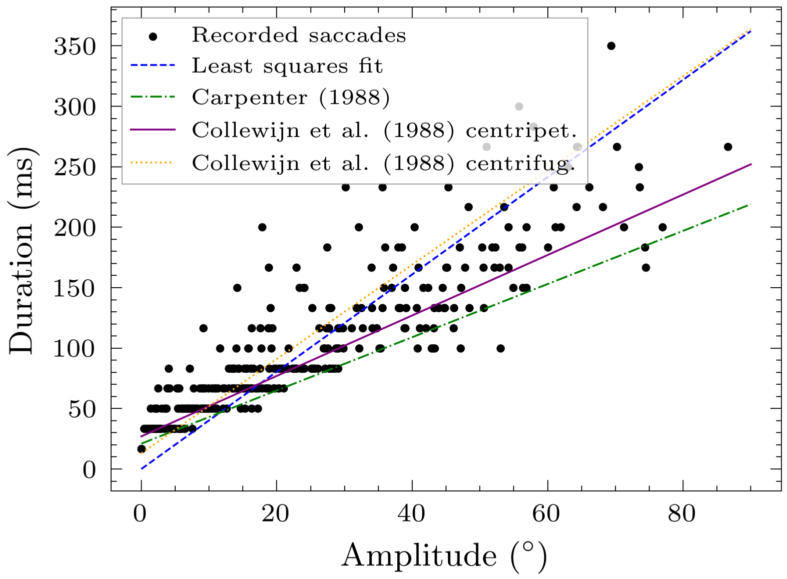


Figure 3.8: The saccade amplitude plotted against the saccade duration for the dataset that was recorded with the Smart Eye Pro system. The blue dashed line is a least squares fit of the data points, which gives a relation of 4.02 ms/°, and the green dash-dotted line is the relation proposed by [39], which corresponds to 2.2 ms/° + 21 ms. The solid purple and dotted green lines were both proposed by [40] and their formulas are displayed in equations 3.2 and 3.3.

3.1.3 IRF dataset

The IRF dataset is a publicly available dataset [41] and consists of manually annotated eye-tracking data. It was collected in 2018 for the purposes of a paper investigating the usage of random forest classifiers for event detection in eye-tracking data [10]. The five participants were asked to track a 0.2° silver dot with a black 2×2 pixel center that jumped around in a 7×7 grid with equal spacing. PsychoPy was used to control this viewing target, and it stayed at each position for 1 second before it jumped to another.

The data was recorded monocularly at 1000 Hz using the Eyelink 1000 eye tracker. This type of tracker has a very small amount of noise. The data is 8.1 min long and contains 847 saccades. It consists of recordings of six different subjects, each being one minute and twenty seconds long. In this study the entire dataset was purely used for training. Since the noise level was so low the dataset was augmented with noise to give it greater resemblance to the self-collected validation and testing data. The added noise had signal magnitude of 1.5° and signal type of 1.24, as defined in Section 2.5.

3.2 Feature extraction

The previously mentioned datasets all contain gaze pitch and heading data. To make the events easier for the models to detect, twelve features have been extracted from the raw data. Table 3.3 contains a list of all features and a description of how they were calculated. All of the features are window-based, which means that for every sample, the feature value is calculated based on the surrounding samples. Three different window widths were set, one for the velocity and acceleration, one for the average acceleration and one for all other features. These window widths were considered hyper-parameters that were tuned specifically for each model type, and can be viewed in Table B.2. Previous works that have relied on window based features have used fixed window widths based on expert knowledge of the time-scales of the eye movement events [6][10]. It is conceivable that the tuned window widths in this study might contribute to a small performance gain.

Since the datasets had different sampling rates, the window widths were expressed in terms of time rather than as the number of samples. Further, when calculating velocity and acceleration, those values were converted to °/s and $°/s^2$ respectively, to remove the effect of varying sampling rates on the feature values.

3.3 Model pipeline

The models were trained using the package Scikit-learn [42]. The velocity and acceleration features were calculated using the savgol_filter function from the scipy package [43]. The other features were calculated using the the rolling window function from the Pandas package [44] in the manners described in Table 3.3.

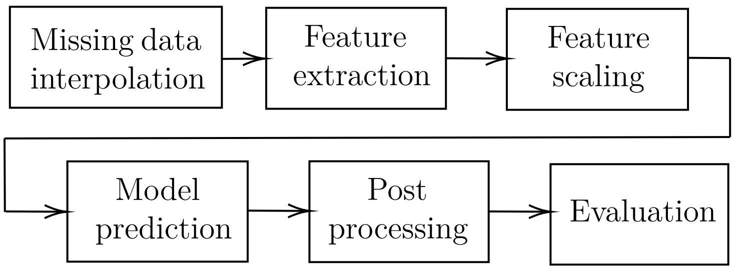


Figure 3.9: Inference pipeline that was ran on validation and testing data. Scaling of the features was not applied for the random forest model.

Figure 3.9 shows a schematic view of the model pipeline. Below some further details will be provided. The validation and testing dataset consisted of 25 % and 50 % of the SER dataset respectively. For training all possible combinations of the SYN, IRF and the remaining 25 % of the SER data was used. Linear interpolation was used to fill in missing values in the data, primarily caused by tracking loss due to blinks. That is not optimal since the real data never is linear, but this issue is mitigated by the fact that only a very small amount of the data is missing. The features were extracted in the manner described above, and then they were scaled to zero mean and unit variance using the scikit-learn `StandardScaler` class. No scaling was performed when random forest models were used. During training the features and corresponding labels were fed to the models with the scikit-learn `fit(feature, label)` function, which uses different loss functions depending on what model is used.

Table 3.3: Extracted features from raw data. All of them except for the avg-acc feature come from [10], with some implementation changes. The `rms` feature in particular deviates significantly, which is discussed in Section 5.2.

Feature	Description	Formula/Method	Units
`vel`	*velocity.* Rate of change of position	Computed using Savitzky-Golay filter (polynomial order 3)	°/s
`acc`	*acceleration* Rate of change of velocity	Computed using Savitzky-Golay filter (polynomial order 3)	$°/s^2$
`avg-acc`	*Mean acceleration*	Moving average of the acc feature	$°/s^2$
`std`	*Standard deviation* of the recorded gaze position	Hypotenuse of the std. dev. in x and y-direction	°
`mean-diff`	*Difference in mean gaze position.* Measures the change in the average gaze position between two windows	Computed as the hypotenuse of the differences between the mean x and y positions in the window before and after the current sample	°
`med-diff`	*Difference in median gaze position.* Measures the change in the median gaze position between two windows	Computed as the hypotenuse of the differences between the median x and y positions in the window before and after the current sample	°
`bcea`	*bivariate contour ellipse area*	Measures the area of an ellipse in which the gaze resides P% of the time during a window. P=68.	$°^2$
`rms`	*root mean square*	The hypotenuse of the root mean square values of the pitch and heading. The rms in one dimension of a window of lentgh n is $\sqrt{\frac{1}{n}\left(x_1{}^2 + x_2{}^2 + \cdots + x_n{}^2\right)}$	°
`std-diff`, `rms-diff`, `bcea-diff`	Absolute difference in: *standard deviation*, *root mean square* and *bivariate contour ellipse area*	These are similar to mean-diff and med-diff in the sense that they use windows before and after the current sample, but instead of differences in gaze position these are differences in measures of noise	°
`disp`	*dispersion* Sum of the range in x values and the range in y values	$\text{disp} = [\max \mathbf{x} - \min \mathbf{x}] + [\max \mathbf{y} - \min \mathbf{y}]$	°

3.3.1 Post processing

After predictions into the classes saccades and fixations had been made for each sample, post processing was applied. It consisted of two different actions, which both are based on knowledge about typical properties of eye movement events. Saccades with an amplitude of less than 1° were converted to fixations. Fixations shorter than 60 ms (corresponds to three samples in validation and testing data since it was recorded at 60 Hz) were labeled as saccades. These values are recommended by [45].

3.3.2 Evaluation

The post-processed predictions were evaluated using the intersection over union matching algorithm with the Matthew's correlation coefficient as described in Section 2.3.2 to quantify the accuracy and reliability of the detected eye movement events.

3.3.3 Hyperparameter optimization

An optimal set of hyperparameters was determined using the Optuna framework [46]. Some of the hyperparameters were related to the feature extraction from the data, for example window sizes and the width of the Savitsky-Golay filter. These were set individually for each model and combination of datasets, to make sure that the features were optimally engineered for each model and dataset combination. This approach ensured that the feature extraction process was tailored to the specific characteristics of each combination of datasets. Other hyperparameters were model specific, like the number of trees and their depth in the random forest model. In Table B.2 the hyperparameters related to the feature extraction are listed, and in Table B.1 the model specific hyperparameters for the best performing model of each type can be found. Hyperparamters not listed in that table were set to the default values in scikit-learn version 1.2.2.

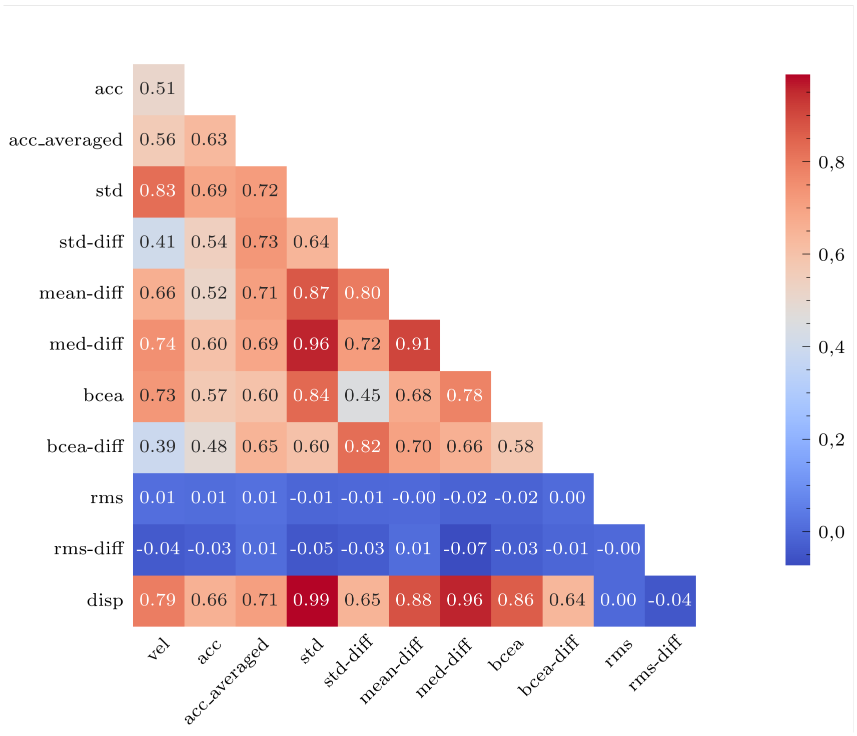
4 Results

This chapter begins with an overview of the extracted features, which is followed by a comparison between synthetic and real data. Next, the results of the hyperparameter tuning and model evaluation are presented, including the evaluation of the different model types trained on all possible combinations of datasets included in the study. Then the two best-performing models are analyzed further, and factors that could potentially explain the difference in performance between test subjects are investigated. This is accompanied by an exploration into how well a set of derived metrics line up with the ground truth. Finally, a comparison with the algorithm included with the Smart Eye Pro eye-tracking system is made.

4.1 Analysis of features

It is reasonable to expect that the features are either correlated or anticorrelated, since they have been engineered to give a response when saccades take place. In Figure 4.1 we see that the features are all fairly correlated with each other, and since the absolute value has been applied to all features the correlation matrix contains mostly positive values. The correlation is calculated as the Pearson correlation coefficient. The values for the features `rms` and `rms-diff` deviate from all others. This is likely due to a mistake in the implementation of the feature calculation or processing by the author, which is discussed further in Section 5.2, since these features appear uncorrelated with the `std` and `std-diff` features in particular. Considering the fact that both these pairs of features are measures of noise, one would expect them to be correlated. Since they are not, the `rms` and `rms-diff` features will be disregarded in the further analysis.

