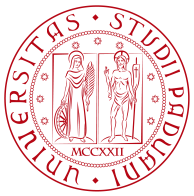


University of Padova

Department of Mathematics

Master Thesis in Data Science

I Spy With My Little Eyes: A Convolutional Deep Learning Approach to Web Eye Tracking

Supervisor
Dr. Lamberto Ballan
University of Padova

Co-supervisors
Dr. Antonia Krefeld-Schwalb
Dr. Sebastian Gabel
Rotterdam School of Management

Master Candidate
Ivan Dragomirov Padezhki

Academic Year
2022-2023

This thesis is dedicated to the people around me that reminded me that the feeling of not knowing anything is only temporary. Feeling incapable is not to be confused with feeling new to a topic of research. I don't remember the number of times I called friends and family to relieve some of the insecurities or to ask for someone to body-double me when I could not shut down the many voices in my head that told me to go back to bed and give up. I dedicate this thesis to the people that lifted me up in these moments.

Y'all know who you are. I can't wait to see everyone be successful. Honestly, work.

Abstract

Eye-tracking is the study of eye movements, blinks, and fixations, and it aims to give insight into visual attention mechanisms. Being common in marketing, usability research, as well as in cognitive science, there are well-established methods for lab eye tracking, yet web eye tracking uses webcams of much lower quality. If the accuracy of web methods would improve, users' engagement with digital content can be analyzed from the comfort of their own homes. This gives designers, developers, and researchers the chance to inform their decisions from data and optimize e.g. user experience while connecting to large and demographically diverse samples without the necessity for lab-level equipment. The limited tools for web eye-tracking are also accompanied by uncertainties coming from the setup environment that need to be addressed before using them for scientific research. The project aims to develop a deep learning solution building on CNNs capable of predicting gaze x/y screen coordinates from the webcam video of users. A new dataset collected in the eye-tracking lab consists of face images and reliable eye-tracker coordinate predictions for each image. Models trained on the dataset achieve lower prediction errors in terms of pixels compared to existing methods for web eye-tracking. Future studies should focus on validating the methods in different experimental conditions.

Contents

Listing of figures

Listing of tables

Listing of acronyms

OCR . . . . . Optical Character Recognition

P-CR . . . . . Pupil-Corneal Reflection

DL . . . . . Deep Learning

ML . . . . . Machine Learning

NN . . . . . Neural Network

LSTM . . . . . Long Short-Term Memory

CNN . . . . . Convolutional Neural Network

VAE . . . . . Variational Autoencoder

GAN . . . . . Generative Adversarial Network

VR . . . . . Virtual Reality

AR . . . . . Augmented Reality

LoG . . . . . Line of Gaze

LoS . . . . . Line of Sight

IR . . . . . Infrared

1 Introduction

Starting from a broader introduction of the focus of this study, eye tracking is often used in neuromarketing research. As a branch of neuroeconomics, neuromarketing uses the study of the functioning of the human mind concerning decision-making, more specifically in economic tasks, and intends to improve traditional marketing techniques. With eye tracking spreading fast in goggles, laptops, VR, and AR headsets [1], it is predicted to become even more relevant soon [2]. Consumers and their different types of responses to marketing stimuli, be they sensorimotor, cognitive, or affective responses, are the focus of such studies. Additionally, there is the possibility to predict consumers' behavior and also manipulate their needs. Predicting user intent and interest is driven by gaze estimation and can be used in marketing analysis [3].

An ongoing effort in marketing, psychology, and neuroscience is to expand research to online data collection. For behavioral data, even with high timing precision [4], various software makes online testing possible (jsPsych [5], PsychoPy [6]), while platforms like Prolific Academic and MTurk ease the contact to large and diverse participant groups [7, 8]. Nevertheless, lab-based experiments often include methods that require special technology. Thus, developing online solutions for researchers is a challenge, especially in the case of eye-tracking. The current study aims to provide an overview of existing options for online eye tracking and develop a novel technique inspired by lab technology. As there are concerns regarding the accuracy, validity, and privacy of web-based eye tracking, further studies are needed to bring these methods to wider use.

In practice, this information can help with the evaluation of reactions and interactions with

websites and the improvement of usability (especially in combination with mouse behavior: clicks and movement). Brands can thus focus on developing features of their products that are most attractive and usable by clients. Specifically, assuming there is a correlation between gaze movement and degree of attention [9] on a visual stimulus, one can optimize the effectiveness of advertising communication [10]. In practice, this can mean improving the product life cycle by utilizing gaze-based preference for products, say on a website, to avert unsuccessful product launches [11]. Use cases of this nature apply to product placement for improving brand recognition[12]; predicting user experience in VR locomotion [13]; and also further developing automatic driving by considering gaze location of both drivers and pedestrians (suggested in [3]). Furthermore, such a product could be used in educational settings [14] in [15]. It can also be extended to populations of people who rely on human-computer interaction devices to communicate or perform certain activities [3].

After covering some application examples of why webcam eye tracking is a valuable tool to improve, a brief introduction of eye tracking methods is necessary to motivate the purpose of the current study. The techniques generally investigate eye movements, gaze behavior, and pupil dilation among others [16]. Different movement types are covered in 2.1. There are three main methods for gaze estimation – 3D eye model recovery, 2D eye feature regression, and appearance-based methods [17, 18]. Each has its strengths and weaknesses, depending on the physical implementation that can sometimes be costly, or on the constraints related to the testing environment. The details of each method can be found in Chapter 2, however, to position the analyses developed in the current story to the three methods, some primary explanations are necessary.

3D model recovery relies on using a geometrical model of the eye to extract features (the center of the cornea and certain axes of the eye [17]) and requires technology such as infrared light sources and high-resolution cameras, as well as controlled testing conditions [16]. A 2D regression model maps the pupil center and corneal reflection vector to eye coordinates on screen using a polynomial function [17]. Lastly, appearance-based methods use predefined algorithms for extracting a person's face, eye regions, and pupil. This can often be done with commercial-grade cameras. An extension of this is deep learning appearance-based methods, which rely on images or videos and on the ability of neural networks to extract features automatically and to learn a function from the image space to the space of screen coordinates. This last technique requires large data sets with annotated images, which are not trivial to obtain [18, 17, 19]. Simultaneously, such solutions for eye tracking with affordable web cameras would allow researchers to collect data simply by sending their participants a link.

To date and to the best of my knowledge, there are no studies that try to combine the strengths of these in-lab and web-based methods to provide researchers with an easy-to-use solution for web eye tracking. Therefore, the novelty of this study lies in using collected webcam videos as input for a neural network, while giving the high-quality predictions of a lab-level eye tracker as labels. First of all, this ensures a higher level of reliability of the targets compared to manually annotated data sets. Secondly, combined with data from a promising online solution for eye tracking, namely WebGazer, the study aims to provide a comparison between the accuracy of the lab-based tracker EyeLink and WebGazer, as well as between WebGazer and the presented model. Thus, the general contribution of the study lies in expanding the range of online eye-tracking methods with a novel idea for training a neural network while maintaining a strict reporting practice when comparing different methods. This entails a clear description of all data collection, and processing steps, ensuring transparency of the method. This way, researchers can reproduce and personalize the algorithms to their needs. The careful consideration of privacy guidelines is also relevant for any eye-tracking research and is considered at every step of the process.

The thesis is structured as follows. The basics of eye tracking, including frequently used methods are detailed in Chapter 2. Then, the experiment and data collection, as well as data processing and algorithms are introduced in Chapter 3. The following Chapter 4 reports the results and evaluation of the algorithms with a discussion. These are summarized and expanded with their implications in Chapter 5 which also provides concluding remarks and limitations of the work.

2
Related Work

This chapter has the goal of providing all necessary preliminary knowledge to readers from any background: data science, marketing, and psychology. Therefore, the first step is a brief description of how eyes are studied. Then, lab and online eye-tracking methods are detailed, followed by an overview of deep-learning solutions for eye-tracking. This way, the reader is capable of understanding the positioning of the current research in this framework for eye-tracking studies and methods.

2.1 Studying human eyes

There is much known about how the human visual system works. Specifically, the field related to eye movements has been researched since the 1900s albeit with invasive technologies, such as scleral coils on an eye lens [17, 16]. The study of eye movements has been developed since and is crucial to understanding visual attention and cognition [17, 19].

With some idea of the eye anatomy given in Figure 2.1, the next important aspect to mention is the different types of eye movements that have been extensively studied. Firstly, there are saccades, short (10-100 ms) moves, usually indicating a visual search between fixations. The latter is described in relation to information acquisition and processing and is comprised of three smaller eye movements: microsaccades, tremor, and drift [19]. Additional markers are the blink rate, smooth pursuits, and changes in pupil size, usually indicative of cognitive workload. For a detailed overview, the reader is advised to consult Table 1 in [17].

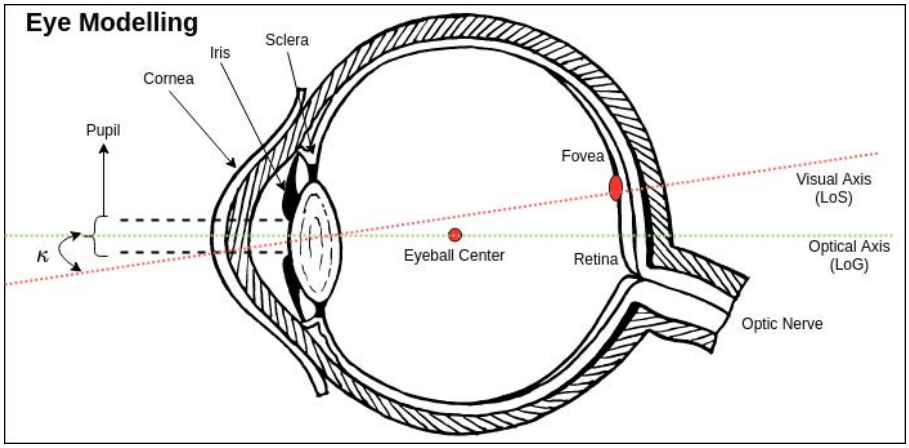


Figure 2.1: The model used commonly to understand eye physiology in the context of tracking gaze. Relevant parts are labeled in the image. The angle between the Line of Gaze (LoG) and the Line of Sight (LoS) is called kappa (κ) and is unique for each person. The anatomy of the human eye is essential for allowing light to hit the fovea, which is the central point of focus humans see. Figure is adapted from [19] .

Studying these types of movements has inspired a wide variety of eye-tracking solutions, both invasive and non-invasive. [16] offers a summary of many of them, including a concrete guideline for reporting the procedures and setup followed in eye tracking research. The different solutions implement algorithms that mostly fall into three ways for estimating gaze: 2D regressions, 3D models, and appearance-based methods. As this study is not considering 3D model recovery methods, these are briefly explained here, while the other methods are detailed in the following sections with examples of implementations.

3D methods aim to compute the center of the cornea and two relevant axes (LoG, LoS) with the help of a geometric 3D model of the eye. The place where the optical and visual axes intersect the scene/screen [17] is what these models are trying to compute. The cited review study reports better accuracy for setups with multiple cameras and involved calibration procedures, potentially due to increased tolerance to head movement resulting from the extensive hardware setups. Related to this, it is important to note that due to the unique characteristics of the eyes (the kappa angle), personal calibration is also required. Despite [20] arguing that the lack of 3D model-based reasoning may cause less stable predictions, this method relies heavily on high-resolution images, and as already mentioned, custom camera setups.

In the next section, a common method for eye tracking is presented: an example of a 2D regression method with its physical implementation is the EyeLink by SR Research (Mississauga, ON, Canada).

2.2 Eye tracking in the lab

One widely-used method is the Pupil-Corneal Reflection video-based system which includes one or more cameras and infrared illuminators. This type of eye tracking system is accompanied by high costs, sometimes up to 40,000 euros [15]. The procedure consists in detecting one or both eyes in video frames and mapping the extracted information from eye geometry, pupil, and corneal reflection to screen coordinates. 2D regression methods rely on a mapping with a polynomial transformation function, such as $f : (X_{eye}, Y_{eye}) -> (X_{screen}, Y_{screen})$. The polynomial equations for X, Y are optimized to the minimal value of the mean squared difference between real and predicted screen coordinates related to the calibration points in the calibration procedure, explained further below.

EyeLink is an example of this type of tracking. The IR light illuminates the eyes sufficiently for the distinction of different features of the eye by the tracker's model [21]. It uses the corneal reflection (CR) induced by the IR light, and the pupil coordinates (P), all captured by a camera as images or video. The relative displacement between the two is used to compute gaze direction, thus giving the method the name P-CR: the corneal reflection coordinate is subtracted from the pupil center coordinate [22, 23]. One reason for choosing this tracker for the study is that there is the biggest number of comparative studies which use the EyeLink 1000 or 1000+ as a benchmark [16]. Furthermore, EyeLink uses feature recognition to track the gaze – discovering the pupil and corneal reflection by thresholding, which is part of the process which should be modeled by a neural network. Thus, comparing EyeLink with the presented models allows a comparison of the underlying algorithms: 2D regression methods and deep learning appearance-based methods.

