applied sciences

MDPI

Article

Gaze Tracking Using an Unmodified Web Camera and Convolutional Neural Network

Mohd Faizan Ansari, Pawel Kasprowski * and Marcin Obetkal

Department of Applied Informatics, Silesian University of Technology, 44-100 Gliwice, Poland; mfansari2395@gmail.com (M.F.A.); marcin.obetkal@gmail.com (M.O.)

* Correspondence: pawel.kasprowski@polsl.pl

Abstract: Gaze estimation plays a significant role in understating human behavior and in human–computer interaction. Currently, there are many methods accessible for gaze estimation. However, most approaches need additional hardware for data acquisition which adds an extra cost to gaze tracking. The classic gaze tracking approaches usually require systematic prior knowledge or expertise for practical operations. Moreover, they are fundamentally based on the characteristics of the eye region, utilizing infrared light and iris glint to track the gaze point. It requires high-quality images with particular environmental conditions and another light source. Recent studies on appearance-based gaze estimation have demonstrated the capability of neural networks, especially convolutional neural networks (CNN), to decode gaze information present in eye images and achieved significantly simplified gaze estimation. In this paper, a gaze estimation method that utilizes a CNN for gaze estimation that can be applied to various platforms without additional hardware is presented. An easy and fast data collection method is used for collecting face and eyes images from an unmodified desktop camera. The proposed method registered good results; it proves that it is possible to predict the gaze with reasonable accuracy without any additional tools.

Keywords: gaze tracking; convolutional neural network; human computer interaction

Citation: Ansari, M.F.; Kasprowski, P.; Obetkal, M. Gaze Tracking Using an Unmodified Web Camera and Convolutional Neural Network. *Appl. Sci.* **2021**, *11*, 9068. https://doi.org 10.3390/app11199068

Academic Editors: Sławomir Nowaczyk, Mohamed-Rafik Bouguelia and Hadi Fanaee

Received: 13 August 2021
Accepted: 23 September 2021
Published: 29 September 2021

1. Introduction

Gaze estimation has long been recognized as an important research topic since it has strong real-world applications, for instance, in human–computer interfaces [1,2], gaze-based interfaces [3,4], virtual reality [5,6], health care [7], behavioral analysis [8], and communication skills [9]. Therefore, gaze estimation has become a well-established research topic in computer vision, especially in human–computer interaction (HCI) [10,11]. Early gaze estimation techniques required strict conditions to calculate gaze points, such as stabilizing a person's head and controlling light conditions. These restraints bounded the applications to relatively restricted laboratory environments. For applying gaze estimation in natural environments, we have proposed many methods to mitigate these restraints and have uplifted gaze estimation towards being calibration-free, without person-specific and light independent gaze tracking [12,13].

Most of the available eye-tracking methods are expensive as they require specialized commercial hardware and software. With the growing number of applications that use eye-tracking, there is also a growing need for cheap and ubiquitous methods to obtain information about human gaze coordinates. Therefore, methods utilizing unmodified cameras currently built into almost every computer setup can play an essential role in popularizing eye-tracking since they are easily available and do not require additional costs.

Unmodified web cameras have been used in this paper to assess the possibility of utilizing them to estimate gaze coordinates. Nowadays, almost every computer is equipped with a camera that potentially may be used for gaze estimation. The main problem is that unmodified web cameras work only in visible light (and most have infrared filters). The next problem is that they typically have a wide-angle lens with limited (if any) possibilities

to zoom. The result is that the image of the eyes is small (low resolution) and is highly dependent on light conditions. That is why using an unmodified camera for eye tracking is a big challenge.

Algorithms that use unmodified cameras are different from classical eye tracking algorithms, which utilize infrared cameras and an additional infrared light source that produces a glint (reflection from eyeball) to track the eyes. Although cameras with infrared light can detect eyes with high accuracy, they add an additional cost to the eye-tracking. In our research, we used standard unmodified cameras to detect face and eye regions, which reduces the eye-tracking cost significantly. Estimating the gaze coordinate from the unmodified cameras is a difficult task as compared to infrared cameras because, in unmodified cameras, the pupil cannot be located clearly all the time. In addition, different illumination conditions, head position, and eye angles can make it hard to estimate the person's gaze.

In this research, the image taken by an unmodified web camera was fed to the state-of-the-art CNN network to produce the information about the region on a screen where the person was looking. The CNN network was trained and tested with several different architectures of the CNN network with several combinations of face, both eyes, and single eyes to estimate the gaze coordinate from the camera images. Our experimental findings show that it is possible to predict human gaze points from an unmodified camera with reasonable accuracy.

The main contribution of the paper is creating and testing a dataset that was collected in the wild by people sitting at their own laptops without any supervision. This dataset was used to train different networks. The low quality of the recordings made it challenging to assess the exact coordinates of the gaze points, but it was shown that it is possible to estimate with high accuracy the 20 screen areas the person is looking at.

The rest of the paper is structured as follows: Section 2 presents related work for the gaze estimation. Section 3 presents our dataset and the different network architectures used in this study. Section 4 presents the experimental results of this study. Finally, Section 5 includes concluding remarks and a summary and suggests future directions in gaze estimation.

2. Related Work

Gaze estimation has been studied for decades due to its variety of applications in different domains. The traditional approach is to record an image of an eye, use image processing techniques to find a pupil, sclera, light source reflections (Purkinje images), and then use their locations to estimate gaze. The main problem with this approach is that it requires a high-quality image of the eye. Therefore, professional video-based (VOG) eye trackers are equipped with high-quality cameras and have many restrictions for participants (such as using a chin-rest or bite-bar) to ensure that an eye image is visible in the camera.