The standard deviation and dispersion features have a particularly high correlation of 0.99. This is logical given that they are both measures of variability in the data, though they assess it in different ways. The dispersion feature is a measure of the range between the smallest and largest values in the window, and the standard deviation provides a measure of the how far the values deviate from the mean.



	vel	acc	acc_averaged	std	std-diff	mean-diff	med-diff	bcea	bcea-diff	rms	rms-diff
acc	0.51										
acc_averaged	0.56	0.63									
std	0.83	0.69	0.72								
std-diff	0.41	0.54	0.73	0.64							
mean-diff	0.66	0.52	0.71	0.87	0.80						
med-diff	0.74	0.60	0.69	0.96	0.72	0.91					
bcea	0.73	0.57	0.60	0.84	0.45	0.68	0.78				
bcea-diff	0.39	0.48	0.65	0.60	0.82	0.70	0.66	0.58			
rms	0.01	0.01	0.01	-0.01	-0.01	-0.00	-0.02	-0.02	0.00		
rms-diff	-0.04	-0.03	0.01	-0.05	-0.03	0.01	-0.07	-0.03	-0.01	-0.00	
disp	0.79	0.66	0.71	0.99	0.65	0.88	0.96	0.86	0.64	0.00	-0.04

Figure 4.1: Correlation of the hand crafted features. In general there is a strong correlation between the features. The values shown are the Pearson correlation coefficient and the features are calculated solely from the SER dataset.

Two other features with a high correlation coefficient are the `mean-diff` and `med-diff` features. They both give insight into how much the center position of the gaze has shifted from a window before the sample point to a window after.

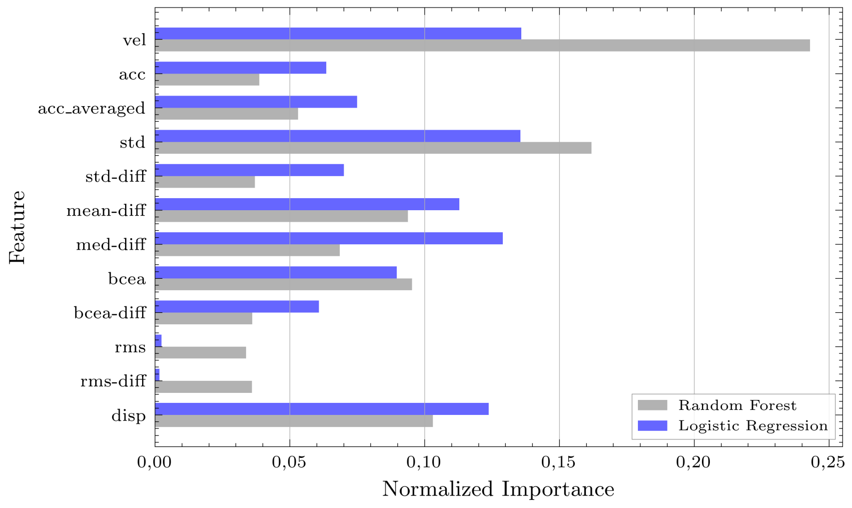


Figure 4.2: The importance of each feature, calculated as the Gini impurity for the random forest model and as the absolute value of the coefficients for the logistic regression model (trained using L2-regularization). Both are normalized such that the sum of all importances is 1.

The importance of the features in the random forest and logistic regression classifiers has been calculated to provide insight into which features carry the most relevance. These results are presented in Figure 4.2. In Figure 4.3 and 4.4 there are examples of a feature that carries a lot of importance and one that is slightly less important, respectively. At first glance it appears reasonable to assume that acceleration would be a good way to detect saccades, as they involve changes in gaze position and acceleration measures the rate of such changes. However, there are two issues with the acceleration feature. During very long saccades the gaze direction moves at a constant velocity for some time (this can be inferred from the fact that 35° and 80° saccades have the same peak velocity in Figure 4.11). As a result, the acceleration feature approaches zero both during fixations and during the middle of very long saccades, so it has reduced predictive power for distinguishing between these event types. Moreover, it is very sensitive to noise since the already noisy gaze position signal has been differentiated twice, amplifying the effect of the noise. That means that very small saccades can get confused with noise. These two issues collectively explain the large overlap between fixations and saccades in Figure 4.4 and the low importance of acceleration in Figure 4.2.

The standard deviation in Figure 4.3 on the other hand has a large separation between values for fixations and saccades. This might be because it is a measure of the variability in the data, and the variability is much higher during saccades than during fixations. Since fixations involve relatively stable gaze, the standard deviation remains low, while saccades involve rapid shifts in gaze position, leading to greater variability. This clear separation explains why the standard deviation feature is assigned higher importance in Figure 4.3, as it effectively distinguishes between the two states, unlike acceleration, which shows overlap between fixations and saccades.

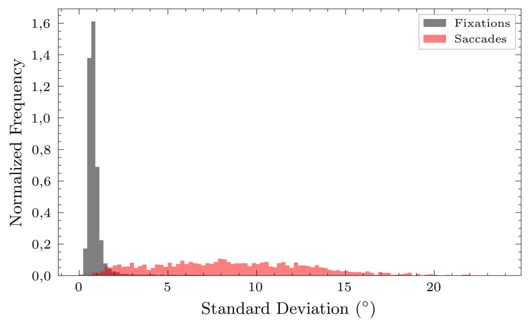


Figure 4.3: The distribution of standard deviation values for saccades and fixations. The majority of the overlap lies between 1 and 2 degrees. In general there is quite a good separation between fixations and saccades in terms of this feature.

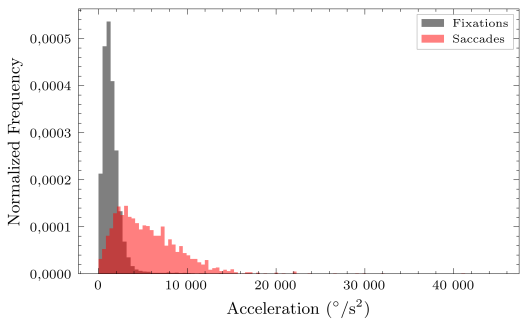


Figure 4.4: The distribution of acceleration values for saccades and fixations. There is a significant region with overlap, between 0 and 4000 °/s^2.

4.2 Synthetic data

Qualitative and quantitative aspects of how the synthetic data relates to the real data will be discussed here. The impact of training on synthetic data, or a combination of real and synthetic data, in relation to model performance is explored in Section 4.3.

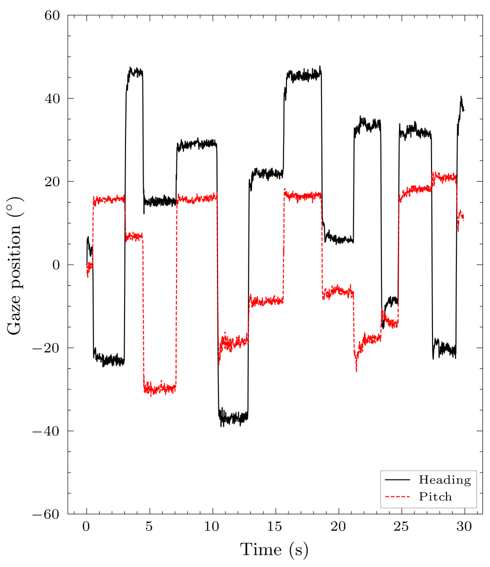


Figure 4.5: Thirty seconds of randomly selected gaze data from the real SER dataset.

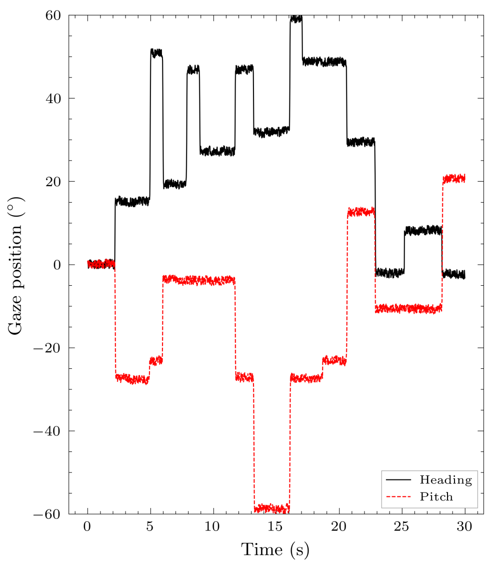


Figure 4.6: Thirty randomly selected seconds of generated synthetic data.

The synthetic data in Figure 4.6 appears somewhat similar to the real data in 4.5. The largest apparent difference is that the synthetic data has a wider field of view, especially along the pitch axis. It also appears that purely horizontal and vertical movements are more common in the synthetic data, whereas movements in the real data tend to be more diagonal.

In Table 4.1 some properties of the synthetic and real data are quantified. They are quite similar, however the average saccade peak velocity is higher for the synthetic data.

Table 4.1: Comparison of eye-tracking metrics for real and synthetic data.

Metric	**Mean**		**Median**	
Dataset	SER	SYN	SER	SYN
Saccade Amplitude (°)	20.49	19.77	13.21	15.23
Sac. Peak Velocity (°/s)	313.74	413.84	302.89	433.62
Saccade Duration	61.93	61.91	49.83	33.33
Fixation Duration (ms)	1369.83	1903.26	1413.65	1866.67

To give some further insight into the distributions of saccade amplitude, peak velocity and duration as well as fixation duration, histograms are presented in Figures 4.7 – 4.10. Some details worth taking note of is that the maximum saccade amplitude only reaches 54° in the synthetic data, whereas it goes above 80° in the real dataset. Since the synthetic saccades do not reach the same amplitudes they also do not reach the same durations as the real saccades. The distribution of peak velocities in Figure 4.8 is slightly skewed toward saccades generally being too fast in the synthetic data, however the maximum values are similar. The synthetic data contains fixations with durations uniformly distributed between 1 s to 3 s. That is the same range as in the stimulus that was displayed to test subjects, so an elevation can be seen between those values in the real data as well in Figure 4.10. Although the real dataset also contains numerous short fixations that occurred in conjunction with small corrective saccades, those are completely lacking in the synthetic data.

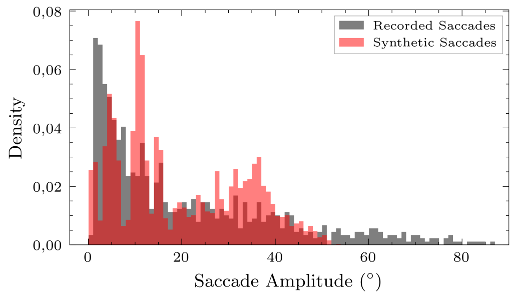


Figure 4.7: Histogram of amplitudes for saccades in the SER and SYN datasets.

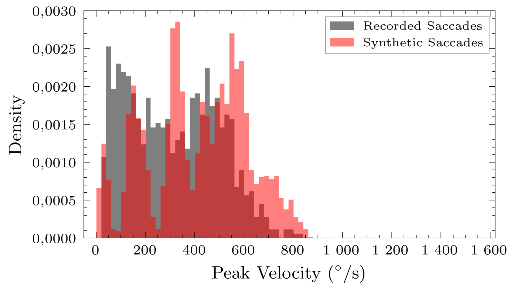


Figure 4.8: Histogram of peak velocities for saccades in the SER and SYN datasets.

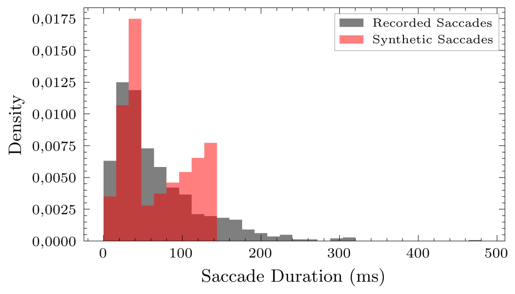


Figure 4.9: Histogram of durations for saccades in the SER and SYN datasets.

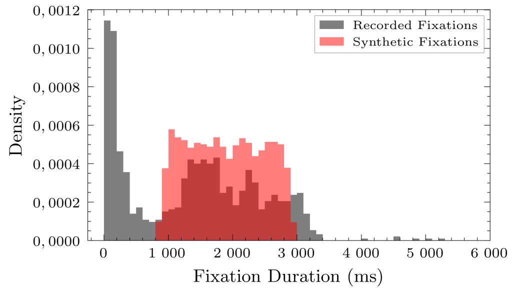


Figure 4.10: Histogram of durations for fixations in the SER and SYN datasets.