In practice, a minimal lab setup includes an eye tracker, a computer screen, a chair for the participant, and a desk [24]. The specific EyeLink setup additionally has a host PC, which runs on DOS and is connected to the eye tracker with an Ethernet cable. This allows for the built-in camera to log data points at very high speed: settings can be changed between 1000 and 2000Hz for most EyeLink models. For any eye tracking research, it is crucial to report the geometry of the setup, defined as the absolute position and orientation of the eyes and the tracker [25] and also of the screen. One argument why reporting these technical specifications is crucial for replicating study results is that if the screen size is too large, it leads to poorer precision of gaze estimation in the outer parts of the screen [26, 27] reported in [16]). Furthermore, the distance to the eye tracker should be considered a relevant factor (approximately 60-70 cm in most lab settings). Overall, one can summarize sources of errors into a list of categories, such as head

pose, display and camera properties, user distance and set-up geometry, illumination, and user characteristics [17].

With many individual and environmental sources of variation to consider, a crucial step in this type of eye tracking is calibration. As mentioned, each individual has unique eye parameters [17], so adapting the model to these parameters takes place during a task in which the user/participant follows a point jumping or moving to predetermined locations on the screen and fixating on them. This allows the tracker to compute a mapping function from the eye positions to screen locations.

As both characteristics of the eye and the geometry of the setup determine the quality of the recording, it is likely that variations in these are linked to artifacts in the data. Relevant artifacts stem from factors such as pupil size, pupil occlusion, and mascara: i.e. very thick eyelashes can obscure the proper detection of the pupil. Lighter eye color is also commonly connected to lower accuracy and precision [28]. Illumination in the room can also cause bad quality of the data. Light changes during recording affect the size of the pupil, leading to a decrease in gaze accuracy [29, 16]. Direct sunlight also negatively affects the quality of the data [30, 31] as infrared radiation can interfere with the one coming from the tracker itself.

Another source of variation comes from the head position. For a specific formulation of the head-eye correspondence, the reader is referred to [17]. In many lab scenarios, the participant is sitting in a chair that does not allow for considerable body movements, and their head is placed on a chin rest. There is some evidence that using a chin/ forehead rest can help against small movements as it restricts the position of the head inside what is called a head box (the relative position of the head to the eye-tracker), thus making it less likely that the tracker will lose the position of the eye [28].

With various sources of error described, it is accepted that methods relying on IR light and high-resolution and high-frequency cameras to have high accuracy with the limitations that the equipment should be acquired first and the participants should be in a controlled environment [3]. Cameras with high sampling frequencies such as the EyeLink (1000Hz and above) allow researchers to track saccades and the even faster micro saccades [32]. The required minimum for recording saccades is 100 Hz, while micro saccades require a minimum of 200 Hz to be recorded [33]. On the other hand, some methods use visible light cameras. Instead of the EyeLink setup, some researchers construct their setup using IR LEDs and webcams which have much lower frequency. Because most web cameras available at home have a sampling rate of about 30Hz, it is not a plausible goal of the current research to develop a method for studying saccades with a webcam.

2.3 Eye tracking outside of the lab

Next, we turn to appearance-based or feature-based methods, which include the transformation of pixels in the eye region to abstract features i.e. pupil, cornea, limbus, eye corners, or also histograms of oriented gradients and other computer vision descriptors. These descriptors are mapped to predicted gaze points or directions [17]. Commonly required components of an appearance-based method are a feature extractor, a regression function, and training samples to optimize the function. A feature extractor has to efficiently compute low-level gaze features such as histograms of oriented gradients [18]. Machine learning algorithms are used as regression functions. Besides many other options, one can name adaptive linear regression [34, 18], support vector regression [35], Gaussian process regression [36, 37], and convolutional neural networks (CNN) [38, 39] have been proposed. Since the last mentioned study demonstrates that this method is not disrupted by low-resolution eye images compared to model-based methods, a CNN is chosen for the modeling in the current study.

Furthermore, [3] summarizes that 2D and 3D model-based methods tend to be better for gaze estimation when the user is at a greater distance and the camera images are of high resolution. Therefore, this research investigates deep learning appearance-based methods. DL-based methods are reported to have the ability to extract high-level gaze features and to learn non-linear functions, thus becoming more robust and accurate than conventional appearance-based methods [18].

Before turning to DL extensions, the current subsection introduces common appearance-based methods. From a broad perspective, these methods attempt to solve the issue of studying eye movements remotely, via an online experiment. There are certain available options - some tools are proprietary and have to be bought from a company, while others are open-source libraries. One option is to approximate the gaze estimation process. An example of this is manual gaze scoring, which includes trained raters evaluating which area on screen a participant is looking at from a webcam video. There is also a method for automating the scoring presented in [40]. This type of paradigm comes with a low spatial resolution (e.g. the scoring of gaze location could be divided into left, middle and right sections of the screen). Yet the automated scoring is supported by evidence to be appropriate for this type of research: GazeScorer is reported to agree with the human scorer both in children and adult samples, with the advantage of requiring much less time. Therefore, more complex models are expected to improve the spatial resolution of the predictions. Another option for online tracking is the use of a proxy for gaze location: this entails the study of mouse tracking and is possible through packages such as

Mouseview.js [41].

The previously mentioned example of webcam tracking that also utilizes an appearance-based method is WebGazer [20]. It is an open-source project written in JavaScript, which can be integrated into any web page to track people's eye movements. WebGazer uses the webcam video stream as input. However, unlike the lab eye tracking described previously, the calibration works by fixating on a dot on the screen while clicking it with the mouse. This is due to the underlying assumption that during web browsing, the gaze location is highly correlated with the click location at the time of a mouse click. Some support for this comes from findings on gaze-cursor distance being smaller in screen areas attended by the user, [42] in [20]. Therefore, clicks are used as weights in the regression to calibrate the model and also to ensure continuous adjustment of the gaze location estimations in the period between clicks. Sometimes, this calibration process can be very involved. In many cases, it has to be repeated throughout the experiment and it also does not prove that the assumption holds. Studies investigating the gaze-cursor distance report 74 pixels distance when a click occurs [43], and also a larger distance for x-axis coordinates [44].

Nevertheless, as an example of an appearance-based method, WebGazer is the tool chosen to investigate in the current study. The model first finds the face and eye regions. The original paper reports that three libraries are tested for this, including dlib's 68-point facial landmark detector. Then, a median blur filter is applied for noise reduction followed by thresholding to find the iris area. This uses the assumption that within the eye region, the pupil is expected to be located in the center of the iris. The latter is circular and darker than its surroundings. After applying a grayscale color transform and histogram equalization, the model's input is ready. The weight vector is represented as:

$$w = (X^T X + \lambda I)^{-1} X^T Y. \tag{2.1}$$

where X is a matrix with eye features and Y is the target vector of screen coordinates, while the regularisation term aims to prevent overfitting ($\lambda = 10^{-5}$). As already mentioned, WebGazer uses information about the clicks to calibrate the model. Added to the regression are the predictions within 500 ms time windows from a click and 72 pixels distance from the click location. The authors name these variations of the algorithms ridge regression and a fixation buffer. In addition to this, WebGazer is expanded by including a cursor movement matrix K in the calculation of the weight vector:

$$w = (X^T K X + \lambda I)^{-1} X^T K Y. \tag{2.2}$$

The matrix has an entry of 1 if a click occurred and 0.5 during the movement of the cursor. A decaying rule is applied, decreasing the cursor location's weight by 0.05 after 20 ms. The results in [45] are based on 802 predictions from the simple linear regression model and 866 predictions from the model variations (cursor behavior, clicks, ridge regression, and fixation buffer). According to the analysis, considering additional information on cursor movements can improve the accuracy of predictions and decrease the error (quantified as the distance between the click and coordinate prediction made by the model).

In addition to the calibration, the WebGazer might also require high experimenter involvement and also lead to high attrition rates [46]. Sometimes it is enough that the lighting conditions in the participant's room are bad and the algorithm will report that the user should try again later. Unstable environments are something that developers of methods for online research should be aware of. Nevertheless, there are big advantages of tools like WebGazer – they allow researchers to reach demographically diverse samples. With extensive validation of the method not present yet, there is only some evidence of its accuracy. WebGazer has been tested in some behavioral and cognitive tasks and has led to discovering effects like the ones found with other eye tracking methods [47, 48]. It has been compared to manual gaze scoring in infant populations [46], yet as the study underlines, until improved deep learning tools for gaze coordinate prediction are developed, WebGazer is the tool that researchers have to rely on.

On a last note, there is a variety of methods available online that researchers could purchase. Unfortunately, the algorithms behind these tools are unknown, and so are the databases used for training them. One example is a gaze detection classifier available on the Amazon Rekognition software [49]. As parts of the algorithm are not open-source, a direct comparison to this method is not possible. Therefore, the next section is concerned with presenting deep-learning solutions which are not proprietary software or require the acquisition of expensive data sets.

2.4 Deep learning solutions

An extension of appearance-based methods is the use of DL algorithms. These have to perform well under a variety of conditions: occlusion, eye size, head pose, and illumination to name a few. In any case, a DL model most commonly receives an RGB image as input, $I \in \mathbb{R}^{W \times H \times 3}$ usually pre-processed to include the face or eye regions. The DL architecture is then responsible

for learning a representation of eye features and mapping them to gaze coordinates.

In practice, studies differentiate between single-path architectures and multi-path ones. In the first case, the network receives only the face, or only the eyes as input, while in the latter, it can be a combination. Furthermore, one can also include other inputs such as head pose and eye models. A natural extension of such architectures considers the dynamic nature of gaze movement, thus working with video as input and including recurrent elements in the network. Lastly, some studies have proposed transformer designs, autoencoders, and adversarial networks as solutions (overview in [18, 19]). In the following paragraphs, these examples will be detailed and their strengths and weaknesses summarized.

Before presenting relevant studies, a first note related to the factor of head position is important. Deep learning architectures and specific techniques have been proposed to tackle the issue of head movement, such as clustering input images with similar head positions ([50] in [19]), or image synthesis ([51] in [19]). Besides existing techniques for dealing with this issue, one study [38] suggests that a CNN is capable of detecting eyes in images as long as the head is positioned in a way that is usually seen when using a computer system. Since the current study does not consider head movement, as it will be detailed in Chapter 3, the reader is referred to these studies for the description of the techniques.

As already mentioned, to estimate gaze coordinates, a model requires an element that extracts features from the visual input and one that maps them to coordinate values. The section begins with a study that separates the two tasks and continues with end-to-end methods.

A paper that illustrates the separation of the feature extraction and coordinates prediction parts of a DL model addresses real-time gaze tracking from video. After extracting facial landmarks related to facial and eye features with a separate open-source toolbox (DeepLabCut), the authors of [52] use the coordinates of these landmarks as input to a shallow feed-forward NN with one hidden layer of 200 neurons, followed by a sigmoid activation function and two output neurons for the coordinates. The authors report using Stochastic Gradient Descent (SGD) with a learning rate of 0.3 and momentum of 0.6. The error function is a derivation of the L2 loss. The full data set used in the study has 3569 frames. Their system is reportedly comparable to the accuracy of previous studies with similar approaches, reaching a median error of one degree of visual angle, even observing the effect that x-axis coordinate predictions have larger errors compared to the y-axis. A similar effect was reported for the WebGazer algorithm above - larger errors for the x-axis predictions. While this study serves as an example of the abilities of neural networks to learn mappings, it also suggests that with a stronger feature extractor, there could be a single network that does all the work.

Thus, convolutional neural networks are discussed next. Firstly, [15] use an IR LED light to illuminate the eyes but a simple webcam to capture images. This low-cost solution for online gaze monitoring is tested by using YOLOv3 to detect the gaze position. Despite the low cost of the method, it still includes an infrared light used to determine the head position in relation to the screen by classic computer vision techniques such as thresholding in OpenCV. Furthermore, according to the authors, considering the pose of the head did not influence the results. While large-scale datasets exist, the authors describe them as unsuitable for developing a low-cost solution, as the images in these datasets have very high resolution. Therefore, a custom dataset of 3000 manually labeled images is used to further train the last three layers of the pre-trained YOLOv3 network. The achieved accuracy is 2 degrees of visual angle. While the authors report higher accuracy for a Mask-RCNN, they warn that using a recurrent network lacks the speed necessary for the tool to be deployed in real-time.

With certain CNN architectures already tested for eye tracking, specifically for iris recognition, it is reasonable to assume that this architecture can be used as a feature extractor [53]. Recent studies include modified versions of the VGG family ([54]), AlexNet, ResNet-18, and ResNet-50 which all learn from RGB images, either presented in one stream (only face or only eyes) or multiple branches (left eye, right eye, face). Commonly, these networks rely on the convolution operation to extract features.

Firstly, GazeNet is a classic CNN example for gaze estimation. Its input is in the form of an eye patch which is processed by a VGG-13 network before having the head pose concatenated to the first fully connected layer. ([55] in [18]). During training, the tracked loss is the sum of individual L2 losses between predicted and true gaze angle vectors [18]. The model reportedly outperforms LeNet [56].