The observation that synthetic saccades are slightly too fast is reinforced by Figure 4.11. For larger saccades, the synthetic data shows a noticeably higher peak velocity compared to the recorded saccades. However, for smaller saccades, particularly those up to around 10°, the synthetic saccades closely resemble the real ones.

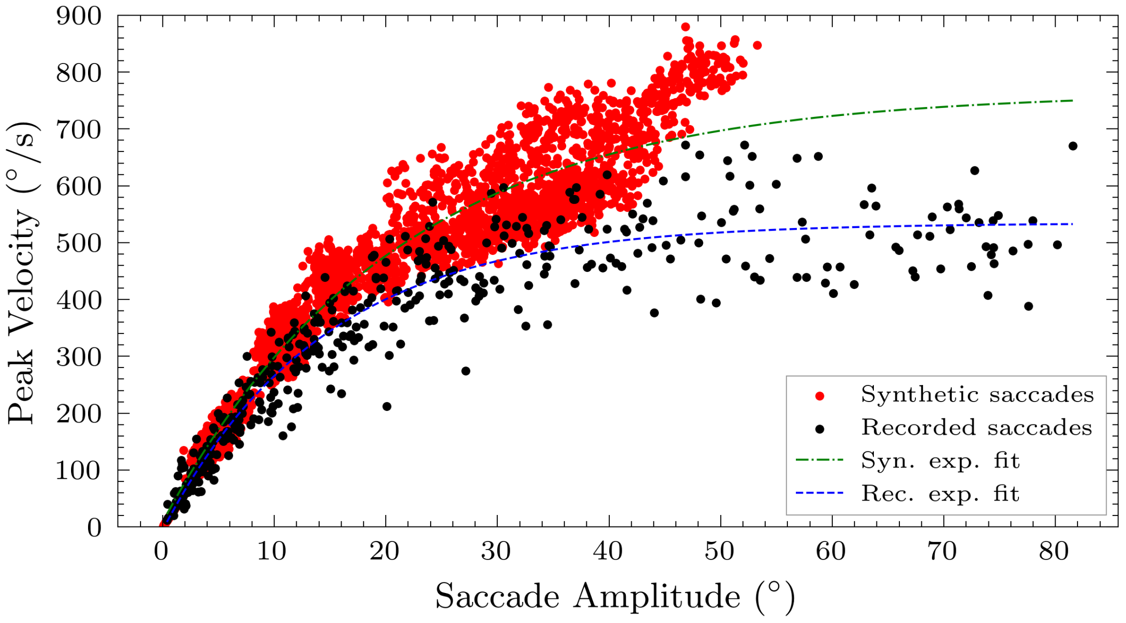


Figure 4.11: Peak saccade velocity against saccade amplitude in synthetic and recorded data. According to [1] the relationship should be approximately linear up to an inflection point at about 15°, which appears to hold true for the saccades in this study as well. The dashed blue and dash-dotted green line are exponential fits for the recorded and synthetic saccades respectively, which fit well for larger amplitudes [47].

4.3 Selection of best models

Hyperparameter sweeps were conducted for the different model types and all combinations of training datasets. The best models found from those studies were then evaluated against the test dataset, and the result of that evaluation is displayed in Figure 4.12. It shows that the kNN model trained on the dataset that was recorded for this study (SER) produced the highest score. The second highest score came from the MLP model trained on a combination of the SER and the synthetic SYN dataset. Interestingly, models trained purely on synthetic data also had decent performance. For the models trained on purely synthetic data the random forest model had the highest score.

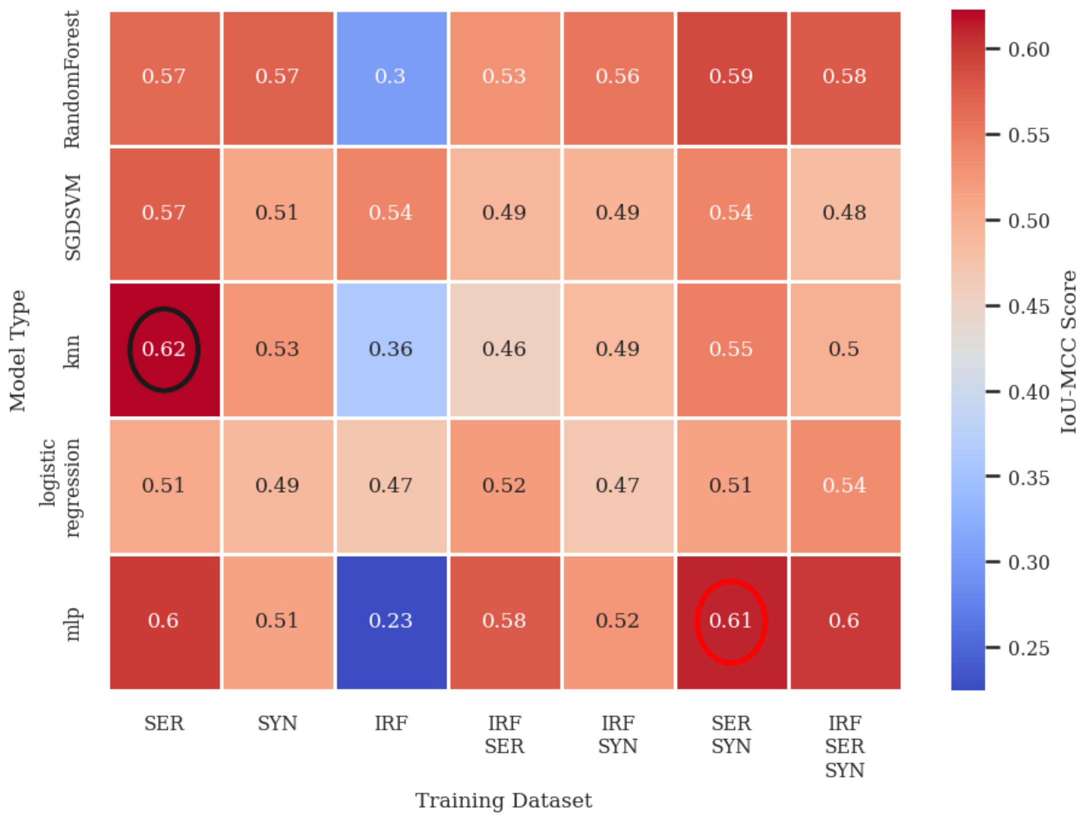


Figure 4.12: Matthews Correlation Coefficient (MCC) score for Intersection Over Union event matching for different combinations of training datasets. These scores come from applying the model to previously unseen testing data. The model with the highest score is the kNN model trained only on the Smart Eye Recorded dataset (SER), which is highlighted with a black circle. The model with the second highest score was the MLP model (marked with a red circle), trained on a combination of the SER and synthetic (SYN) datasets.

It is as previously mentioned of interest to find a small model in terms of storage, for it to be able to fit on embedded hardware where it can eventually be implemented. In Figure 4.13 the performance in relation to the size is displayed for the best model of each model type that was investigated. The SVM and logistic regression models were both very small, with their sizes being less than 1 kB, but their performance was significantly worse than the other models. The kNN model has the best performance but also the largest size. In contrast, the MLP model is significantly smaller while retaining performance similar to the kNN model.

Since the kNN and MLP models had the best performance they will be investigated further.

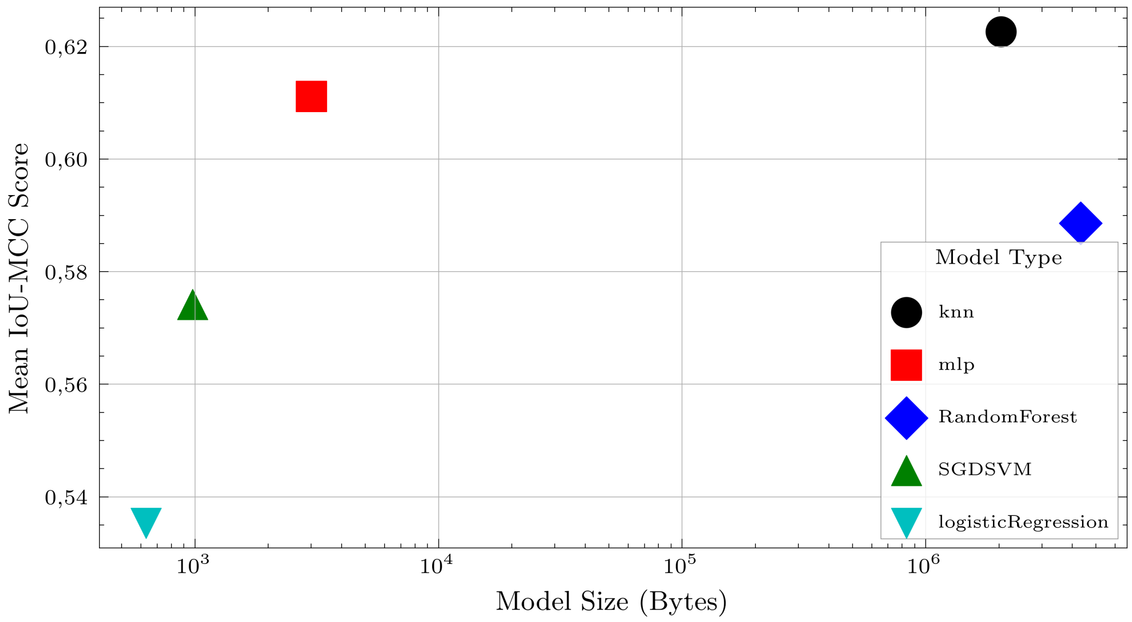


Figure 4.13: Average performance across all participants in the test dataset against model size for different models. The models displayed here are the ones trained on the collection of training datasets that gave the best result for each respective model type. Note that the x-axis is logarithmic.

4.4 Further evaluation of selected models

In Figure 4.14 and Figure 4.15 are the confusion matrices for the kNN and MLP models. Interestingly, the kNN model does not misclassify any saccades as fixations or the other way round. It does however completely miss 226 saccades and 223 fixations, which amount to 24 % of the total number of events. The MLP model is more sensitive, it only misses 206 saccades and 199 fixations (22.5 % of total events), but at the same time it has falsely classified 1 fixation as a saccade.

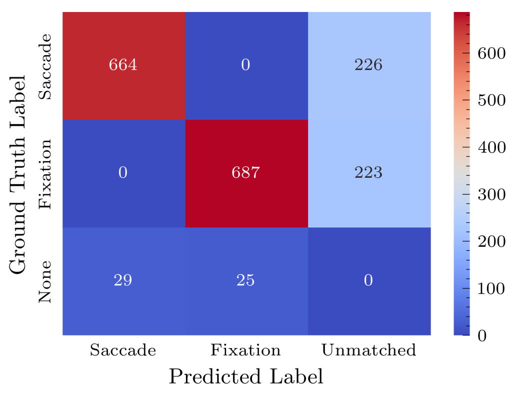


Figure 4.14: Confusion matrix for the kNN model. In the bottom row are predicted events that could not be matched with an event in the ground truth. In the column furthest to the right are events in the ground truth that could not be matched with an event in the predictions.

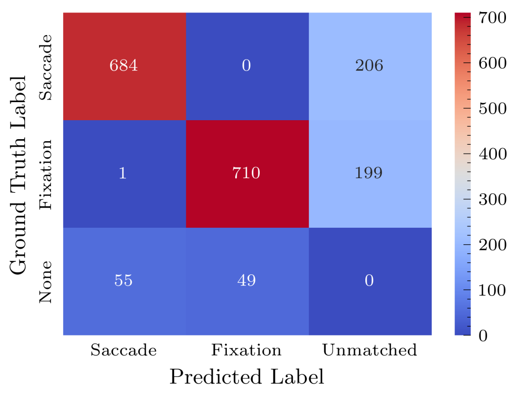


Figure 4.15: Confusion matrix for the MLP model. In the bottom row are predicted events that could not be matched with an event in the ground truth. In the column furthest to the right are events in the ground truth that could not be matched with an event in the predictions.

4.4.1 Factors that impact model performance

The models performed dissimilarly on different test subjects. Impressively, the kNN model managed to perfectly classify every single event correctly in the recording of one subject. For the most difficult subject only a IoU-MCC score of 0.27 was obtained by the kNN model. It is reasonable to assume that those differences in model performance between can be explained by some quantifiable factors. In Figure 4.16, 4.17 and 4.18 three such potential factors are investigated.

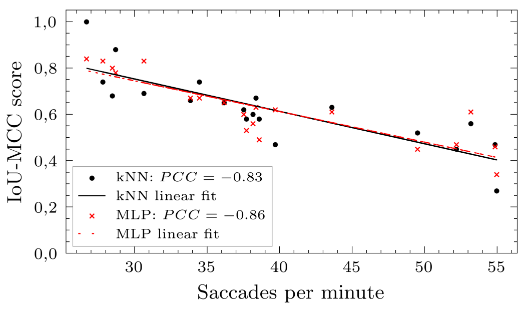


Figure 4.16: Relationship between saccade rate and performance for the kNN and MLP models. Saccade rates are calculated as the average across a recording of a single subject. The lines are linear fits to the datapoints. There is a strong negative correlation between saccade rate and performance for both models.

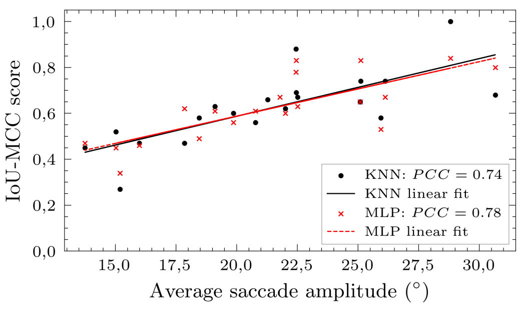


Figure 4.17: Relationship between average saccade amplitude and performance for the kNN and MLP models. Each data point is the average value for all saccades from a recording of one subject.

Figure 4.16 shows that if there are more saccades in a given timeframe the performance tends to decrease. This can be due to multiple causes. On a sample level it can for example be challenging to find a very short fixation in between two saccades, since a window based approach is used to calculate the features. If samples from both the preceding and subsequent saccade are present in the window centered at a sample in the fixation they will impact the feature values and make the fixation harder to classify correctly. From an event-level perspective, a higher concentration of fixations and saccades increases the likelihood that small detection errors, such as inaccurately estimating event duration, will result in incorrect matches.

As mentioned in Section 3.1.1.2 the viewing target made 30 jumps in a period of roughly one minute in the stimulus that was provided to the participants. In Figure 4.16 the highest scores are obtained for participants that only made those 30 saccades, without having to do any additional corrective saccades to reach the viewing target with their gaze. In the trial with 55 saccades per minute where the models performed the worst, the subject was not able to properly fixate on the stationary viewing target. They made one or two 1° to 2° saccades in every period that was supposed to be a fixation. Those saccades proved very hard to detect.

Figure 4.17 shows that trials with higher average saccade amplitude tended to get a higher score. This is in line with what is reasonable to expect, since larger movements are easier to detect than small ones. The subjects with the lowest average amplitudes either made many corrective saccades, or they were not able to fixate properly on the viewing target.

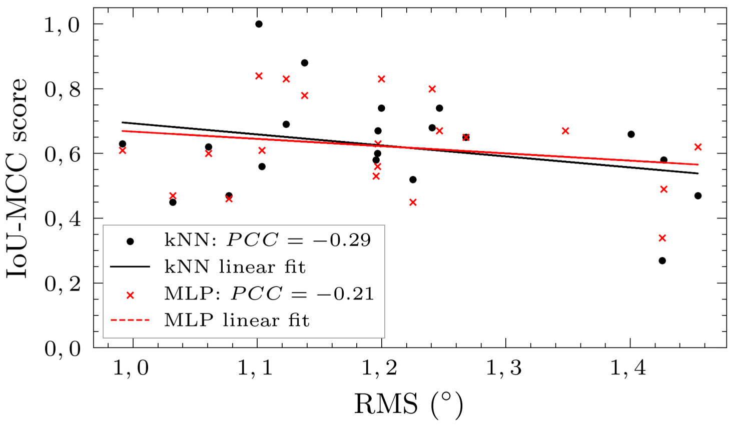


Figure 4.18: Relationship between fixational noise (RMS) and model performance. Each data point represents a recording of one subject. There is a weak negative correlation: with increased noise the score tends to decrease but it is not a majorily contributing factor.

Figure 4.18 shows the relationship between fixational noise and model performance. The fixational noise is calculated as the average noise for each fixation, where the noise for a fixation with length n is defined as

$$\mathrm{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n-1}\left[(X_{i+1,1} - X_{i,1})^2 + (X_{i+1,2} - X_{i,2})^2\right]}. \tag{4.1}$$

In Table 3.3 it was mentioned that RMS is used as a feature to detect saccades. In this case the formula used is the same, but here only segments of consecutive data points that have been labeled as fixations are considered, instead of the whole recording. Therefore, the value can in this case be seen as a measure of the background noise that is present in the recording, since the gaze is supposed to be stationary during the considered samples. This noise primarily comes from two sources, noise inherent to the measurement device and rotations of the eyeball itself [33]. Seeing that the fixational RMS values in Figure 4.18 range from numbers as high as 1 to 1.4 degrees, the largest contribution to the noise likely comes from the measurement devices, and not from movements of the eye. The variation between subjects potentially stems from how well the focus of the camera was set during the recording. Even though an adjustable chair was used to place the eyes of all subjects at the same distance from the cameras, some variation was still observed. In the recordings with the highest fixational noise the eyes of the subjects are slightly out of focus. Interestingly the negative correlation between noise and model performance between subjects is rather weak, so the saccade rate and average saccade amplitude are stronger predictors of the differences between subjects. Something to consider is that saccades with amplitudes lower than the fixational noise effectively are impossible to detect, since they are hidden below the noise floor.

In Figure 4.19 and 4.20 the saccade amplitudes of detected and undetected saccades are shown for the kNN and MLP model respectively. The smallest detected saccades had an amplitude of 2.8° and 2.4° respectively. However, below five degrees the performance was quite poor and the vast majority of saccades were missed. The blue vertical lines in the figures show the amplitude above which 95 % of saccades were detected. For the kNN model this was 4.7° and for the MLP model it was 4.3°. The amplitudes in these figures were calculated based on the ground truth, and a saccade was considered to be detected if a match between the saccade event in the ground truth and the predictions could be established.

The large spike in the saccade amplitude distribution for small values is likely due to corrective saccades, and to some extent involuntary saccades (known as microsaccades) that were conducted when the subject was supposed to be fixating. The amplitude distribution of movements in the stimulus, resulting from the parameters set in the procedure described in Section 3.1.2, was like a right skewed normal distribution.

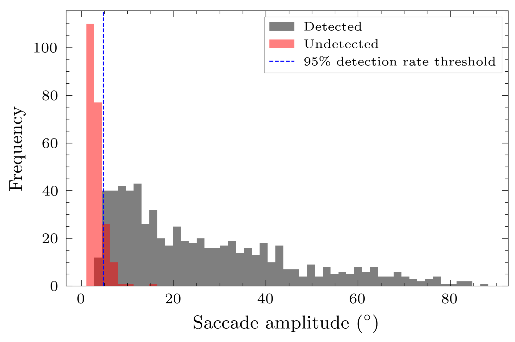


Figure 4.19: Amplitudes of saccades detected and undetected by kNN model. The amplitude of the smallest detected saccade was 2.8°, and the largest undetected saccade was 15.5°. The blue dashed line at 4.7° represents the amplitude above which 95 % of the saccades are detected.

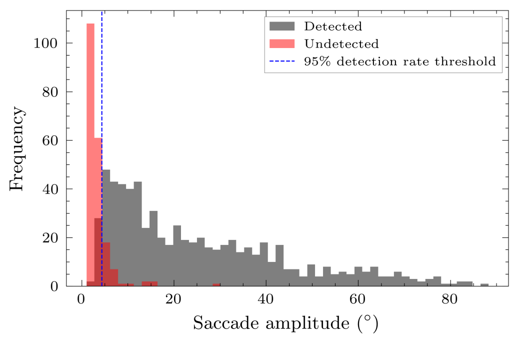


Figure 4.20: Amplitudes of saccades detected and undetected by MLP model. The smallest detected saccade was 2.4°. The largest undetected saccade had an amplitude of 29°, and is further analysed in Figure 4.21. The blue dashed line at 4.3° represents the amplitude above which 95 % of the saccades are detected.

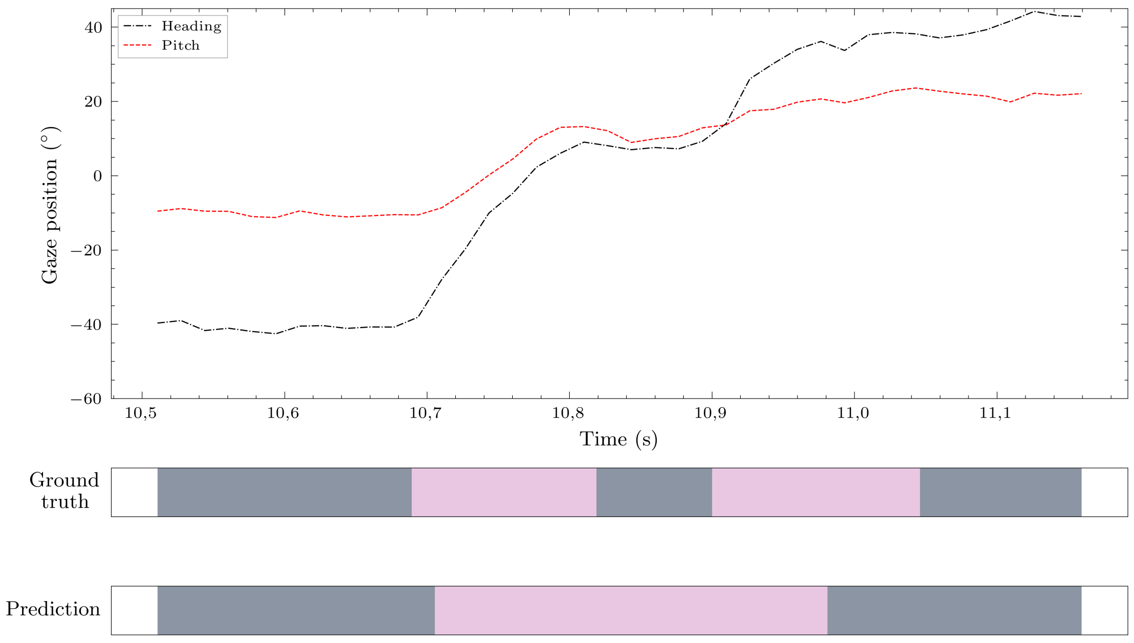


Figure 4.21: A case where the MLP model does not manage to find a large saccade. Fixations are grey and saccades are pink. The first saccade in the ground truth is matched with the predicted saccade. The second saccade (which is 29°) remains unmatched.

The largest undetected saccade in Figure 4.20 was significantly larger than the other undetected saccades. We shall examine the reasons for this discrepancy. In Figure 4.21 the raw data recorded gaze pitch and heading data is displayed over the ground truth and prediction labels. Here the viewing target has moved about 80° from the middle right side to the top left side of the screen. The participant divided this large movement into two saccades with an intermediary fixation. This intermediary fixation is not observed in the MLP model's prediction. The first saccade in the ground truth is matched with the predicted saccade because the intersection divided by the union is maximal for that matching. The following fixation and saccade remain unmatched, and therefore the second saccade is counted as not detected, even though there are samples that are labeled as saccades in the predictions for that time frame. These unmatched events correspond to the top right of the confusion matrix in Figure 4.15.

4.5 Eye-tracking metrics

In many cases where saccade detection is used, the user is primarily interested in the metrics that can be derived from those detections, rather than the detected events themselves. Such derived metrics include the mean, median and distribution of saccade amplitudes, durations and peak velocities, as well as the mean, median and distribution of the fixation durations.

Metric	**Mean**			**Median**		
Model	**GT**	**kNN**	**MLP**	**GT**	**kNN**	**MLP**
Saccade Amplitude (°)	20.49	25.35	23.85	13.21	19.99	17.31
Sac. Peak Velocity (°/s)	313.74	383.56	364.77	302.89	391.19	368.66
Saccade Duration (ms)	61.93	72.22	74.58	49.83	50.05	49.93
Fixation Duration (ms)	1369.83	1643.75	1451.55	1413.65	1696.38	1513.43

Table 4.2: Comparison of eye-tracking metrics between the ground truth (GT) and predictions from the kNN and MLP models. Both mean and median values are provided.