In a further effort to increase the receptive fields without further reduction in the spatial resolution of images due to the high number of convolutional layers, the authors ([57] in [18]) use dilated convolution while keeping the VGG-like architecture. Specifically, it has four blocks of stacked dilated convolutional layers, a max-pooling layer, and two fully connected ones. Another important difference in this study is the multi-input nature of the CNN - each eye patch as well as an image of the full face goes through the whole network and the results are concatenated before the fully connected layers. The decision to use dilated convolution is motivated also by the fact that some eye movements lead to very small changes in pixel appearance, thus the authors wanted the network to extract these features from the images with higher resolution. The formulation of dilated convolution is expressed as follows. The output feature map v is:

$$v(x,y) = \sum_{k=1}^{K} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} u(x + nr_1, y + mr_2, k) w_{nmk} + b. \tag{2.3}$$

where U is a feature map of kernel size $N \times M \times K$, which are height, width and channel respectively. The parameters (r_1, r_2) stand for dilation rates. The network is trained using Euclidean distance as the error, Adam as an optimizer, and a learning rate of 0.001, which was decreased according to a fixed scheme.

Similarly to the previous study others consider that including multiple inputs improves the inferential ability of the network [19]. So, there have been various combinations of input data. Some refrain from including images of the whole face, as it may contain unnecessary information ([58] in [19]), keeping a branch for each eye patch. Others, such as the iTracker [59] include a binary mask indicating the face location as additional input besides the eye patch branches. Lastly, some studies propose mechanisms such as spatial weighting that learns to encode the face location in the input images [60].

Next, to consider the dynamic nature of the gaze, recurrent architectures are employed for modeling this temporal aspect. This turns the input from images to videos, notwithstanding that in practice, each frame undergoes a static CNN for feature extraction. Then, the features are fed into an RNN. For this, common recurrent structures have been explored, such as GRU [61] in [62]. Recurrent modules are supposed to additionally improve gaze estimates of the model given the high correlation between subsequent frames [57, 19]. The first mentioned study [57] reports an improved accuracy of 2.49 degrees angular error over similar static methods by using a ResNet-18 component for feature extraction and a GRU cell. The study additionally evaluates an LSTM [63], a convolutional recurrent cell [64], and RNN [65]. Similarly, [66] takes advantage of pinball LSTM, using a window of seven frames for predicting the gaze direction of the frame in the middle. Like [61], the main CNN is inspired by ResNet-18, which receives the full facial region as input. Overall, despite comparable results from studies considering temporal information, reported open issues remain to be addressed, among these the gaze directions, velocities, and trajectories [19].

On a last note, [19] summarizes transformer-based efforts in the field of gaze estimation, revealing a potential application of fast-spreading transformer architectures. Two main explorations in this field are GazeTR-Pure and a hybrid version, which combines a CNN with a transformer [67]. The main motivation behind this choice of a hybrid network is the attention mechanism, which is expected to consider information beyond the local correlation in patches that the pure transformer computes. According to the authors, the hybrid network produces

better results in gaze estimation than the pure transformer and than a simple CNN (ResNet-18). Since this is one of the few reviewed methods from the last years, transformer architectures seem to be a promising path for gaze estimation. Other networks include GAN and VAE solutions, which reportedly suffer from time complexity issues [19].

Overall, the choice of eye tracking method and gaze estimation algorithm should be based on what is required for one's research questions. Before experimenting, one should get familiar with the available tools and their limitations. Another issue accompanying the development and popularization of tools for remote eye tracking is the ethical and legal concerns of web eye tracking. Most methods described above are based on the user's webcam stream as input. This raises considerable privacy concerns, as researchers should value the subjects' right to anonymity. A review study [68] has discussed the possibility of extracting unique biometric data and personal attributes such as age, personality traits, and cognitive processes from eye-tracking data. Fraser et al., 2021 suggest precautions in every step – from collecting the data to storing and retaining it. Therefore, following the GDPR is crucial for this type of project. The section about data collection covers the privacy procedure followed during the current study.

3 Methods

Motivated by the purpose of the study, which is to develop an alternative low-cost solution for web eye-tracking, and by existing experiments with DL approaches, a clear description of the research process is required. Past review studies have pointed out the lack of consistent reporting of experimental conditions [16]. Therefore, this chapter covers the specifics of the data collection, including the task, setup, and experimental protocol. Then, the dataset and its preprocessing are described. The final section is concerned with the algorithms chosen for tackling the task of learning to predict screen coordinates from images. As the focus of this study is to expand the range of online eye-tracking solutions, supervised algorithms are selected as the learning strategy. This decision is further motivated by the goal of comparing WebGazer with a lab-based method, EyeLink, and identifying limitations in both approaches. Data from these sources is collected simultaneously, further enabling the comparison between WebGazer's predictions and the trained models' predictions. To this end, a simple online choice task is created and administered to participants in the Erasmus Behavioral Lab.

3.1 Experiment and data collection

As an open-source package, WebGazer can be added to any webpage. This way, on top of any normal functions of the web page, a user can record the location of their gaze on the screen. Therefore, a choice task was adapted to a web page. Common experiment builder software and packages, including the EyeLink Experiment Builder, are not equipped for presenting a

web page as a stimulus while recording gaze data. However, EyeLink has a new software, WebLink, which is specifically designed for this purpose. The software was acquired by the Erasmus Behavioral Lab. It enables the synchronized collection of EyeLink data as well as a video recording of the participant while they interact with a web page. WebLink itself is responsible for triggering the EyeLink tracker, the webcam video stream, and for opening the web pages in Chrome. WebGazer collects task choices and predictions in a separate database. The process is explained in detail in the following paragraphs.

The task given to participants consists in selecting between two options, each presenting a sum of money that would be received after some period of time. In 36 trials in a randomized order, participants had time to observe the options and make a choice by clicking on a button located below the information boxes. As WebGazer was discovered to suffer from a decreasing accuracy with experiment duration, another student in the same faculty (Rotterdam School of Management) aimed to tackle this issue by creating two conditions. One sees the choice buttons in a fixed location, the other condition randomizes the location of the buttons. The latter is expected to lead to increased accuracy of WebGazer's predictions, as participants would have to look at the button's new location while clicking it, hence utilizing the continuous calibration functionality of the tool.

Examples of both types of trials can be seen in Figure 3.1. More specifically, the randomized button positions are calculated each time the user presses "Next". For the left button, its new position is calculated as a distance from the bottom and left border of the page:

$$\text{Math.floor(Math.random()} * 25 + 10) \tag{3.1}$$

. The JavaScript function $Math.random()$ returns a float between 0 and 1. For the bottom offset, the random number was multiplied by 5 instead of 25. This has the goal of guaranteeing the choice buttons do not appear on the wrong side of the "Next" button. The location of the right button is determined in a mirrored way. The contents of each trial are predetermined, yet the order of their presentation is randomized.

The 36 trials are interrupted by frequent calibration steps every 5 trials. As already mentioned, the calibration process for WebGazer is different compared to standard lab-based trackers. The user is asked to fixate on each red dot and click it 5 times until it changes color to yellow, see Figure 3.2. Once all 5 dots have been completed, an accuracy estimation is performed: the participant has to fixate on a central dot for 5 seconds without moving the cursor. This step records an accuracy measurement (distance between estimated gaze location and actual dot

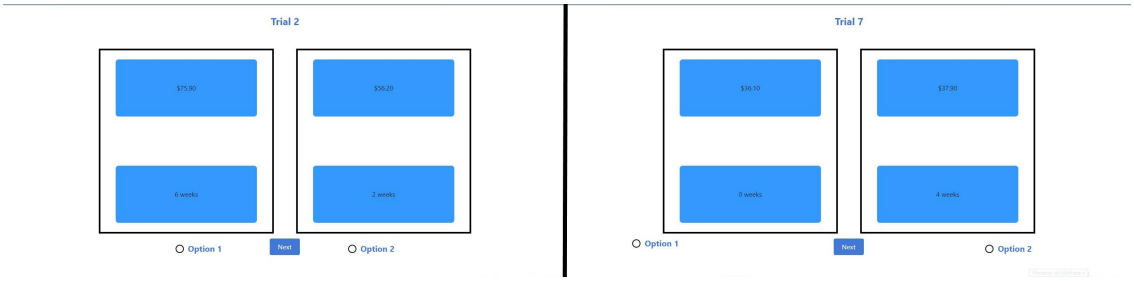


Figure 3.1: The fixed button condition is on the left, and an example of a potential randomized position is on the left. The base experiment was created by Paul Kievits, a Scientific Developer at Rotterdam School of Management, Erasmus University, who adapted an experiment from the Brown HCI Group. The button positions and trial information was added by the experimenter.

location). The involved, and repeated, calibration process aims to continuously improve the prediction accuracy.

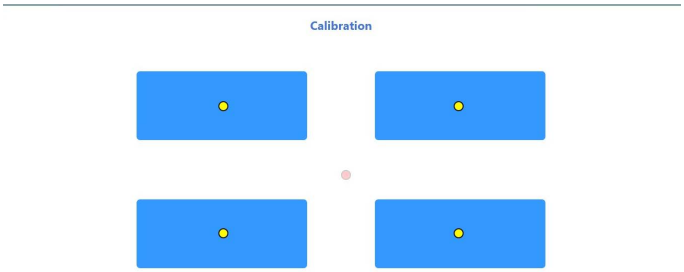


Figure 3.2: In the process of calibration participants initially see the four dots on the blue background and only once these have been completed, the central one appears. The location of the central calibration dot is the same as the accuracy estimation dot.

The WebGazer functionality is embedded into a web page study. In turn, the study, including the database, is hosted on a personal server on the experimenter's computer. As this would ensure higher privacy of the data by directly sending WebGazer's coordinate predictions, a private connection was chosen for this purpose. The study is accessed via a link that has certain URL parameters determining the participant ID and the condition. Again, due to data protection reasons (keeping data from different sources separately), certain demographic questions such as age, gender, and education level were presented using a Qualtrics study, which assigns a random condition and increments the participant ID. These parameters are passed to WebGazer while the participant is automatically redirected to the experiment page. The Qualtrics dataset is stored separately and it additionally includes the participant consent and demographic information. Once subjects have finished the WebGazer task, they are redirected

to a separate Qualtrics questionnaire on impulsive behavior. The questionnaire serves more purpose for the graduation project of the other student, whose focus is the analysis of accuracy between the two experimental conditions. Figure 3.3 presents the order of task elements schematically.

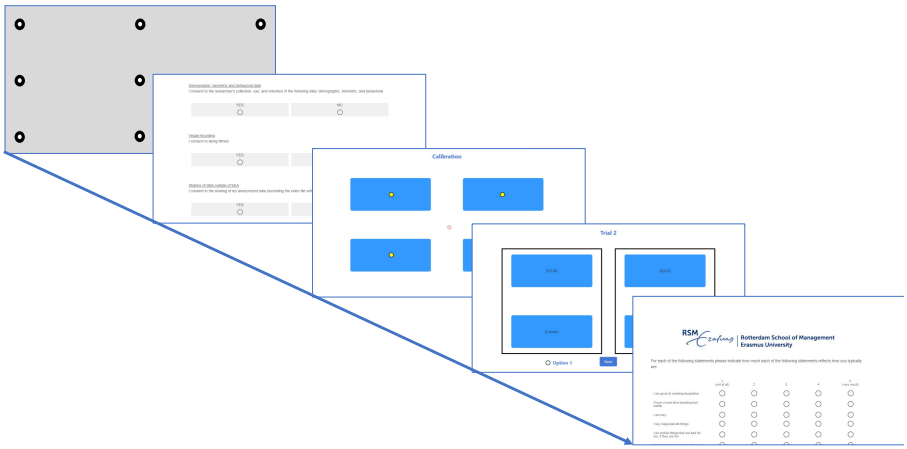


Figure 3.3: The experiment begins with a standard 9-point calibration and validation of the EyeLink. Then, participants are asked for their consent and demographic details. Following the WebGazer task, subjects are asked to fill out a self-controlling behavior questionnaire.

While the WebGazer task is the source for x and y coordinate predictions for this tool, EyeLink predictions are stored as well. The lab uses the EyeLink 1000 eye-tracking hardware, software version 4.56. The tracker has a sampling frequency of 1000 Hz and is set to track the right eye. In the calibration of EyeLink, participants are asked to follow and fixate on targets that appear on the screen. As seen in the top-left corner of Figure 3.3, there are 9 locations for the calibration dot to appear in. It jumps between these points in a randomized order. If the calibration is satisfactory, the predictions are validated and the participant is redirected to the first Qualtrics survey. A desired validation accuracy for the EyeLink method entails two recommendations: an average error value below 0.5 degrees visual angle and a maximum error below 1.0 degrees. The visual angle is a measure of quantifying the size of an object on one's retina. It considers the size of the object and the distance to the retina. For example, if an object has a size of 2 cm on screen and its distance to the retina is approx. 57 cm, the visual angle is the object is 2 degrees. For a more direct interpretation of the results from the methods tested for this thesis, the error is computed as the distance between the validation dots on the screen and the estimated gaze position.

The eye-tracker is positioned between the participant (eye-tracker-to-eye distance is 51cm) and the monitor, and the webcam is placed on top of the monitor (Webcam-to-eye distance is 84cm). The distance from the eye to the top of the screen is 85 cm and to the bottom is 87 cm. This smaller distance to the top is dictated by practical instruction in the EyeLink manual that the participant should be facing the upper half of the screen. Figure 3.4 provides a concrete idea of the geometric characteristics of the setup. Further technical details include the resolution of the screen, 1920 by 1200 pixels, with a refresh rate of 60 Hz and physical size of 52x32.5cm or a diagonal of 61 cm. As uncontrolled light conditions can influence the data, the lab has its windows blackened and additionally covered by blinds. The light source is a cold ceiling light above the participant.