As demonstrated by Table 4.2 the mean and median saccade amplitude in the ground truth and predictions are quite close to each other, however the predictions produce a slightly higher value. That is likely because short saccades are missed more often than large ones, which means that the average predicted saccade amplitude becomes somewhat larger than the true value. A similar reasoning can be applied to the peak saccade velocity. In accordance with Figure 4.11 larger saccades have a higher peak velocity, up to about 30° where the curve begins to flatten out, and since shorter saccades are more often missed the average peak velocity is prone to be overestimated. The same principle applies to the overestimation of saccade duration. Saccades with shorter duration are more likely to be missed, so the mean saccade duration is a little bit overestimated. Note however that the predicted median saccade duration for both models almost perfectly align with the corresponding ground truth value. The overestimation of the fixation duration is probably due to missed saccades, which mean that multiple fixations get interpreted as one.

The distributions of the metrics in Table 4.2 are presented in Appendix C. In general the predictions follow the ground truth distributions well.

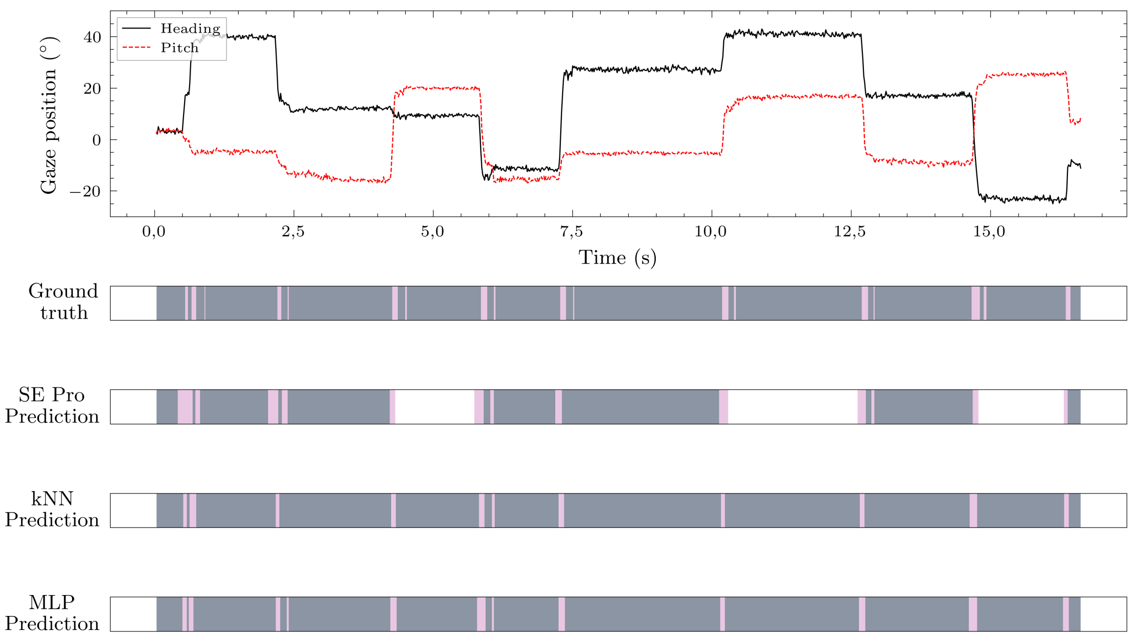


Figure 4.22: The figure shows a 16 second long purposefully selected segment where the built-in Smart Eye Pro event detector has performed well. Grey areas are fixations and pink areas are saccades. The white areas are sections where no predictions are available.

4.6 Comparison to Smart Eye Pro event detector

The Smart Eye Pro system that was used to record the SER dataset has a built in algorithm for detecting saccades and fixations. The way it is implemented is not known to the public nor the author of this thesis, but it can nonetheless be an interesting reference point for comparison of the performance of the models created in this paper. Note that the Smart Eye Pro software runs on a normal PC, and not on embedded hardware.

Two sixteen second long segments have been selected, one where the saccades are large and the Smart Eye Pro system performs reasonably well (Figure 4.22) and one where they are smaller and it struggles to make good predictions (Figure 4.23). The output of the kNN and MLP models for the same segments of data is also presented in those figures. The figures show that the built-in algorithm misses more saccades than the kNN and MLP models, and it fails to provide any predictions at all for many samples. In Figure 4.22 it appears to overestimate the length of some saccades.

Figures 4.22 and 4.23 have both given qualitative insights into how the algorithm that is included with the Smart Eye Pro software performs in comparison with the models developed in this study, for specific cases. In Figure 4.24 on the other hand, a quantitative assessment of the algorithms performance on all individuals in the test dataset is provided. The results point towards the models developed here being clearly better than the included algorithm. The score of the easiest subject for the Smart Eye Pro built in model is almost on par with the scores of the most difficult

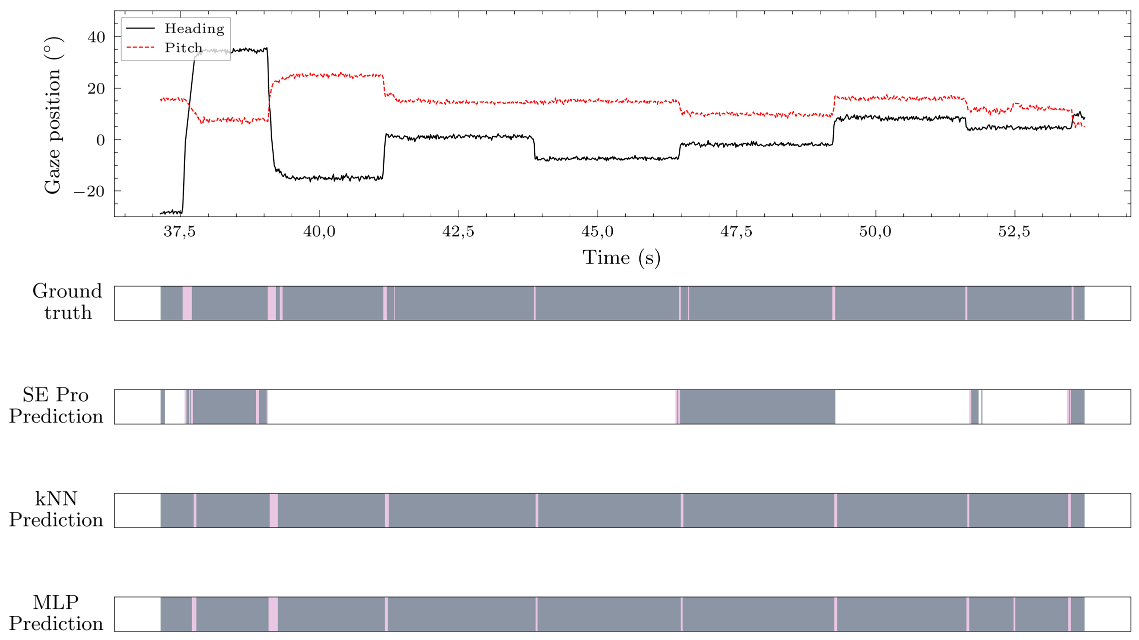


Figure 4.23: A 16 second long segment where the Smart Eye Pro built-in event detector appears to perform rather poorly. Grey areas are fixations, pink are saccades and white areas signify that no prediction has been provided for that sample.

cases for the kNN and MLP models.

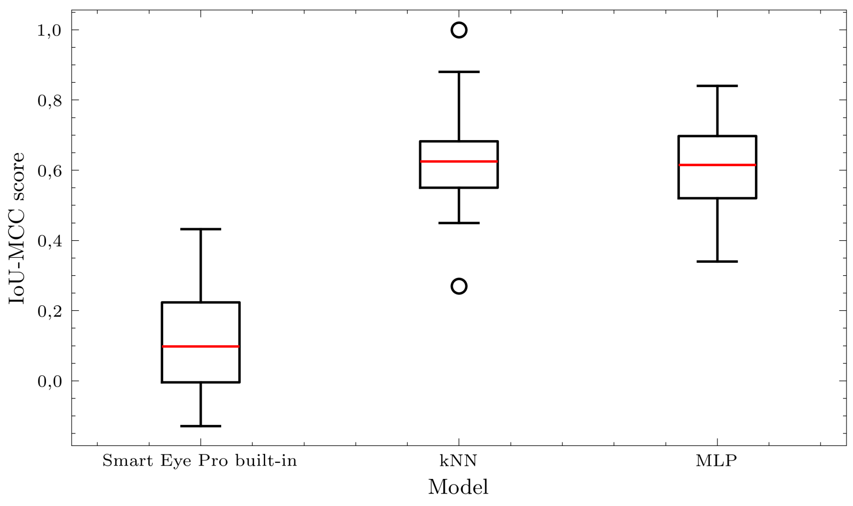


Figure 4.24: Box plot for IoU-MCC scores attained across different subjects in the test dataset for the Smart Eye Pro built-in, kNN and MLP models. Note that some scores for the built-in algorithm are even negative.

5 Discussion, conclusion and outlook

The primary objective of this thesis was to create and evaluate models for detecting events in eye-tracking data, with a particular emphasis on creating models that are small in terms of memory usage. This focus on model size is essential to enable deployment on low-performance embedded hardware, or on hardware where limited resources are shared between many processes, such as in-vehicle systems. The thesis aimed to answer three research questions:

1. **Model size and performance:** Can a data-driven model for detecting events in eye tracking data be created with satisfactory performance while maintaining a small model size suitable for embedded hardware?
2. **Use of synthetic data:** Can synthetic training data be used to enhance model performance?
3. **Training with synthetic data only:** Is it possible to train a model solely on synthetic data, that still achieves sufficient performance, potentially eliminating the need for real-world data collection?

5.1 Summary of key findings

After an initial evaluation of 5 different models, the two best performing models were more closely evaluated. They both achieved commendable performance with some small differences. The kNN model trained on the Smart Eye Recorded (SER) dataset achieved the highest performance in terms of the Matthews Correlation Coeffiecient (MCC) applied to the Maximum intersection-over-union (IoU) event matches. The MLP model trained on a combination of the SER and synthetic (SYN) datasets followed closely. Models trained on purely synthetic data also demonstrated fair performance, particularly the random forest model, albeit slightly lower than models trained on real or mixed datasets.

There was a clear trade-off between model performance and size. The kNN model, despite its superior performance, was the largest in size (2.0 MB), making it less suitable for for embedded hardware deployment. In contrast the MLP model offered a balance between size and performance, being significantly smaller (3.0 kB) than the kNN model while retaining comparable accuracy. Also note that the kNN algorithm has the inherent property that the model size scales with the size of the training dataset, which means that if a larger and more diverse training dataset were to be collected before deployment the difference in size between the models would become even larger.

The confusion matrix analysis showed that the kNN model had a high specificity, with no misclassifications between saccades and fixations. However it completely missed a significant portion of events (24 %). The MLP model on the other hand was more accurate and missed fewer events (22.5 %) but it introduced some misclassifications, falsely identifying a small number of fixations as saccades and vice versa.

A few factors in the testing data were investigated to see if and how they influenced detection performance. The first one was saccade rate. A higher saccade rate negatively impacted model performance. Trials with fewer saccades, where participants more accurately followed the visual stimuli yielded better results. The reason for increased number of saccades in some recordings was two-fold, some participants where unable to fixate without doing small micro saccades and some participants followed each large saccade with a small corrective saccade. The impact of the saccade amplitude was also investigated. Larger saccades were easier to detect, with performance improving as saccade amplitude increased. A factor that only had a small impact on performance was the amount of background noise present in the gaze signal. Although noise had a weak negative correlation with performance it was not the primary factor affecting model accuracy. However, high noise levels could potentially have obscured the smallest saccades, making them more challenging to detect.

Metrics such as average saccade amplitude, peak velocity and duration were slightly overestimated compared to the ground truth, likely due to the models missing smaller, shorter saccades. Specifically, saccade amplitude was overestimated by 16.3 %, peak velocity by 16.2 %, and duration by 20.4 %. Average fixation duration was also overestimated, which is probably a consequence of missed saccades leading to longer contiguous periods being labeled as a single fixation.

5.2 Interpretation of findings

The conclusion of the experiment is that it is possible to create a small learning based model for detecting eye movement events. The developed models perform really well for saccades larger than about 4.5°. Depending on the specific intended use case that might be sufficient, but there are applications where better performance on small saccades is required.

The easiest way to improve event detection performance further is likely to use better hardware with higher resolution and sampling rate. Both those improvements come at the cost of increased demand for processing capability. However, in vehicles where computational resources are shared across all programs and systems that are running such increased demands are often not feasible to fulfill. For systems that run on personal computers, however there are no such tangible hardware constraints, so in settings where such systems can be used performance is possible to be improved.

Discussion of method Something to consider is that throughout this thesis a high IoU-MCC score has been equated with good performance, and is what has been optimized for. So in essence most findings and conclusions that can be drawn from this study hinge on the idea that a high value in this metric properly encapsulates what it means for an eye-tracking event detector to be 'good'. It is claimed by [28] that this is the case, but they also point out that if event timing errors are of particular interest to the end user of the model, it is better to use a matching method known as the earliest overlap method. If that method of evaluating performance would have been used it is conceivable that other model types would have been the highest performing ones. However, other studies like [48] and [9] used sample-based evaluation to assess models, so relative to their approaches, this study likely performs well regardless of the specific matching algorithm used.

A post processing algorithm is applied to the models output that merges fixations that have very small saccades between them, and also merges saccades with short fixations between them. This post processing likely stands for a large contribution to the overall performance of the algorithms, as each such case would otherwise result in an additional event that needs to be matched. Potentially the built-in Smart Eye Pro algorithm could attain better performance with such post processing measures, but that is uncertain, since the inner workings of the Smart Eye Pro algorithm is unknown to the author of this study. It could even be the case that such post processing rules already are applied.

The implementation of the RMS feature in this thesis is faulty. Here the RMS value was calculated directly for the gaze pitch and heading signals, whereas in previous works the RMS value was calculated on derivatives of those signals. By differentiating the position those previous works effectively attained a RMS value for the velocity. This oversight explains why there is no separation between fixations and saccades in Figure A.10, and why the RMS features are uncorrelated with other features in Figure 4.1. Further, it also explains the reason behind the low importance of the RMS features in Figure 4.2.

Discussion of data Using the IRF dataset for training rendered poor results. It is completely expected for it to be outperformed by the SER dataset, since the validation and testing datasets were recorded under the same conditions and with the same experimental setup as the self-collected training dataset. The IRF dataset was recorded on cameras with significantly less noise, and even though synthetic noise was added to make it more similar to the testing data there might still have been differences in the noise levels. Another possible explanation is that the IRF dataset is smaller than the SER dataset, but since only 25 % of the SER data was used for training the actual difference ended up being quite small (8.1 and 11.2 minutes for the IRF and SER datasets respectively).

Three different window widths were used when extracting the features, one for the Savitzky-Golay filter, one for the average acceleration, and one for all other features. It is possible that results could have been improved slightly by tuning the window widths for all features individually. Although, only a small improvement is to be expected since it is reasonable to assume that the optimal length for all windows is similar to each other, as they all aim to capture saccades, which occur on a certain time scale.

In Figure 4.1 it can be seen that several features are strongly correlated (standard deviation and dispersion for example). This means that they largely contain the same information, so it's likely that some of them could be removed without affecting the model's performance. That would align with one of the goals of this thesis, which is to create a solution that can run on low-power hardware. If computational resources can be saved by eliminating the need to calculate unnecessary features, that would be beneficial. However, in Figure 4.2 it appears that almost all features are important. For random forests it can happen that some trees rely on certain features, while other trees rely on others. This can make multiple correlated features all appear important. For the logistic regression based feature importance calculation it might have been better to use L1 regularization instead of L2, as this potentially could have zeroed out the importance of some of the features in Figure 4.2 [23]. Future research could explore removing correlated features to see if performance is maintained. In general, reducing the number of features could decrease the computational load for feature calculation and reduce the model size.

An area where this thesis is slightly lacking is that linear interpolation has been used to fill in the approximately 1 % of values that are missing. It would have been much better to interpolate saccades and fixations in different manners. The fixations could still have linear interpolation, but it would be preferable to add some suitable noise to them, to make it more similar to the real fixation data. Saccades could be interpolated either with a sigmoid function or a higher order polynomial, depending on how much of it is missing.

There are some way that the synthetic data could be improved. From Figure 4.7 and 4.10 it can be inferred that small corrective saccades that took place in conjunction with a larger primary saccade were very common. Those saccades were the most difficult to detect correctly, and Figure 4.21 even shows an instance where the MLP model was unsuccessful at such a task. A possible reason why such short fixations might be difficult to detect is that the saccades on both sides of the fixation might end up being part of the window used to calculate features for the samples within the saccade. In the synthetic data there are absolutely no such cases, since the minimum fixation duration was set to one second. So to improve performance of models trained on synthetic data on small saccades it is not enough to include more short saccades, the fixation duration also has to be shortened. A viable strategy could be to introduce a probability of fixations being split into multiple segments with one or more short corrective saccades in between.

Another way that synthetic data can be improved is to make the saccade peak velocities better resemble the real data. In Table 4.1 it can be seen that the synthetic data has significantly higher peak velocities than the real data. In the generation of synthetic data, the saccade peak velocity is influenced by two parameters: the average saccade velocity v which was based on the fitted line in Figure 3.8, and the scaling factor of the composite sigmoid function, $g(t) = \sigma(5(2t - 1))$, which was set to 5 based on the value used by [49]. These two parameters could be fine tuned further until the peak velocities of the synthetic data match those found in the real dataset.

The noise type and magnitude in the synthetic data was set to an average of the values that were found in the recorded data. To make the models trained on the synthetic data able to generalize better to other setups and conditions, it would be a good idea to vary the noise type and magnitude when generating the synthetic data.

Practical considerations If one were to use the models created in this thesis in real time there would mainly be two sources of delay. Firstly, all the features in Table 3.3 that have `diff` in their name would contribute to close to 300 ms, since they consider windows both in front and behind of the current sample. Secondly, the post processing where event merging takes place requires at least two events in the buffer to know what events to merge. Two fixations could potentially take up to 3 s. Therefore the total delay could be around 3.5 s. The fact that there is a delay does not pose a problem for applications like driver monitoring systems, where real-time feedback is not critical, particularly in tasks such as drowsiness detection. These systems can tolerate delays as their purpose is to analyze eye movements over time rather than provide instantaneous responses. Therefore, even with a total delay of around 3.5 seconds, the models can still perform effectively in such contexts. For psychology research where the eye tracking recording can be processed after the fact, this delay does not pose an issue at all.

The models proposed in this thesis perform well for large saccades, however for small ones the performance is a bit lacking. That limits the range of possible practical applications of them. For example, performing neuroscience research on micro-saccades is not possible with the models and system used in this thesis. Nevertheless, for drowsiness detection in cars, it should be sufficient.

5.3 Ethical and Sustainability Aspects

A large ethical consideration when working with machine learning models that get applied to humans is that they might have inconsistent performance depending on factors such as gender and ethnicity of the person. It has previously been shown that ethnicity has a large impact in the performance of eye-tracking systems [50]. It is possible that the models produced in this thesis suffer from similar effects, possibly as a cascading consequence of disparities in the raw eye-tracking performance. However, this has not been investigated.

This thesis proposes a way to synthesize data, which can alleviate some privacy concerns that might arise from collecting data from real humans. Nevertheless, it introduces the challenge of ensuring that the synthetic data accurately represents real-world conditions.

5.4 Final conclusions and outlook

MLP based models appear to be a good fit for eye movement event detection running on embedded hardware, since they strike a good balance between model size and performance. The raw performance is likely on a similar level as RF based models proposed by previous studies, but the large benefit and novelty here is the fact that the MLP model can be made significantly smaller while retaining performance. The size of the random forest model proposed by [10] is ~200 MB, whereas the MLP model proposed in this study is only 1 kB. However, the comparison is not entirely just, since the previously mentioned random forest classifier was capable of detecting additional events beyond just fixations and saccades.

The models trained on purely synthetic data had reasonable performance, performing better than models trained on the IRF dataset but worse than models trained on the self-collected dataset. The conclusion that can be drawn from this result is that synthetic data, with some additional development, can be a viable alternative to real data. In addition, combining the self-collected dataset with synthetic data generally led to performance increases in this study.

Future studies can look into further development of the synthetic data. It is likely possible to make the synthetic data better resemble the real data. Furthermore, if evaluation of models point toward them under-performing in certain cases more synthetic data resembling those cases can be produced.

Bibliography

[1] Holmqvist K, Nyström M, Andersson R, Dewhurst R, Jarodzka H, Van de Weijer J. Eye tracking: a comprehensive guide to methods and measures. 1st ed. Oxford ; New York: Oxford University Press; 2011.

[2] Majaranta P, Bulling A. Eye Tracking and Eye-Based Human–Computer Interaction. In: Fairclough SH, Gilleade K, editors. Advances in Physiological Computing. London: Springer; 2014. p. 39-65. Available from: `https://doi.org/10.1007/978-1-4471-6392-3_3`.

[3] Singh H, Bhatia JS, Kaur J. Eye tracking based driver fatigue monitoring and warning system. In: India International Conference on Power Electronics 2010 (IICPE2010); 2011. p. 1-6. ISSN: 2160-3170. Available from: `https://ieeexplore.ieee.org/abstract/document/5728062`.

[4] Oyster CW. The human eye: structure and function. Sunderland, Mass: Sinauer; 1999.

[5] Boyce PR, Ditchburn R. Monocular fixation in human eye movement. Proceedings of the Royal Society of London Series B Biological Sciences. 1967 Mar;167(1008):293-315. Publisher: The Royal Society. Available from: `https://royalsocietypublishing.org/doi/epdf/10.1098/rspb.1967.0028`.

[6] Nyström M, Holmqvist K. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. Behavior Research Methods. 2010 Feb;42(1):188-204. Available from: `https://doi.org/10.3758/BRM.42.1.188`.

[7] Hessels RS, Niehorster DC, Kemner C, Hooge ITC. Noise-robust fixation detection in eye movement data: Identification by two-means clustering (I2MC). Behavior Research Methods. 2017 Oct;49(5):1802-23. Available from: `https://doi.org/10.3758/s13428-016-0822-1`.

[8] Salvucci DD, Goldberg JH. Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the 2000 symposium on Eye tracking research & applications. ETRA '00. New York, NY, USA: Association for Computing Machinery; 2000. p. 71-8. Available from: `https://dl.acm.org/doi/10.1145/355017.355028`.

[9] Birawo B, Kasprowski P. Review and Evaluation of Eye Movement Event Detection Algorithms. Sensors. 2022 Jan;22(22):8810. Number: 22 Publisher:

Multidisciplinary Digital Publishing Institute. Available from: https://www.mdpi.com/1424-8220/22/22/8810.

[10] Zemblys R, Niehorster DC, Komogortsev O, Holmqvist K. Using machine learning to detect events in eye-tracking data. Behavior Research Methods. 2018 Feb;50(1):160-81. Available from: https://doi.org/10.3758/s13428-017-0860-3.

[11] Zemblys R, Niehorster D, Holmqvist K. gazeNet: End-to-end eye-movement event detection with deep neural networks. Behavior Research Methods. 2018 Oct;51.

[12] Ieva M. Eye Movement: Types and Functions Explained;. Available from: https://www.tobii.com/resource-center/learn-articles/types-of-eye-movements. Accessed: 2024-08-26.

[13] Schleicher R, Galley N, Briest S, Galley L. Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired? Ergonomics. 2008 Jul;51(7):982-1010. Available from: https://doi.org/10.1080/00140130701817062.

[14] Vorstius C, Radach R, Lang AR, Riccardi CJ. Specific visuomotor deficits due to alcohol intoxication: Evidence from the pro- and antisaccade paradigms. Psychopharmacology. 2008 Feb;196(2):201-10. Available from: https://doi.org/10.1007/s00213-007-0954-1.

[15] Human Eye Diagram - Human Body Pictures & Images - Science for Kids;. Available from: https://www.sciencekids.co.nz/pictures/humanbody/eyediagram.html. Accessed: 2024-08-20.