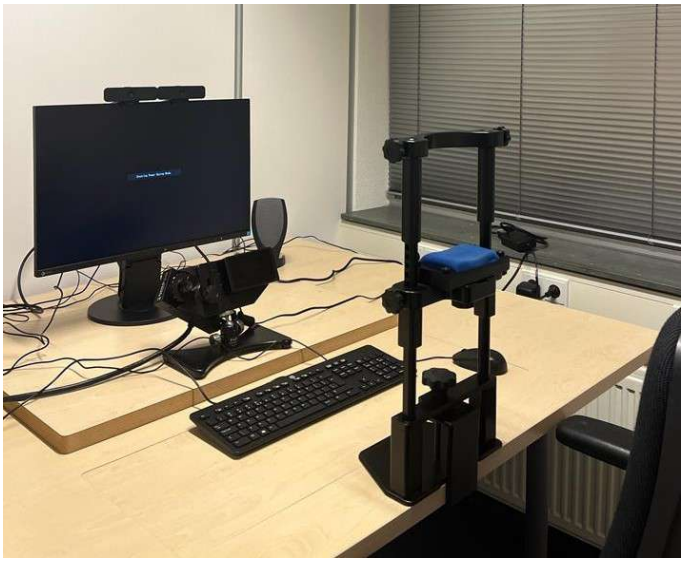


Figure 3.4: The experiment is carried out in the eye-tracking lab of the Erasmus Behavioral Lab. The setup consists of a chair with adjustable height for the participant, a headrest, the EyeLink device, the experiment display, and two identical webcams.

As the video stream from one camera could only be utilized by one application at a time, the setup includes two webcams with the same characteristics, one next to the other with a 12 cm distance between the centers of the camera lenses. The camera used for collecting video data is the same as the one used by WebGazer: a Logitech Webcam C925e, 1080p quality, 16:9 ratio, and 30 fps.

3.2 Dataset and preprocessing

To summarize, there are three main data sources. WebGazer stored x and y coordinate predictions with epoch clock timestamps along with the choices participants make in the task. The frequency of predictions is determined by the webcam, so 30 frames per second. EyeLink collects coordinate predictions, mouse and click data at 1000 Hz, which is a considerably higher frequency than WebGazer and the video. EyeLink has an internal clock and these timestamps are saved together with the predictions. The video of the participants is collected by WebLink, the overarching experiment builder, with timestamps following the same internal EyeLink clock. The timestamps are embedded into the frames as pixel values in the top left corner. Additionally, to prepare a dataset for training any DL algorithms, multiple processing steps are applied to the three data sources. This entails the use of a character recognition model for reading out timestamps from the frames. These are then matched to EyeLink coordinates, discarding a large part of the EyeLink recording. In the next step, the epoch clock stamps of WebGazer are aligned with the EyeLink stamps. This ensures that the same amount of data points that are used for training the network are also available for comparing EyeLink and WebGazer.

The process begins with parsing every video frame using the OpenCV library. Then, a region of interest (top left corner) is defined and the area is transformed to grayscale, followed by the application of a binary threshold to ensure the result is black digits on white background. This pre-processing step increases the success of the Tesseract library, created for optical character recognition. (https://github.com/tesseract-ocr/tesseract) TesseractOCR uses an LSTM architecture to read out characters from images. The algorithm receives small rectangles of the image as input. Multiple networks (a forward and backward LSTM) are used on the pixels in these rectangles and their outputs are stacked afterward. Tesseract then classifies the characters by comparing them to its existing language model. A model parameter asserts that a single uniform block of text should be present in the frame. These decisions were dictated by the fact that this configuration lead to the highest number of frames extracted successfully. After a video has been processed, the faces are cropped out of each frame and saved as key-value pairs with the respective timestamps. Unlike other online eye-tracking solutions which have to apply some technique for searching the face region, in the present study subjects have their head positioned in the same location due to the headrest. Further arguments for this decision is the reduction of the computational complexity of the CNN [69].

Next is the matching between video frames and coordinate predictions. Starting with EyeLink, the data is saved in its proprietary $.edf$ format, which is then transformed to $.asc$ using

the EyeLink data converter tool. The files are then read in R using the "Eyelinker" library. As EyeLink data is commonly analyzed using EyeLink's DataViewer, there aren't well-supported solutions for easy processing in Python and R. However, the Eyelinker library allows access to the raw data with little manual processing. After reading in the .asc file, rows with missing data (e.g., due to out-of-bound gaze coordinates or undetected pupil) are removed. The stamps in the EyeLink recording that match the extracted frame stamps are added to a dictionary containing the frame, timestamps, and two predicted screen coordinates.

To be able to compare these values with the ones predicted by WebGazer, the two different clocks have to be aligned. The EyeLink recording has as one of its first entries a line matching its internal clock with the epoch time clock of the Windows System of the display PC. Therefore, adding this offset to the EyeLink timestamps allows for finding the exact or closest match between the already aligned frames and EyeLink coordinates, and the WebGazer predictions.

When it comes to the dataset collected for this study, it contains data from more participants than previous studies. Compared to publicly available datasets for iris recognition or gaze estimation, the current one has a large amount of data points, albeit not as many as MPIIGaze. In total, 56 participants were invited to the Erasmus Behavioral Lab. The data collection took place in spring 2023 and continued for around two months to reach this sample size. Three of these 56 participants did not have their data recorded: one due to a technical error in the EyeLink Host PC, and the other two due to poor calibration. For these two participants, the experiment was interrupted after the calibration, as the EyeLink tracker was not able to discover their pupils, thus leading to error values above 3 degrees visual angle for multiple validation points even after repeated attempts to calibrate. The resulting 53 participants all have full datasets recorded and are subject to further exploratory data analysis.

These 53 participants have different video lengths, as some people require longer to read the instructions or need time to understand the WebGazer calibration process. After parsing the videos through Tesseract OCR, only 169 frames were not successfully read. For these frames, the model was not able to recognize the full string of digits, instead returning non-numeric characters. Considering that the full dataset after aligning all data sources based on the timings is 447241 data points, not a large portion of the data is lost to the optical character recognition step. In other words, the total duration of the dataset, if considered in terms of a video with 30 fps, is 4.14 hours, or 4.68 minutes on average per person. This period includes only the duration of the WebGazer task.

As mentioned previously, the accuracy measurement was described when introducing the WebGazer calibration procedure. Across participants and conditions (button position fixed or

randomized), the mean accuracy is 83.98 with a $SD = 5.087$. As the manipulation of the button position is subject to another student's graduation project, the conditions will not be separately considered in the models trained here. However, to provide some motivation for collapsing the data across the two conditions, a simple Welch Two sample t-test was performed to test for significant differences in the accuracy estimates between conditions. The number of participants is 27 in the randomized versus 26 in the fixed position condition. The resulting p-value of 0.912 ($n = 53, df = 48.887, 95\%CI = (-3.049, 2.764)$) fails to confirm the hypothesis that the two groups differ in mean validation accuracy. The test is repeated for the EyeLink validation accuracy. With a p-value of 0.746, ($df = 48.406, 95\%CI = (-0.105, 0.075)$), the null hypothesis cannot be rejected. Thus, it is assumed that data from the two conditions can be collapsed to train a neural network and compare the results with WebGazer's predictions.

Another reason why the conditions are not essential to the analysis presented in this study is the random nature of the train-test split. Splitting the whole dataset essentially means treating any differences between the images and predictions in the two conditions as random noise. The split is performed on the full set of 447241 points. Starting from the commonly used train-test split of 75% of the data for the training set, the full set is divided and the mean, standard deviation, and standard error are computed for each division. These statistics are computed for the labels and pixel intensities in the grayscale images. This procedure is repeated until the mentioned statistics were comparable between the train and test set. This aims to ensure that both sets are representative of each other. The final train set contains 335430 points, while the test set has the other 111811.

3.3 Machine Learning baseline

To provide a baseline for comparing the performance of deep learning methods, a common tool from machine learning is used. The purpose of this is also to give the reader an idea of the quality of specific preprocessing methods often used for eye detection in appearance-based eye-tracking literature, namely Haar cascades.

The chosen ML method is XGBoost (Version 1.7.6), widely used for classification and regression tasks due to its computational power. It is a gradient-boosting ensemble method combining the output of multiple decision trees. The latter represent features as nodes and branches as decisions, leading to predictions in the leaf nodes. Learning typically occurs through the minimization of the squared error, however, eye-tracking results are interested in the absolute deviation of a prediction from the real gaze location. So, the mean absolute error is the desired

objective function. Yet, it has mathematical limitations during training, as its second derivative is zero. Therefore, the XGBoost implementation offers an alternative, the Pseudo-Huber loss, used for the training and validation procedure in this paper. This loss function is a twice differentiable approximation of the absolute loss. While the Pseudo-Huber loss function is used for training and validation, the evaluation metric reported in the results is the mean absolute error. This aims to ease comparability with the rest of the results.

Due to limited computational resources, the input had to be changed before running the algorithm. The DL models use full-face grayscale images of size 100x100 pixels which is possible due to the GPU and batch-wise processing. In the case of XGBoost, the available RAM of 25GB ran out almost immediately when trying to prepare the input, even when processing it in batches. This is one reason why the XGBoost is trained on eye patches instead of face images.

The other reason is, as mentioned, to illustrate the abilities of eye detection methods such as Haar cascades. Appearance-based algorithms for eye-tracking often include an initial model-informed search for the eye region. Haar cascades come from the object detection literature in machine learning and can be used in the form of pre-trained cascade classifiers. The OpenCV library offers such models which return boundary points for the discovered eyes. While the common order of processing begins with face detection, the input images in the current project already contain the faces of participants, therefore the eye detection cascades are applied to the images after histogram equalization and image denoising (OpenCV's fastNlMeansDenoising function with recommended parameters: h = 10, a template window size of 7 and a search window size of 21). These two preprocessing techniques are applied to the original image in order to increase the success of the Haar Cascade Classifier. On a side note, without these steps, the resulting dataset after applying Haar Cascades is 8.78% of the size of the original set.

The Haar Cascade function requires manual tuning of parameters, such as the scale factor ($= 1.1$), the minimum number of neighboring pixels classified as part of the eye ($= 5$), and the minimum size of the eyes ($= 10x10$). To gain a better idea of the accuracy of this method, the original size of the complete data set (independent of train or test split) is 447241 while the resulting set contains 75925 data points, which is 17% of the original set. It is also worth noting that independent runs of the eye-detection algorithm lead to slightly varying numbers of eye patches. The preprocessing is separately applied to the train, validation, and test sets ($70 - 10 - 20\%$). Although this method can commonly be used on images with a higher success rate, this is determined by conditions such as illumination and image quality, which varies across previous datasets.

Once the borders are found, the two eye patches are compared in size and one is padded with

black pixels in case it is smaller. Then, the patches are concatenated and compared throughout the whole set. In case of varying sizes, the minimum patch size is taken and the rest of the patches are reshaped to match the minimum. This step ensures uniform input size across the dataset. Last, the input is flattened before starting the training.

Finally, the pixel values are scaled and normalized by the computed mean (62.238) and standard deviation (25.228) of the training set images. Standard scalers are trained separately for the horizontal and vertical targets of the training set and applied to the data. A grid search is implemented over the following parameters: number of boosted trees ($100, 200, 300, 500$), learning rate ($0.1, 0.01, 0.001$), and maximum depth of each tree ($3, 5, 7, 15, 20$). This aims to also inform the starting learning rate for the DL architectures used in the next sections. After finding the best model in terms of mean absolute error, the regressor is trained and validated. Accuracy is measured as the number of predictions falling within 10% of the target value divided by the total number of samples.

3.4 Deep Learning architectures

A commonly suggested network architecture is the convolutional neural network. Partly due to its ability to extract features, a CNN is a good network to apply to the data first. The operation this type of network relies on is the computation of cross-correlation of the pixels in a given neighborhood:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=1}^{C_{in}-1} weight(C_{out_j}, k) * input(N_i, k) \tag{3.2}$$

This operation has many advantages over a feed-forward NN in the context of computer vision. The small filters ensure parameter sharing, locality, and certain translational invariance. In CNNs, fully connected layers are used after the convolutional blocks and consist of all neurons in layer l receiving input from all neurons in layer $l-1$. Other important elements are pooling layers such as the max-pool, which is commonly used in VGG architectures after the ReLu activation function is applied on top of the convolutional layers. Lastly, a dropout operation is often used to increase generalization and prevent overfitting.

3.4.1 Convolutional Neural Network

The first architecture tested to predict screen coordinates from webcam images is inspired by the VGG family. However, unlike the very deep versions, the current one does not extend the full depth. The reason for this is the size of the input images, which is cropped to be 100x100 pixel patches of the face of participants. The cropping was performed automatically when aligning the datasets. A manual check ensured that for every participant their face is still in the patch. Furthermore, the small size of the input images means that the dataset was rather small in size for the amount of data points inside. The input images have a shape of (1, 100, 100), as they only have one channel (grayscale). The tested VGG consists of 6 convolutional layers, each followed by a ReLU activation function, and a Max pool layer after every two convolutional layers. The exact parameters and a visualization of the architecture can be seen in the Appendix. Following the convolutional part of the network, which aims to extract features of the eyes, the outputs are flattened and fed into two fully connected layers, the last of which has two output neurons for each coordinate axis of the screen. A dropout operation was applied after the first two FC layers with the default value of 0.5.

A weight initialization is also part of the training, where for convolutional and FC layers, a Kaiming normal initialization was applied. The parameters of the Kaiming function in Pytorch are set to use the ReLU function and preserve the magnitudes of the variance of the weights in the backward pass (fan_out). Biases are initialized to zero.

For the version of the network tested only on manually cropped eye patches, the convolutional blocks are reduced to 4, as otherwise, the images become too small.

3.4.2 CNN and dilated convolution

For the dilated convolution architecture, the VGG-inspired network and training regimes were kept, the only difference being the change in the convolution operation. The PyTorch conv2d function has a parameter that directly determines the dilation rate and is set to 3. In the case of eye patches as train data, the architecture is the same as the CNN with normal convolution: consisting of four blocks of convolution instead of 6 as the full-face network.

3.4.3 CNN and recurrent modules

To account for the characteristic of the data that gaze movements are a continuous signal, a recurrent component may be beneficial to the learning process. To this end, the original CNN

tested within this project is expanded with an LSTM layer that receives frames output by the CNN part blocks. Considering the temporal aspect of the data is expected to provide better predictions. Besides the addition of this LSTM layer of size 512, the architecture and training regime remains the same as for the CNNs explained before. In the experiment using eye patches as inputs, the convolutional blocks are only 4, the rest of the network remains the same.

3.5 Training procedure

For the VGG-like network, the following training regime is implemented. The optimizer is chosen to be Adam [70] as it combines momentum and adapted learning rate (RMSProp) [71]. The motivation behind the added momentum is the smooth weight update.

As the dataset is large and the computational resources available for analysis were limited, the training is performed on Colab Premium + GPU engines. A batch size of 30 data points is defined. The model is trained for 50 epochs with an adaptive learning rate, which in the beginning is $1e^{-4}$. The function responsible for tracking the L1 loss is $ReduceLROnPlateau$, which reduces the learning rate by a factor of 0.1 if the loss has not changed for 5 epochs in a row. The function is part of $torch.optim.lr\_scheduler$. The MAE cost function is chosen as we are interested in finding out the absolute deviation of a predicted point on the screen from the target. The cost is computed for the horizontal and vertical predictions separately and the sum of the individual losses is backpropagated during training. The accuracy is computed as the number of samples within 10% absolute difference between the true and predicted coordinates divided by the total number of samples.

4

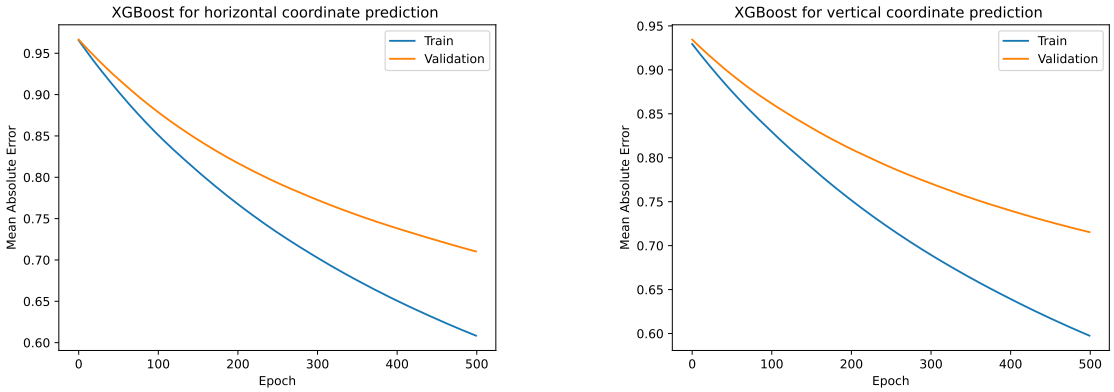
Results and discussion

The present chapter of the thesis is concerned with providing an overview of the experiments introduced in the previous Chapter 3. Each method's training and testing results are presented separately, followed by an evaluation of the performances. The best model in terms of accuracy and time complexity is then compared with the performance of an online tool for low-cost eye tracking, the WebGazer.

4.1 Machine Learning Methods: XGBoost

The previously described grid search was applied in the same way for both the horizontal and vertical coordinate models. The search took 4.21 hours and 4.28 hours respectively and found the best model parameters to be a learning rate of 0.001, the maximum depth of trees of 20, and the training duration - 500 epochs. For horizontal coordinate learning, which took 52 minutes on a train set of 51286 samples, the learning curves can be seen in Figure 4.1. The graph on the right shows the training on the vertical coordinates for a duration of 57 minutes. The plotted error function is the MAE computed on the scaled values.

Making predictions on the test set (18887 images) was very fast for both models: 1.9 seconds for both predictors. The test set MAE is 210.689 pixels for the horizontal coordinates and 174.972 pixels for the vertical ones. The achieved accuracy, however, is 0.007 for the horizontal coordinates and 0.01 for the vertical ones.



(a) Training on horizontal coordinates

(b) Training on vertical coordinates

Figure 4.1: Learning and validation curves of the Mean Absolute Error for the best model resulting from the grid search.

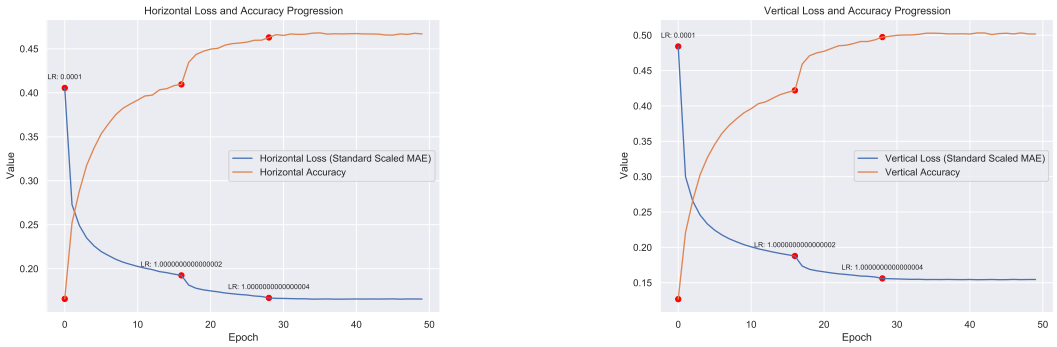
The XGBoost algorithm does not reach an acceptable performance on the test set, despite the evident decrease in training loss. The model is visibly not converging for the number of training epochs. The loss curves are not yet stable. Nevertheless, this result suggests that with longer training, better performance is to be expected, yet even if the performance multiplies compared to the observed values after 500 epochs, it still will not be satisfactory. Nevertheless, there is a sign of learning from the input data. Thus, the results from DL methods are expected to achieve lower error.

4.2 Network performance on full face images

Moving on to the deep-learning methods selected for the analysis of the data, there are three models which are trained both using the images of participants' faces and the manually cropped eye patches. The first set of experiments concerns the full facial images. While these images might contain unnecessary information besides the eye region, it is also possible that slight changes in the head position (within the limits of the chin rest) might have a positive impact on the results. In each subsection, the training curves are plotted and the time requirements are described. The graphs illustrate the accuracy and the MAE computed on the standard scaled targets and predictions. This only serves the purpose of combining both metrics in a single graph. In order to gain a concrete idea of the MAE, the absolute values in terms of pixel distance are presented in Table 4.1. The table contains the test set performance of each model.

4.2.1 Simplified VGG

The simplified VGG architecture is trained for 50 epochs with a mean duration of 373.370 seconds. The total duration is 18668.520 s (5.186 hours). The testing time is 94.106 seconds.



(a) Training on horizontal coordinates

(b) Training on vertical coordinates

Figure 4.2: VGG: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.

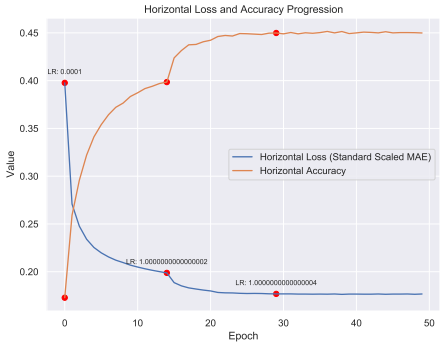
4.2.2 Simplified VGG with dilated convolution

The same VGG-inspired architecture is trained with a simple difference. Instead of a normal convolution, a dilation factor of 3 is added. The training takes 350.208 s per epoch on average, and a total of 17510.39 s (4.864 hours). At test time, the model predicts within 106.962 seconds. The training curves are presented in Figure 4.3

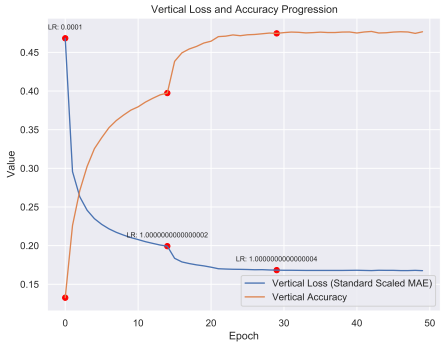
4.2.3 LSTM-VGG

For the last network, the LSTM component is applied to the image frames after they pass through the convolutional blocks. The mean training time per epoch is 1055.516 seconds. The overall training time is 52775.790, or also 14.660 hours. The prediction of test data requires 115.278 seconds.

Out of the described models, the best-performing one on the test set is the CNN with a recurrent component. Overall, the presented learning curves illustrate continuous, but very slow improvement in terms of the L1 loss and the accuracy metric. Furthermore, the metrics plateau after a couple of learning rate decreases. One potential reason for this might be the quality and size of the input images. The resolution and the fact that no preprocessing techniques were

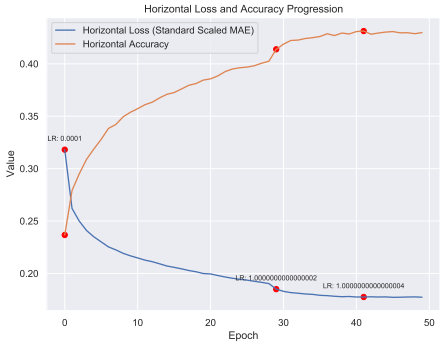


(a) Training on horizontal coordinates

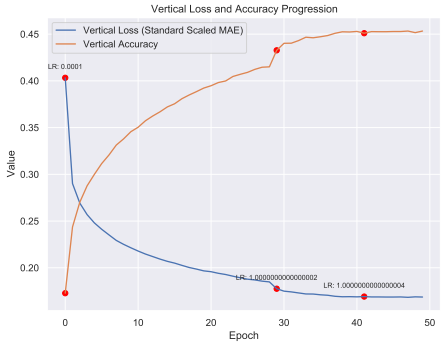


(b) Training on vertical coordinates

Figure 4.3: VGG with dilated convolution: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.



(a) Training on horizontal coordinates



(b) Training on vertical coordinates

Figure 4.4: VGG-LSTM: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.

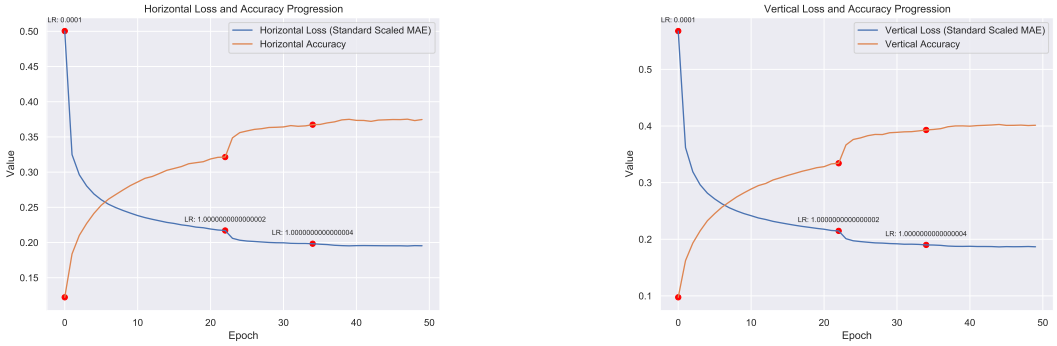
applied to the input data to train this set of models, could further impact the performance of these models.

4.3 Network performance on eye patches

In the next step, further simplified versions of a CNN architecture are applied to smaller images, namely restricted regions of the eyes. As the position of the head remains fixed during the experiment, so does the position of the eyes.

4.3.1 Simplified VGG

The convolutional neural network is simplified to only include four convolutional layers. For this architecture, training on the eye patches continues for 18179.710 seconds, or 5.06 hours (363.592 s on average per epoch). The prediction of test set data takes 89.526 seconds.



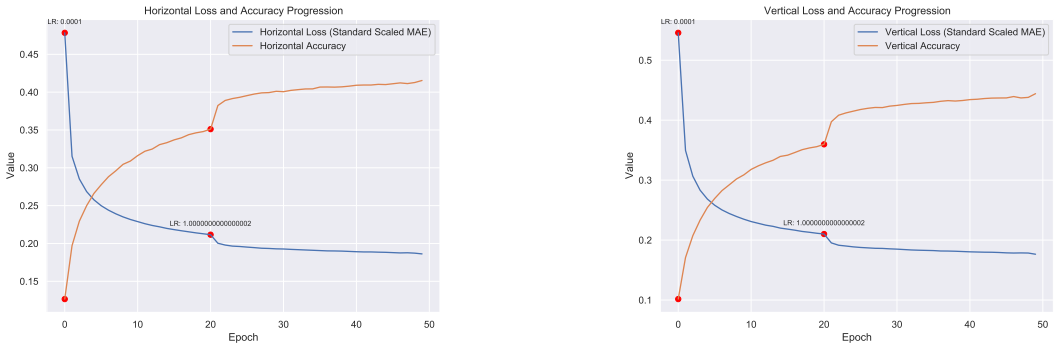
(a) Training on horizontal coordinates

(b) Training on vertical coordinates

Figure 4.5: VGG: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.