[16] Nyström M, Hooge I, Holmqvist K. Post-saccadic oscillations in eye movement data recorded with pupil-based eye trackers reflect motion of the pupil inside the iris. Vision Research. 2013 Nov;92:59-66. Available from: https://www.sciencedirect.com/science/article/pii/S0042698913002356.

[17] Holmqvist K, Andersson R. Eye-tracking: A comprehensive guide to methods, paradigms and measures; 2017.

[18] Smart Eye Pro – Remote Eye Tracking System;. Available from: https://www.smarteye.se/smart-eye-pro/. Accessed: 2024-08-20.

[19] Wolf M. Mathematical Foundations of Supervised Learning; 2023. Available from: https://mediatum.ub.tum.de/doc/1723378/1723378.pdf. Accessed: 2024-08-20.

[20] Parmigiani G. Decision Theory: Bayesian. In: International Encyclopedia of the Social & Behavioral Sciences. Elsevier; 2001. p. 3327-34. Available from: https://linkinghub.elsevier.com/retrieve/pii/B0080430767004034.

[21] Murphy KP. Machine learning: a probabilistic perspective. 4th ed. Adaptive computation and machine learning series. Cambridge, Mass.: MIT Press; 2012.

[22] Bishop CM. Pattern Recognition and Machine Learning. Softcover reprint of

the original 1st edition 2006 (corrected at 8th printing 2009) ed. Information science and statistics. New York, NY: Springer New York; 2016.

[23] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.

[24] Ranganathan P, Pramesh CS, Aggarwal R. Common pitfalls in statistical analysis: Logistic regression. Perspectives in Clinical Research. 2017 Sep;8(3):148. Available from: `https://journals.lww.com/picp/fulltext/2017/08030/common_pitfalls_in_statistical_analysis__logistic.9.aspx`.

[25] Steinwart I, Christmann A. Support vector machines. 1st ed. Information science and statistics. New York: Springer; 2008.

[26] Olson M. Essays On Random Forest Ensembles [PhD Diss.]. University of Pennsylvania; 2018.

[27] Liu Y, Wang Y, Zhang J. New Machine Learning Algorithm: Random Forest. In: Liu B, Ma M, Chang J, editors. Information Computing and Applications. Berlin, Heidelberg: Springer; 2012. p. 246-52.

[28] Startsev M, Zemblys R. Evaluating Eye Movement Event Detection: A Review of the State of the Art. Behavior Research Methods. 2023 Jun;55(4):1653-714. Available from: `https://doi.org/10.3758/s13428-021-01763-7`.

[29] He H, Ma Y. Imbalanced learning: foundations, algorithms, and applications. Hoboken, New Jersey: John Wiley & Sons, Inc; 2013.

[30] Hoppe S, Bulling A. End-to-End Eye Movement Detection Using Convolutional Neural Networks. arXiv; 2016. ArXiv:1609.02452 [cs]. Available from: `http://arxiv.org/abs/1609.02452`. Accessed: 2024-08-22.

[31] Startsev M, Agtzidis I, Dorr M. 1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits. Behavior Research Methods. 2019 Apr;51(2):556-72. Available from: `https://doi.org/10.3758/s13428-018-1144-2`.

[32] Savitzky A, Golay MJE. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry. 1964 Jul;36(8):1627-39. Publisher: American Chemical Society. Available from: `https://doi.org/10.1021/ac60214a047`.

[33] Niehorster DC, Zemblys R, Beelders T, Holmqvist K. Characterizing gaze position signals and synthesizing noise during fixations in eye-tracking data. Behavior Research Methods. 2020 Dec;52(6):2515-34. Available from: `https://doi.org/10.3758/s13428-020-01400-9`.

[34] Peirce J, Gray JR, Simpson S, MacAskill M, Höchenberger R, Sogo H, et al. PsychoPy2: Experiments in behavior made easy. Behavior Research Methods. 2019 Feb;51(1):195-203.

[35] Thaler L, Schütz AC, Goodale MA, Gegenfurtner KR. What is the best fixation target? The effect of target shape on stability of fixational eye

movements. Vision Research. 2013 Jan;76:31-42. Available from: https://www.sciencedirect.com/science/article/pii/S0042698912003380.

[36] Fridman L, Lee J, Reimer B, Victor T. Owl and Lizard: Patterns of Head Pose and Eye Pose in Driver Gaze Classification. arXiv; 2016. ArXiv:1508.04028 [cs]. Available from: http://arxiv.org/abs/1508.04028. Accessed: 2024-07-16.

[37] Joseph VR. Optimal ratio for data splitting. Statistical Analysis and Data Mining: The ASA Data Science Journal. 2022 Aug;15(4):531-8. Available from: https://onlinelibrary.wiley.com/doi/10.1002/sam.11583.

[38] Ludwig CJH, Gilchrist ID. Measuring saccade curvature: A curve-fitting approach. Behavior Research Methods, Instruments, & Computers. 2002 Nov;34(4):618-24. Available from: https://doi.org/10.3758/BF03195490.

[39] Carpenter RHS. Movements of the eyes, 2nd rev. & enlarged ed. Movements of the eyes, 2nd rev. & enlarged ed. London, England: Pion Limited; 1988. Pages: 593.

[40] Collewijn H, Erkelens CJ, Steinman RM. Binocular co-ordination of human vertical saccadic eye movements. The Journal of Physiology. 1988;404(1):183-97. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1988.sp017285. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1988.sp017285.

[41] IRF dataset;. Available from: https://github.com/r-zemblys/irf/tree/master/etdata/lookAtPoint_EL. Accessed: 2024-09-28.

[42] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825-30.

[43] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods. 2020 Mar;17(3):261-72. Available from: https://www.nature.com/articles/s41592-019-0686-2.

[44] pandas development team T. pandas-dev/pandas: Pandas. Zenodo; 2024. Available from: https://zenodo.org/doi/10.5281/zenodo.3509134. Accessed: 2024-09-06.

[45] Hooge ITC, Niehorster DC, Nyström M, Andersson R, Hessels RS. Fixation classification: how to merge and select fixation candidates. Behavior Research Methods. 2022 Dec;54(6):2765-76. Available from: https://doi.org/10.3758/s13428-021-01723-1.

[46] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework. arXiv; 2019. ArXiv:1907.10902 [cs, stat]. Available from: http://arxiv.org/abs/1907.10902. Accessed: 2024-04-15.

[47] Gibaldi A, Sabatini SP. The saccade main sequence revised: A fast and repeatable tool for oculomotor analysis. Behavior Research Methods. 2021 Feb;53(1):167-87. Available from: `https://link.springer.com/10.3758/s13428-020-01388-2`.

[48] Andersson R, Larsson L, Holmqvist K, Stridh M, Nyström M. One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms. Behavior Research Methods. 2017 Apr;49(2):616-37. Available from: `https://doi.org/10.3758/s13428-016-0738-9`.

[49] Zemblys R. ETRA2024 Demo.ipynb; 2024. Available from: `https://github.com/r-zemblys/EM-event-detection-evaluation/commits/main/demo/ETRA2024%20Demo.ipynb`.

[50] Blignaut P, Wium D. Eye-tracking data quality as affected by ethnicity and experimental design. Behavior Research Methods. 2014 Mar;46(1):67-80. Available from: `https://link.springer.com/10.3758/s13428-013-0343-0`.

A

Feature Distributions for Saccades and Fixations

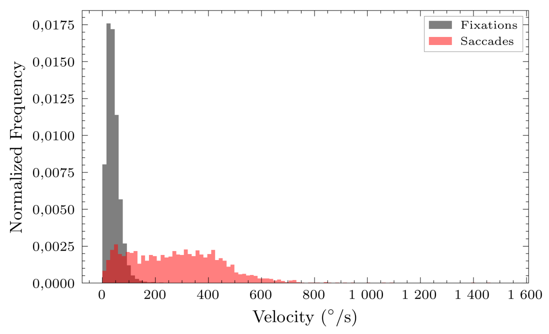


Figure A.1: Distribution of calculated velocity for samples labeled as fixations and saccades.

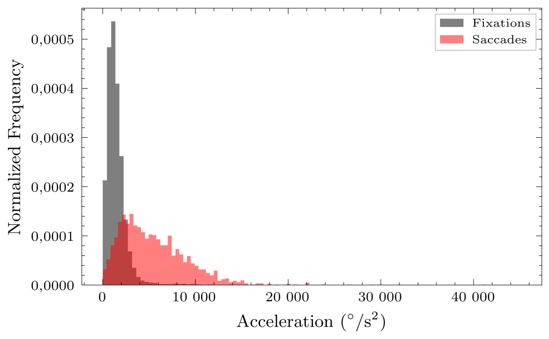


Figure A.2: Distribution of calculated acceleration for samples labeled as fixations and saccades.

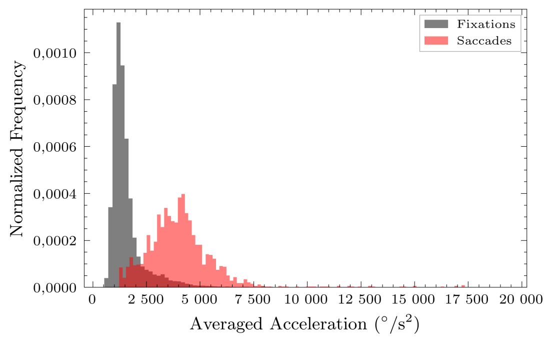


Figure A.3: Distribution of averaged acceleration for samples labeled as fixations and saccades.

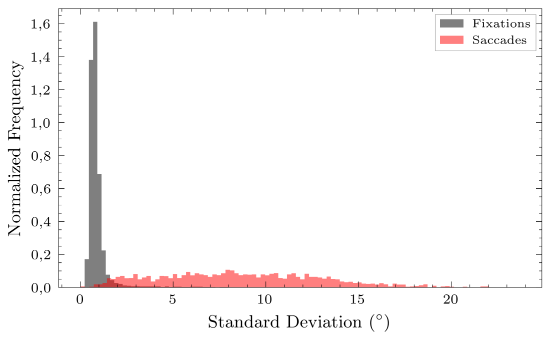


Figure A.4: Distribution of standard deviation for samples labeled as fixations and saccades.

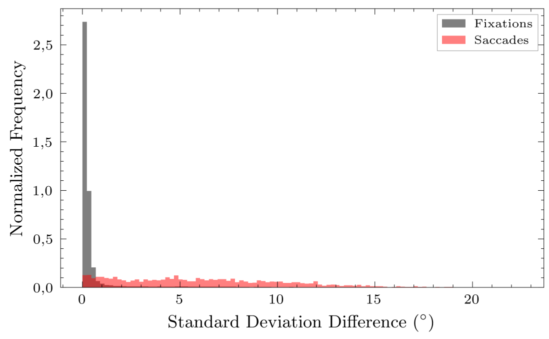


Figure A.5: Distribution of standard deviation difference for samples labeled as fixations and saccades.

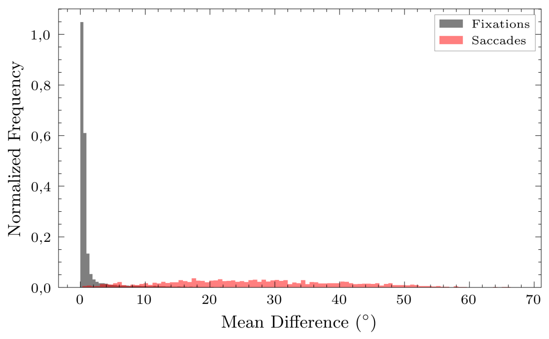


Figure A.6: Distribution of mean difference for samples labeled as fixations and saccades.

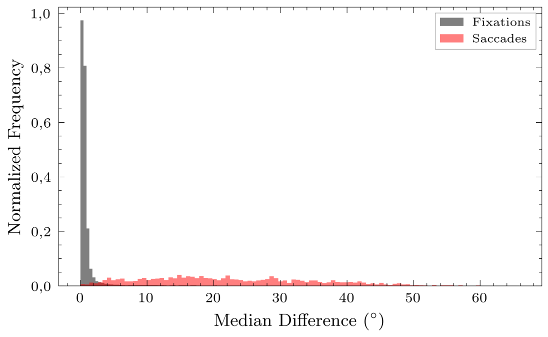


Figure A.7: Distribution of median difference for samples labeled as fixations and saccades.