4.3.2 Simplified VGG with dilated convolution

The dilated version of the network requires 14755.14 s, or 295.10 s per epoch, to be trained and 94.995 seconds for the test set predictions. The training curves are presented in Figure 4.6



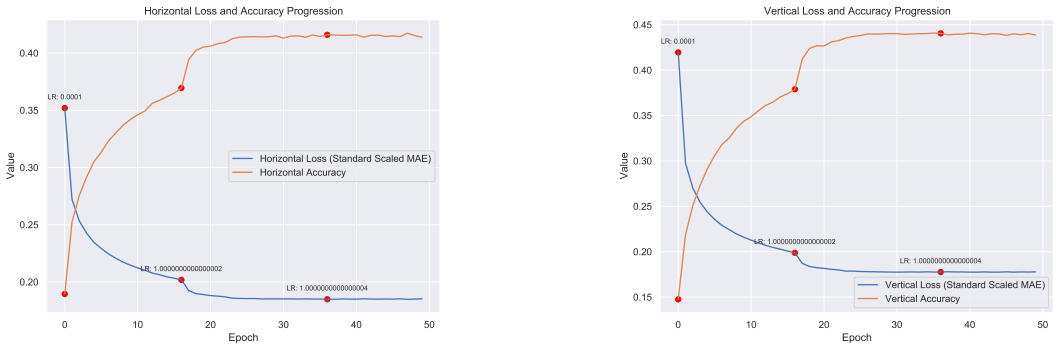
(a) Training on horizontal coordinates

(b) Training on vertical coordinates

Figure 4.6: Dilated VGG: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.

4.3.3 VGG-LSTM

Finally, the CNN with a recurrent module requires 14977.89 s to be trained (299.558 s per epoch). The test time is 69.288 seconds.



(a) Training on horizontal coordinates

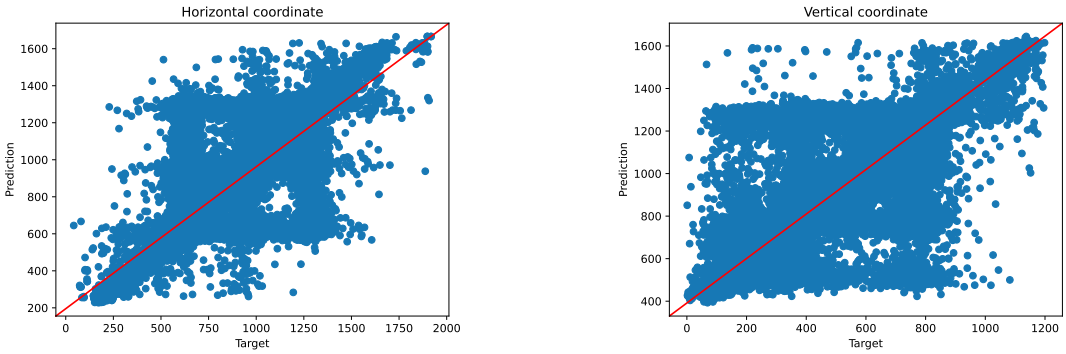
(b) Training on vertical coordinates

Figure 4.7: VGG-LSTM: Learning curves of the Mean Absolute Error and accuracy. Red dots represent changes in the learning rate.

	Horiz. MAE	Vert. MAE	Horiz. Accuracy	Vert. Accuracy
VGG	44.258	35.349	0.6315	0.6741
VGG dilated	44.910	36.312	0.6267	0.6562
VGG-LSTM	43.932	35.052	0.6368	0.6817
VGG-Eyes	48.406	39.034	0.5690	0.5928
VGG-Eyes dilated	47.434	38.323	0.592	0.614
VGG-LSTM-Eyes	46.3842	37.4094	0.6028	0.6269

Table 4.1: Test set metrics for different models trained on face images and manually trimmed eye patches for 50 epochs. The MAE is interpreted as the absolute distance between a target and the predicted value.

Overall, the best-performing model in terms of test loss and accuracy is the LSTM-CNN using the full face as input. However, it is the slowest model to train. The recurrent component was introduced to account for the continuous nature of gaze movements. The results support the idea that this information is relevant for predicting the gaze location on the screen already reported in [57, 19]. The following illustration 4.8 aims to show the alignment between the target and predictions on the test set, separately for the horizontal and vertical coordinates. The graph shows that despite many points laying outside of the diagonal, there is already a level of alignment after 50 epochs of training.



(a) Horizontal coordinate comparison on test set

(b) Vertical coordinate comparison on test set

Figure 4.8: VGG-LSTM: Comparison of test set predictions versus targets

In terms of the results, all models achieve higher accuracy for the vertical coordinate, which could be due to various factors such as the illumination, such as the smaller range of the values compared to the horizontal coordinates. This finding fits with previously reported ones [52], which see a larger error for horizontal predictions. In our case, the finding could also be partially driven by the experimental design. The areas of interest mostly visited by the participant's gaze during the experiment are concentrated in three columns (the information boxes and the accuracy validation dot in the middle). However, the two choice option boxes move around the screen in one experimental condition. Considering the larger distance between the two option boxes, the longer visual search in this horizontal direction might increase the error.

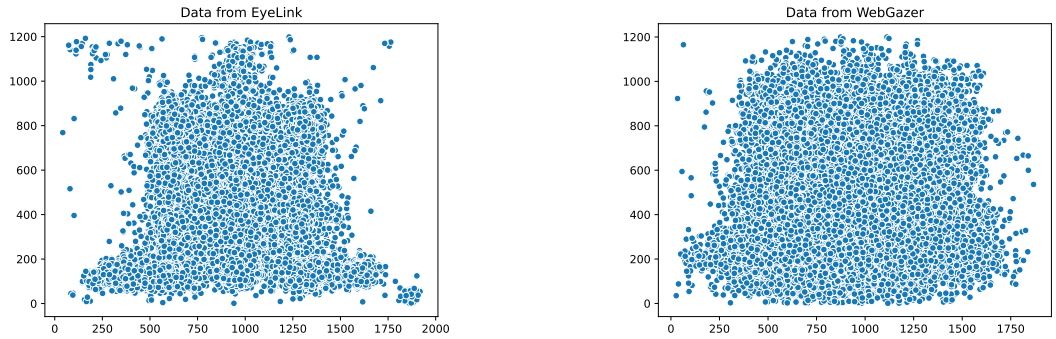
Another relevant point to discuss in relation to previous studies [57] is the performance of the dilation factor in the VGG-like architecture. The dilated convolution was expected to lead to better performance when applied to eye patches, as the authors hypothesize that the operation can detect smaller changes in pixel appearance better than normal convolution. In fact, the dilated version of the CNN performs slightly better than the normal when eye excerpts are used as input. Meanwhile, the performance of the dilation is almost identical to the normal convolution in the case of full-face input (Table 4.1) for the horizontal coordinate. For the vertical coordinates, there is a slight decrease in the performance of the dilated CNN over the normal one. Yet the difference is small, potentially accountable to irrelevant information coming from the head pose.

Lastly, the performance between the models using the face images versus the eye patches should be discussed. Previous studies either report no improvement in results when considering head pose information [15], or refrain from using head pose information as it might contain

unnecessary information [58]. These studies, however, define the head pose information in a different way than the inclusion of the full face as input. The head pose can be coded as a vector of head direction, or information about the face can be extracted previously (e.g. as a face mesh) and then added to the fully-connected layers. In our case, the full-face image does not necessarily convey information regarding the head pose. This is due to the fact that participants use a headrest. Therefore, this slight improvement in prediction (between 1 and 4 pixels on the test set) might be related to some other factors in the larger face image.

4.4 Comparison with WebGazer predictions

Finally, the results are compared to the existing open-source tool for online eye-tracking WebGazer. The following figure is concerned with providing the base comparison between EyeLink and WebGazer visually. The plotted values and the following metrics were calculated for the matched indices of the test set. In terms of MAE, the horizontal coordinates differ by 144.75 pixels when comparing the full datasets. The vertical MAE amounts to 146.34 pixels. In both cases, the error in the WebGazer predictions is large. Compared to the performance of the available neural networks trained within this project, WebGazer is less accurate.



(a) Training on horizontal coordinates

(b) Training on vertical coordinates

Figure 4.9: Visual comparison of EyeLink and WebGazer predictions on the test set. The EyeLink data is plotted on the left.

Lastly, to provide the reader with a concrete idea of the achieved performance improvement, a website is chosen. In many eye-tracking experiments, there are predefined regions of interest (ROI), which can differ in size. So, it is important for researchers to know the expected accuracy of different methods before choosing one for their analysis. The visual experiment aims to show the difference between what WebGazer is capable of predicting versus the developed

methods. With a test set MAE of 146.34 pixels for a vertical coordinate prediction, WebGazer could differentiate whether the user is looking at the logo of a university course (central panel with course descriptions) or at the course name below the logo. However, if the researcher is interested in the placement of the course names in the dashboard panel on the left side, WebGazer would not be able to provide accurate information. One course name section has a height between 47-68 pixels approximately. As the tested neural networks have a vertical MAE of 35.3 up to 39 pixels, they would be able to differentiate between two neighboring course name sections in the dashboard panel.

.

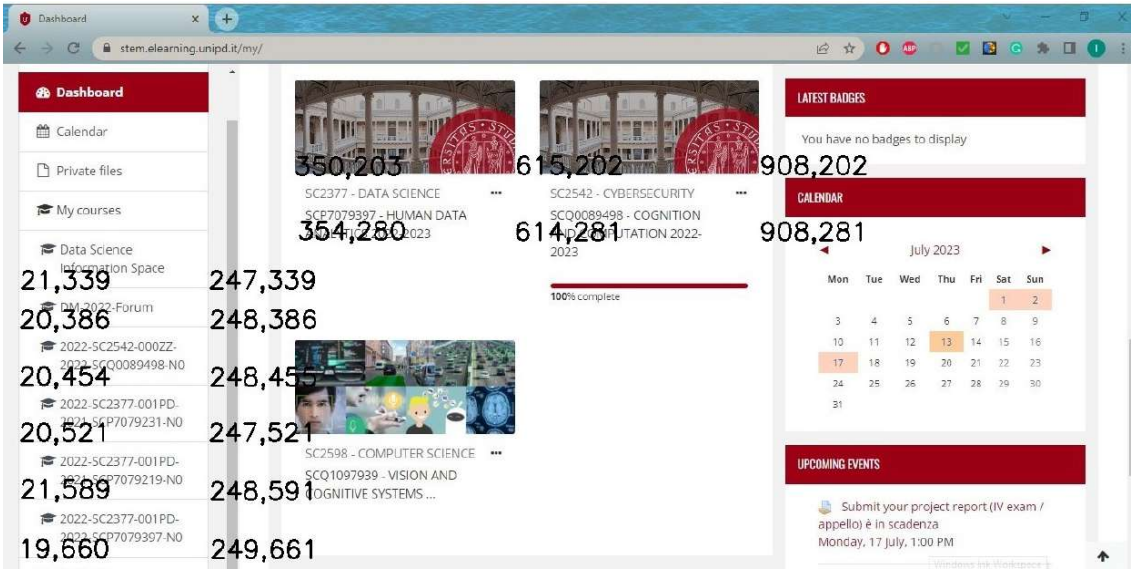


Figure 4.10: Screenshot of an example website with overlaid coordinates in terms of pixels. The coordinate for a point is placed to the top right of the dot itself.

5 Conclusion

5.1 Summary of results

In sum, the results confirm multiple previous findings. Most importantly, the idea is that a neural network can be used as a feature extractor and a way to learn to predict screen coordinates. Including information regarding the head position leads to lower error. Furthermore, the long short-term memory component increases the accuracy of the predictions, and with further engineering prior to its deployment, it is a reasonable model to adopt in web eye-tracking research.

On another hand, the experiment with Haar Cascades illustrates the method's slowness and dependency on various characteristics of the data. The resolution of the images, the illumination, distance from the camera to the face, can all be detrimental to the success of this algorithm. An additional review study of datasets and their respective conditions is necessary to quantify their effects on eye-detection algorithms. The use of online tools such as WebGazer remains under question, as its accuracy is also influenced by environmental conditions.

5.2 Relevance and applications

As described in the introduction, web eye-tracking algorithms are a tool that requires further improvement. The applications of such methods in usability, marketing, and behavioral re-

search are various. The example provided at the end of the previous section gives an idea of the achieved improvement over existing methods. Researchers interested in evaluating the placement of ads or certain website elements can rely on web eye-tracking models to a certain extent. The models, however, can for now be applied directly to a dataset that already has detected face or eye regions. This is an aspect that limits the usability, yet does not refute the fact that such prediction accuracy is achievable with relatively simple neural networks.

The algorithms presented in the current study rely on a new dataset, collected for the purpose of using webcam images as input and EyeLink predictions as target values. One possible application of this paradigm is the chance for researchers to first collect a small sample in a lab. Then, training the aforementioned models, or using transfer learning can improve the predictions in the context of the specific experiment. Finally, the optimized trained model can be deployed for the researcher's experiment online.

Pre-testing characteristics of stimuli (such as the size and valence) used for an online experiment with a large at-home sample is a plausible use case of the algorithms. Although further validation is necessary before this is possible, the results are promising.

Overall, the findings support the idea that eye-tracking with a webcam is possible. Nevertheless, there are limitations to the current study that argue for the difficulty of the task.

5.3 Limitations and suggestions

On one hand, the issue of varying experimental conditions is not solved. The algorithms need to be validated for different head poses, illumination conditions, and camera resolutions.

As the head remains in a fixed position in the current study, no object detection algorithms had to be applied to restrict the search area for the eyes in the image. As a continuation of the research topic, two paths can be laid out. In a lab experiment without the headrest, participants can be instructed to refrain from sudden head movements in one condition. The other condition would see subjects freely moving their head. The EyeLink algorithm is capable of detecting the pupil within a certain range of movement (headbox). This study would allow using the data from the first condition as training data for a NN, while the free head movements are used to test whether the results can be generalized to different head poses. An alternative way for validating the accuracy of neural networks for head movements is by relying on existing datasets. Some contain images of different head poses that can be used for testing the presented NNs. The issue is that these datasets do not contain the target labels for screen coordinates. Therefore, this second suggestion would instead imply comparing the achieved predictions with pre-

viously published results from training certain algorithms on such datasets.

Similarly, the illumination conditions remained constant throughout the data collection. A follow-up study can proceed in one of three directions. On one hand, one can attempt to collect a new dataset in the lab with the same setup, yet with a desktop light that is continuously moved around to introduce a traceable variation in the illumination. This option, however, directly affects the accuracy of the EyeLink predictions as well. The machine is sensitive to changes in the light conditions. Thus, the reliability of the target data would suffer and its comparison with NN predictions would not be as accurate. An artificial way to test the adaptability of NN algorithms to illumination changes would be the introduction of image transformations. The simplest solution is Adobe Photoshop which has tools for changing the light source in an image. On a larger scale, DL methods can be used for relighting datasets: Deep Relighting Network [72] is an example of this. This way the existing dataset collected in the lab can be transformed to create images with different illumination conditions without affecting the accuracy of EyeLink predictions.

A limitation stemming from the data collection process in the lab is the sharing of the room with another set of students. Throughout the collection period, the students repeatedly changed the positioning of the eye-tracker. The geometric setup distances were initially recorded and then continuously measured and readjusted back to the original. Nevertheless, it is possible that certain angles of the camera were not the same as it is unknown if any untracked factors were modified by the other group of students.

Moving on to the neural architectures themselves, a previous effort is to combine different inputs, such as the extracted information regarding the head position, with the one from the eyes. Having a multiple-channel network is something that can be tested as a follow-up to this project: applying the convolutional parts to the face and eye patches and concatenating these prior to the fully-connected layers might lead to higher prediction accuracy as hypothesized by previous studies.

Lastly, the current study does not present a reason behind the low performance of the Haar Cascades in the XGBoost algorithm. The datasets previously tested for eye detection algorithms vary in head positions, lighting, and image quality to name a few factors. Additionally, many algorithms for eye detection require manual tuning of parameters. Therefore, providing a reliable prediction of the success rate for the Haar Cascade algorithm requires further exploration of the available datasets.

5.4 Concluding words

In sum, the topic of online eye-tracking gains importance with every new DL model for eye detection, and also with every new piece of legislation concerning the use of artificial intelligence in websites. From this point of view, the results that gaze location on screen can not yet be accurately predicted in an uncontrolled home environment, gives institutions such as the EU time to develop appropriate guidelines for data collectors and also for the users of websites. Although the presented methods perform well on the presented novel dataset, their generalizability to more flexible environmental conditions, such as an at-home setup, is not yet validated. These issues are present in most eye-tracking methods, both in the lab and at home. The many categories of possible error sources require an extensive review and quantification. Ideally, with the extensive description of the experimental setup, other researchers are capable of reviewing findings in a reproducible way. Nevertheless, there is a certain level of usability of such tools for webcam eye-tracking that can allow researchers to expand their efforts to collect reliable data outside of the lab.

References

[1] A. Inc. Augmented reality - apple. [Online]. Available: https://www.apple.com/augmented-reality/

[2] V. Clay, P. König, and S. Koenig, "Eye tracking in virtual reality," *Journal of eye movement research*, vol. 12, no. 1, 2019.

[3] K. Tamura, R. Choi, and Y. Aoki, "Unconstrained and calibration-free gaze estimation in a room-scale area using a monocular camera," *IEEE Access*, vol. 6, pp. 10 896–10 908, 2017.

[4] D. Bridges, A. Pitiot, M. R. MacAskill, and J. W. Peirce, "The timing mega-study: Comparing a range of experiment generators, both lab-based and online," *PeerJ*, vol. 8, p. e9414, 2020.

[5] J. R. De Leeuw, "jspsych: A javascript library for creating behavioral experiments in a web browser," *Behavior research methods*, vol. 47, pp. 1–12, 2015.

[6] J. Peirce, J. R. Gray, S. Simpson, M. MacAskill, R. Höchenberger, H. Sogo, E. Kastman, and J. K. Lindeløv, "Psychopy2: Experiments in behavior made easy," *Behavior research methods*, vol. 51, pp. 195–203, 2019.

[7] K. B. Sheehan, "Crowdsourcing research: data collection with amazon's mechanical turk," *Communication Monographs*, vol. 85, no. 1, pp. 140–156, 2018.

[8] A. M. Turner, T. Engelsma, J. O. Taylor, R. K. Sharma, and G. Demiris, "Recruiting older adult participants through crowdsourcing platforms: Mechanical turk versus prolific academic," in *AMIA Annual Symposium Proceedings*, vol. 2020. American Medical Informatics Association, 2020, p. 1230.

[9] A. L. Yarbus and A. L. Yarbus, "Eye movements during perception of complex objects," *Eye movements and vision*, pp. 171–211, 1967.

[10] R. d. O. J. dos Santos, J. H. C. de Oliveira, J. B. Rocha, and J. d. M. E. Giraldi, "Eye tracking in neuromarketing: a research agenda for marketing studies," *International journal of psychological studies*, vol. 7, no. 1, p. 32, 2015.

[11] G. van Loon, F. Hermsen, and M. Naber, "Predicting product preferences on retailers' web shops through measurement of gaze and pupil size dynamics," *Journal of Cognition*, vol. 5, no. 1, 2022.

[12] J. Kongmanon and P. Petison, "What do you see and what do you recall?: Using eye tracking to understand product placement," *Cogent Business & Management*, vol. 9, no. 1, p. 2120263, 2022.

[13] H. Gao and E. Kasneci, "Eye-tracking-based prediction of user experience in vr locomotion using machine learning," in *Computer Graphics Forum*, vol. 41, no. 7. Wiley Online Library, 2022, pp. 589–599.

[14] J. C.-Y. Sun and K. Y.-C. Hsu, "A smart eye-tracking feedback scaffolding approach to improving students' learning self-efficacy and performance in a c programming course," *Computers in Human Behavior*, vol. 95, pp. 66–72, 2019.

[15] I. Rakhmatulin and A. T. Duchowski, "Deep neural networks for low-cost eye tracking," *Procedia Computer Science*, vol. 176, pp. 685–694, 2020.

[16] K. Holmqvist, S. L. Örbom, I. T. Hooge, D. C. Niehorster, R. G. Alexander, R. Andersson, J. S. Benjamins, P. Blignaut, A.-M. Brouwer, L. L. Chuang *et al.*, "Eye tracking: empirical foundations for a minimal reporting guideline," *Behavior research methods*, vol. 55, no. 1, pp. 364–416, 2023.

[17] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, pp. 16 495–16 519, 2017.

[18] Y. Cheng, H. Wang, Y. Bao, and F. Lu, "Appearance-based gaze estimation with deep learning: A review and benchmark," *arXiv preprint arXiv:2104.12668*, 2021.

[19] S. Ghosh, A. Dhall, M. Hayat, J. Knibbe, and Q. Ji, "Automatic gaze analysis: A survey of deep learning based approaches," *arXiv preprint arXiv:2108.05479*, 2021.

[20] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays, "Webgazer: Scalable webcam eye tracking using user interactions," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*, 2016.

[21] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 478–500, 2009.

[22] J. Merchant, R. Morrissette, and J. L. Porterfield, "Remote measurement of eye direction allowing subject motion over one cubic foot of space," *IEEE transactions on biomedical engineering*, no. 4, pp. 309–317, 1974.

[23] E. D. Guestrin and M. Eizenman, "General theory of remote gaze estimation using the pupil center and corneal reflections," *IEEE Transactions on biomedical engineering*, vol. 53, no. 6, pp. 1124–1133, 2006.

[24] R. S. Hessels and I. T. Hooge, "Eye tracking in developmental cognitive neuroscience–the good, the bad and the ugly," *Developmental cognitive neuroscience*, vol. 40, p. 100710, 2019.

[25] I. T. Hooge, D. C. Niehorster, R. S. Hessels, D. Cleveland, and M. Nyström, "The pupil-size artefact (psa) across time, viewing direction, and different eye trackers," *Behavior Research Methods*, pp. 1–21, 2021.

[26] D. C. Niehorster, R. Zemblys, T. Beelders, and K. Holmqvist, "Characterizing gaze position signals and synthesizing noise during fixations in eye-tracking data," *Behavior Research Methods*, vol. 52, pp. 2515–2534, 2020.

[27] K. Schlegelmilch and A. E. Wertz, "The effects of calibration target, screen location, and movement type on infant eye-tracking data quality," *Infancy*, vol. 24, no. 4, pp. 636–662, 2019.

[28] R. S. Hessels, R. Andersson, I. T. Hooge, M. Nyström, and C. Kemner, "Consequences of eye color, positioning, and head movement for eye-tracking data quality in infant research," *Infancy*, vol. 20, no. 6, pp. 601–633, 2015.

[29] K. W. Choe, R. Blake, and S.-H. Lee, "Pupil size dynamics during fixation impact the accuracy and precision of video-based gaze estimation," *Vision research*, vol. 118, pp. 48–59, 2016.

[30] D. W. Hansen and A. E. Pece, "Eye tracking in the wild," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 155–181, 2005.

[31] K. Holmqvist and R. Andersson, "Eye tracking: A comprehensive guide to methods," *paradigms and measures*, 2017.

[32] R. Engbert, L. O. Rothkegel, D. Backhaus, and H. A. Trukenbrod, "Evaluation of velocity-based saccade detection in the smi-etg 2w system," *Technical report, Allgemeine und Biologische Psychologie, Uni-versität Potsdam, March*, 2016.

[33] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer, *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011.

[34] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Adaptive linear regression for appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 10, pp. 2033–2046, 2014.

[35] Z. Zhu, Q. Ji, and K. P. Bennett, "Nonlinear eye gaze mapping function estimation via support vector regression," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1. IEEE, 2006, pp. 1132–1135.

[36] O. Williams, A. Blake, and R. Cipolla, "Sparse and semi-supervised visual mapping with the s^ 3gp," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1. IEEE, 2006, pp. 230–237.

[37] B. Noris, K. Benmachiche, and A. G. Billard, "Calibration-free eye gaze direction detection with gaussian processes," in *International Conference on Computer Vision Theory and Applications*, vol. 2. SCITEPRESS, 2008, pp. 611–616.

[38] C. L. L. Jerry and M. Eizenman, "Convolutional neural networks for eye detection in remote gaze estimation systems," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1. Citeseer, 2008.

[39] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4511–4520.

[40] A. Fraser, K. Hurman, M. Robinson, M. Duta, G. Scerif *et al.*, "Automated gaze direction scoring from videos collected online through conventional webcam." 2021.

[41] A. L. Anwyl-Irvine, T. Armstrong, and E. S. Dalmaijer, "Mouseview. js: Reliable and valid attention tracking in web-based experiments using a cursor-directed aperture," *Behavior research methods*, pp. 1–25, 2021.

[42] M. C. Chen, J. R. Anderson, and M. H. Sohn, "What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing," in *CHI'01 extended abstracts on Human factors in computing systems*, 2001, pp. 281–282.

[43] J. Huang, R. White, and G. Buscher, "User see, user point: gaze and cursor alignment in web search," in *Proceedings of the sigchi conference on human factors in computing systems*, 2012, pp. 1341–1350.

[44] Q. Guo and E. Agichtein, "Towards predicting web searcher gaze position from mouse movements," in *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, 2010, pp. 3601–3606.

[45] A. Papoutsaki, "Scalable webcam eye tracking by learning from user interactions," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, 2015, pp. 219–222.

[46] A. Steffan, L. Zimmer, N. Arias-Trejo, M. Bohn, R. D. Ben, M. A. Flores-Coronado, L. Franchin, I. Garbisch, C. Grosse Wiesmann, J. K. Hamlin *et al.*, "Validation of an open source, remote web-based eye-tracking method (webgazer) for research in early childhood," 2023.

[47] K. Wisiecka, K. Krejtz, I. Krejtz, D. Sromek, A. Cellary, B. Lewandowska, and A. Duchowski, "Comparison of webcam and remote eye tracking," in *2022 Symposium on Eye Tracking Research and Applications*, 2022, pp. 1–7.