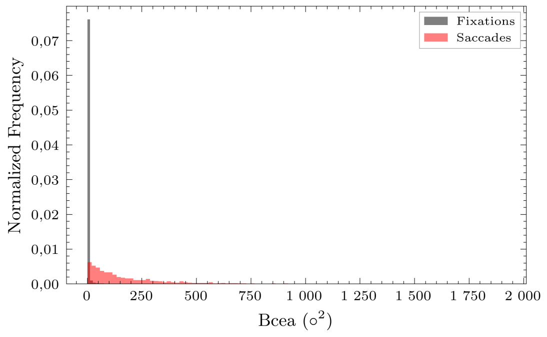


Figure A.8: Distribution of bivariate contour ellipse area for samples labeled as fixations and saccades.

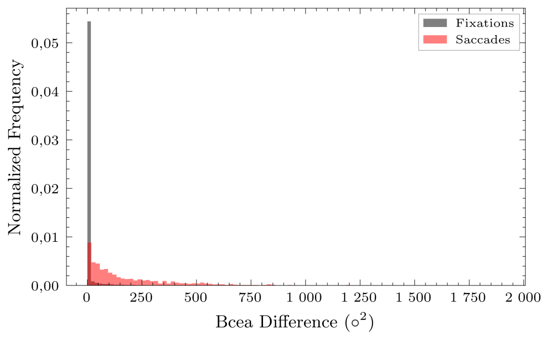


Figure A.9: Distribution of differences of bivariate contour ellipse areas for samples labeled as fixations and saccades.

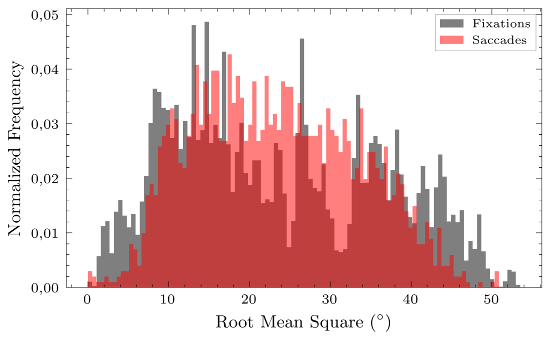


Figure A.10: Distribution of rms values for samples labeled as fixations and saccades.

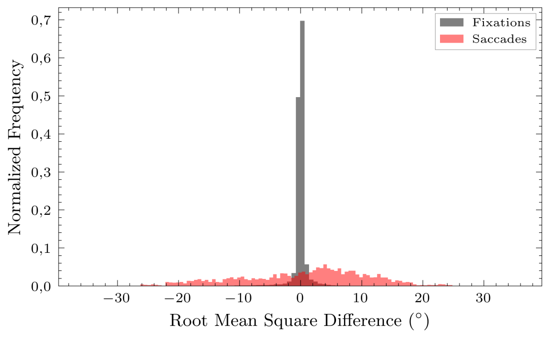


Figure A.11: Distribution of differences in rms values for samples labeled as fixations and saccades.

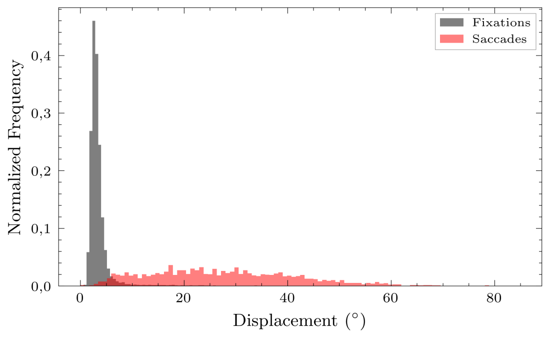


Figure A.12: Distribution of differences in dispersion values for samples labeled as fixations and saccades.

B

Hyperparameters

Model	**Hyperparameters**
MLP	hidden_layer_sizes: *30*, activation: *tanh*, solver: *lbfgs*
RandomForest	n_estimators: *41*, max_depth: *25*, max_features: *3*
KNN	n_neighbors: *8*, weights: *distance*, metric: *euclidean*
SGD SVM	alpha: *4.2379552507070424e-10*, penalty: *l1*
Logistic Regression	C: *6.619705958704763e-07*, penalty: *l2*

Table B.1: Hyperparameters for the models with best performance for each type (those in Table B.2 with yellow highlighting). Parameters not mentioned are the set to default values in Scikit-learn version 1.2.2.

Model	Dataset	Window width for other features	Avg acc window width	Savgol Filter Width
MLP	SER + IRF	287	213	159
	SER	77	278	85
	SER + SYN + IRF	196	269	104
	SER + SYN	104	278	88
	IRF	229	111	94
	SYN + IRF	138	262	58
	SYN	206	234	69
Random Forest	SER	77	292	73
	SER + IRF	230	218	125
	SER + SYN + IRF	85	182	61
	SER + SYN	80	294	56
	IRF	199	169	192
	SYN + IRF	268	269	123
	SYN	190	232	121
kNN	SER + IRF	266	232	120
	SER	71	270	58
	SER + SYN + IRF	244	251	126
	SER + SYN	92	257	131
	IRF	179	200	123
	SYN + IRF	181	211	126
	SYN	154	251	120
SVM	SER	72	283	51
	SER + IRF	224	249	129
	SER + SYN	97	280	69
	SER + SYN + IRF	227	253	158
	IRF	250	68	162
	SYN	112	176	93
	SYN + IRF	245	250	164
Logistic Regression	SER	88	189	164
	SER + IRF	89	247	160
	SER + SYN	195	149	127
	IRF	296	279	99
	SYN	181	96	137
	SYN + IRF	285	168	127
	SER + SYN + IRF	84	256	154

Table B.2: Window widths used for various models and datasets. The numbers presented are in milliseconds. However, only the IRF dataset was recorded at 1000 Hz. Since the SER and SYN dataset are 60 Hz values within 16 ms of each other can be considered the same. The rows highlighted in yellow shows the best performing model for each model type.

C

Comparison of Eye-Tracking Metrics: Ground Truth Versus Model Predictions

In this appendix histograms comparing metrics from the recorded data and predictions are presented. These are the underlying distributions that form the mean and median values presented in Table 4.2.

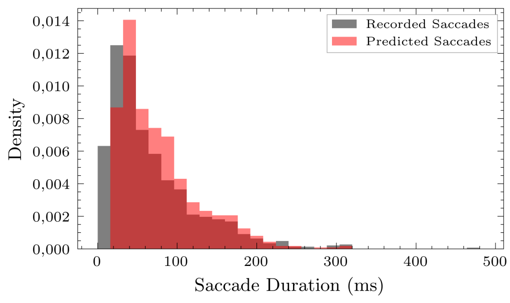


Figure C.1: Histogram of saccade durations comparing kNN model predicted saccades (red) with ground truth saccades (grey background). The Kullback-Leibler (KL) divergence between the distributions is 1.7988, indicating the difference between the predicted and actual distributions.

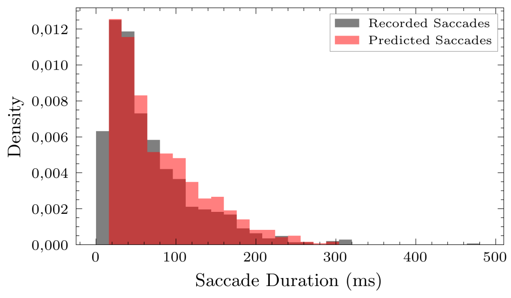


Figure C.2: Histogram of saccade durations comparing MLP model predicted saccades (red) with ground truth saccades (grey background). The Kullback-Leibler (KL) divergence between the distributions is 1.8152, indicating the difference between the predicted and actual distributions.

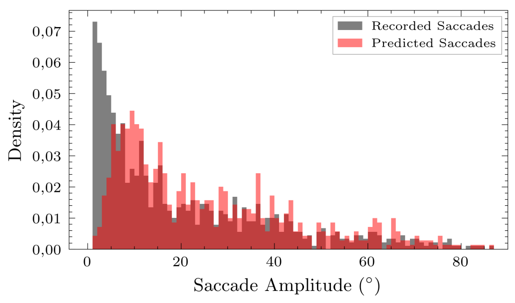


Figure C.3: Histogram of saccade amplitudes comparing kNN model predictions (red) with ground truth saccades (grey background). The Kullback-Leibler (KL) divergence between the distributions is 0.4020, indicating the difference between the predicted and actual distributions.

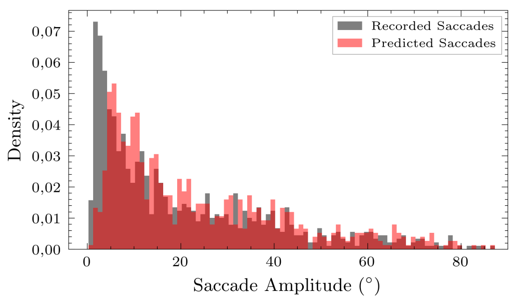


Figure C.4: Histogram of saccade amplitudes comparing MLP model predictions (red) with ground truth saccades (grey background). The Kullback-Leibler (KL) divergence between the distributions is 0.3290, indicating the difference between the predicted and actual distributions.

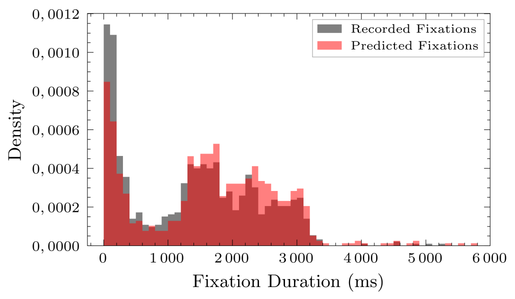


Figure C.5: Histogram of fixation durations comparing kNN model predictions (red) with ground truth fixations (grey background). The Kullback-Leibler (KL) divergence between the distributions is 0.0838, indicating the difference between the predicted and actual distributions.

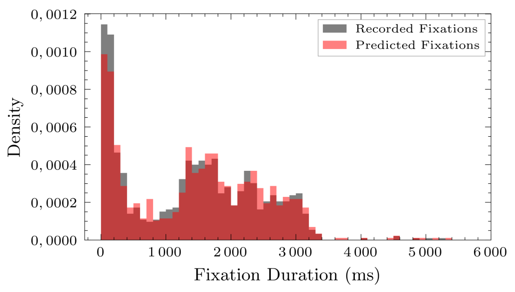


Figure C.6: Histogram of fixation durations comparing MLP model predictions (red) with ground truth fixations (grey background). The Kullback-Leibler (KL) divergence between the distributions is 0.0489, indicating the difference between the predicted and actual distributions.

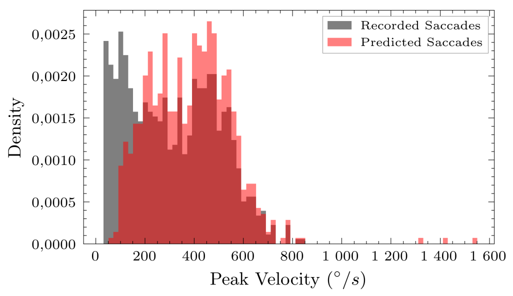


Figure C.7: Histogram of peak saccade velocity comparing kNN model predictions (red) with ground truth fixations (grey background). The Kullback-Leibler (KL) divergence between the distributions is 1.0040, indicating the difference between the predicted and actual distributions.

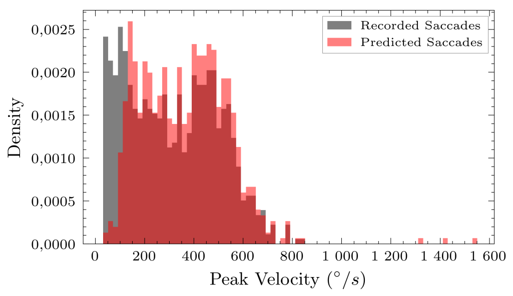


Figure C.8: Histogram of peak saccade velocity comparing MLP model predictions (red) with ground truth fixations (grey background). The Kullback-Leibler (KL) divergence between the distributions is 0.2381, indicating the difference between the predicted and actual distributions.