[48] K. Semmelmann and S. Weigelt, "Online webcam-based eye tracking in cognitive science: A first look," *Behavior Research Methods*, vol. 50, pp. 451–465, 2018.

[49] B. Chouinard, K. Scott, and R. Cusack, "Using automatic face analysis to score infant behaviour from video collected online," *Infant Behavior and Development*, vol. 54, pp. 1–12, 2019.

[50] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike, "An incremental learning method for unconstrained gaze estimation." in *ECCV (3)*. Citeseer, 2008, pp. 656–667.

[51] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, "Gaze estimation from eye appearance: A head pose-free method via eye image synthesis," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3680–3693, 2015.

[52] N. Zdarsky, S. Treue, and M. Esghaei, "A deep learning-based approach to video-based eye tracking for human psychophysics," *Frontiers in human neuroscience*, vol. 15, p. 685830, 2021.

[53] S. Nanayakkara and R. Meegama, "A review of literature on iris recognition," *International Journal of Research*, vol. 9, pp. 106–120, 2020.

[54] S. Minaee, A. Abdolrashidiy, and Y. Wang, "An experimental study of deep convolutional features for iris recognition," in *2016 IEEE signal processing in medicine and biology symposium (SPMB)*. IEEE, 2016, pp. 1–6.

[55] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Mpiigaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 162–175, 2017.

[56] ——, "Appearance-based gaze estimation in the wild," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4511–4520.

[57] Z. Chen and B. E. Shi, "Appearance-based gaze estimation using dilated-convolutions," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part VI*. Springer, 2019, pp. 309–324.

[58] T. Fischer, H. J. Chang, and Y. Demiris, "Rt-gene: Real-time eye gaze estimation in natural environments," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 334–352.

[59] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[60] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "It's written all over your face: Full-face appearance-based gaze estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 51–60.

[61] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[62] S. Park, E. Aksan, X. Zhang, and O. Hilliges, "Towards end-to-end video-based eye-tracking," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer, 2020, pp. 747–763.

[63] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[64] P. Linardos, E. Mohedano, J. J. Nieto, N. E. O'Connor, X. Giro-i Nieto, and K. McGuinness, "Simple vs complex temporal recurrences for video saliency prediction," *arXiv preprint arXiv:1907.01869*, 2019.

[65] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[66] P. Kellnhofer, A. Recasens, S. Stent, W. Matusik, and A. Torralba, "Gaze360: Physically unconstrained gaze estimation in the wild," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6912–6921.

[67] Y. Cheng and F. Lu, "Gaze estimation using transformer," *arXiv preprint arXiv:2105.14424*, 2021.

[68] J. L. Kröger, O. H.-M. Lutz, and F. Müller, "What does your gaze reveal about you? on the privacy implications of eye tracking," *Privacy and Identity Management. Data for Better Living: AI and Privacy: 14th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2. 2 International Summer School, Windisch, Switzerland, August 19–23, 2019, Revised Selected Papers 14*, pp. 226–241, 2020.

[69] A. S. Al-Waisy, R. Qahwaji, S. Ipson, S. Al-Fahdawi, and T. A. Nagem, "A multi-biometric iris recognition system based on a deep learning approach," *Pattern Analysis and Applications*, vol. 21, pp. 783–802, 2018.

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[71] A. G. Govindaswamy, E. Montague, D. S. Raicu, and J. Furst, "Cnn as a feature extractor in gaze recognition," in *2020 3rd Artificial Intelligence and Cloud Computing Conference*, 2020, pp. 31–37.

[72] L.-W. Wang, W.-C. Siu, Z.-S. Liu, C.-T. Li, and D. P. Lun, "Deep relighting networks for image light source manipulation," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 550–567.

Acknowledgments

The acknowledgment section regards people that made this project possible, beginning with my supervisor, Antonia, who arranged so many opportunities for me to explore projects at Erasmus University and brought me onto the current one with Sebastian. I always had the feeling that they relied on me to do my best effort and trusted me in all steps of the process. I want to thank the team of the Erasmus Behavioral Lab for always being there to help with any questions and issues before and during the data collection.

A

First appendix

A.1 Characteristics of the participant sample

A.2 Neural Network Architectures

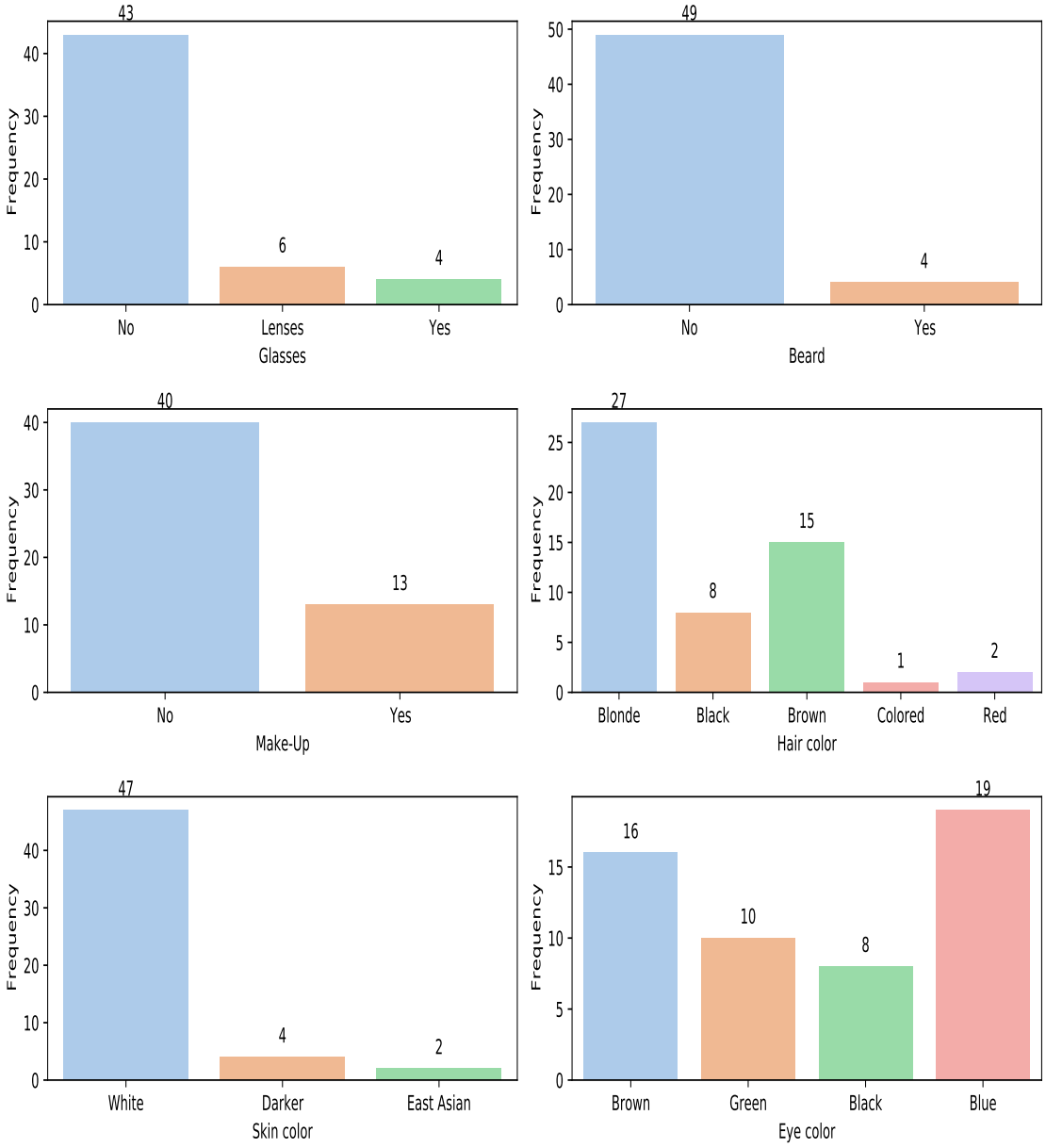


Figure A.1: Extra variables observed for the participant sample.

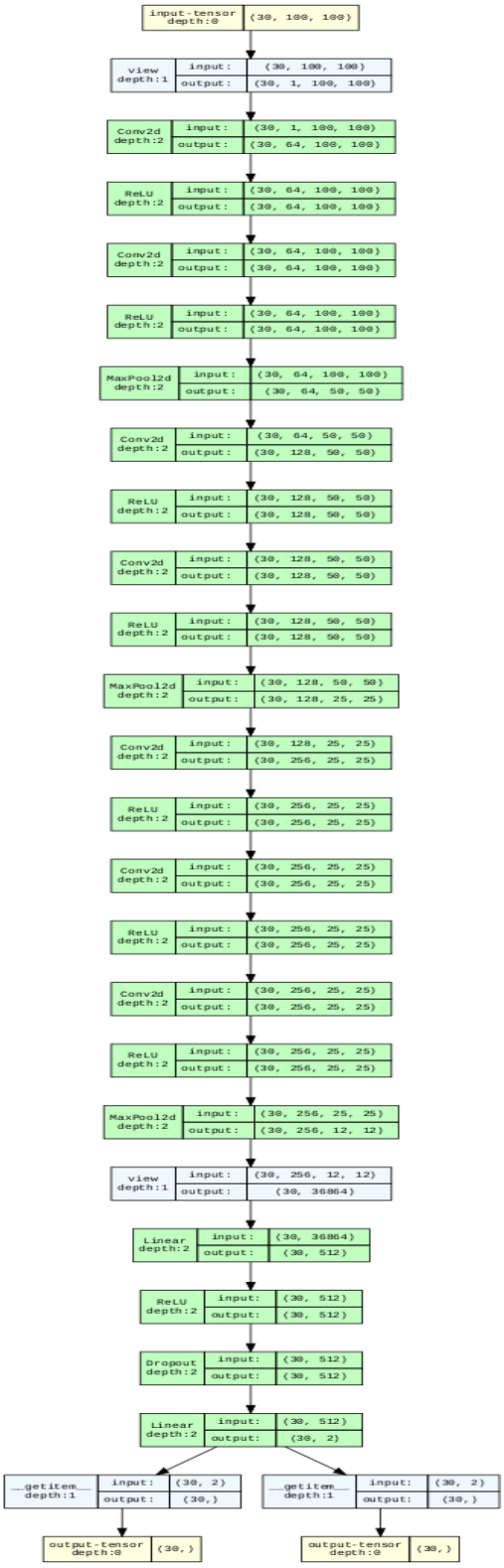


Figure A.2: Simplified VGG architecture with normal convolution.

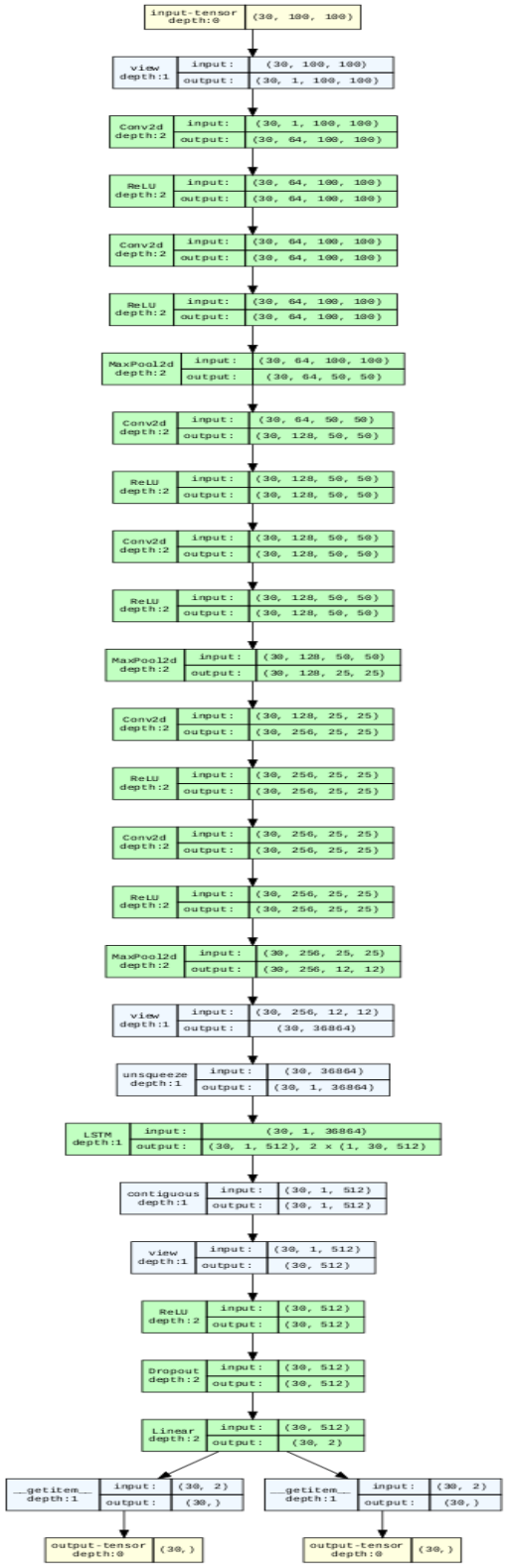


Figure A.3: Simplified VGG architecture with normal convolution and LSTM module.