Gaze estimation methods can be categorized into model-based, also known as feature-based, and appearance-based approaches [14]. Model-based methods make use of 3D geometric eye models to estimate gaze direction [15,16]. They usually explore the characteristics of the human eye to identify a set of distinctive features of the eyes. The limbus, pupil (dark/bright pupil images), and cornea reflections are common features used to estimate gaze direction. Furthermore, these methods also require infrared light sources together with high-resolution cameras. Although this approach can accurately estimate gaze direction, it requires specialized hardware that limits its range of application. If the iris contour alone is used to detect the line of sight, the need for specialized eyewear can be relaxed [17]. In conclusion, these methods are effective for short-distance scenarios in which high-resolution observations are available, but their effectiveness in mid-distance scenarios is unclear.

Most commercial eye trackers ordinarily depend on active illumination, i.e., pupil center corneal reflections (PCCR) based eye-tracker uses infrared illumination, which

provides high accuracy [18]. However, these types of eye trackers are expensive and generally used in a laboratory setting together with a chin rest or bite bar that stabilizes the head. To reduce the restriction on head movement, the authors proposed different solutions [19,20]. However, these eye trackers do not work well in outdoor situations because of the sun's infrared radiation.

Appearance-based methods have achieved great success in gaze estimation in recent years due to their ability to learn the gaze direction directly from the image of the eye without any feature extraction. These methods require only off-the-shelf cameras, which are normally available, that simplify capturing gaze coordinates in indoor and outdoor conditions. However, attaining a high accuracy is challenging due to many factors such as changes in appearance, lighting conditions, and head pose [21]. With the success of convolutional neural networks (CNNs) in computer vision, researchers have adopted using CNN in appearance-based gaze estimation, which has reduced estimation errors [22]. Trained with high quality diverse real and synthesized datasets covering a wide range of variation [22,23], a deep CNN network can learn to compensate much of the variability present in the dataset [13,24].

Deep learning is a powerful technique that has developed fast and is widely used in many applications [25,26] including computer vision [27] and gaze estimation [28,29]. Furthermore, there have been several open-source datasets for gaze estimation from the research community, for instance, MPIIGaze [22], GazeCapture [30], TabletGaze [31], including head pose and gaze database [32], ETH-XGaze [33], and RT-GENE [34]. Several approaches have been proposed for appearance-based gaze estimation. For instance, in [22], the authors proposed a multimodal convolutional neural network, but before sending input to this network, the authors used the SURF cascade method [35] and constrained local mode framework to detect faces and facial landmark [36]. Furthermore, the authors also used a space normalization technique to normalize the input image and head pose space into a polar coordinate angle space [23]. Furthermore, in [30], the authors trained an iTracker based on CNN network on the GazeCapture dataset; the network takes the eye image and face image with its location associated in face grid. The face-based 2D and 3D gaze estimation was proposed in [37]. The authors used a special weight mechanism that takes the information of different parts of the entire face and learns the pattern using a standard CNN network. In [24], the authors addressed the person and head pose as an independent problem for 3D gaze estimation by using a recurrent convolution network from remote cameras. They combined the eyes region, face, and facial landmark points as an independent input into the CNN network, and they used a many-to-one recurrent network for gaze coordinate prediction.

In [38], the authors proposed a so-called *Gazemap* for gaze estimation, which is an abstract pictorial representation of the eyeball, iris, and pupil at its center. Instead of directly regressing two angles of the eyeball, the authors regressed an intermediate pictorial representation that simplifies the gaze estimation task. For estimating the gaze information from the ocular region of an image, a capsules network was proposed [39]. The authors used the concept of capsules that converts pixel intensities to instantiation parameters of features, which accumulate into a higher level of features as the depth of the network increases, and proposed pose-aware Gaze-Net architecture for gaze estimation.

Although several studies are available for gaze estimation, only a few studies have covered gaze estimation using unmodified cameras. For instance, in [40], the authors propose a two-stage gaze estimation method that relies on deep learning methods and logistic regression. Their proposed method can be applied to various mobile platforms without additional hardware devices or systematic prior knowledge. In [28], an artificial neural network is proposed to estimate the gaze coordinate from the standard webcam. Their experiment showed a promising result for the development of low-cost tracking using a standard webcam. Model-based gaze estimation is also proposed by Wood et al. for unmodified tablet computers [41]. They presented an EyeTab method based on a model-based approach for binocular gaze estimation that runs solely on an unmodified tablet.

There are some works that are similar to our work. For instance, the authors in [42] designed a differential gaze estimation by training a differential neural network. However, the authors in this paper used calibrated high-quality eye images for training their network, which is different from our method because, in our approach, we used only low-resolution images without calibration. In another work [43], the authors designed a hardware-friendly CNN model that utilizes a minimum computational requirement for efficient gaze estimation on low consumer devices. However, they tested their model on MPIIgaze, the dataset that is a calibrated dataset in which on-screen gaze position is converted to the 3D position in the camera coordinate system using a camera calibration procedure from the OpenCV library.

The work most similar to the investigation presented in this paper is [40]. However, the authors of that paper proposed their method solely for mobile devices. Furthermore, they tested their method on a dataset collected from a single device. Contrary to that, the model presented our paper is tested on desktop computers and laptops. Additionally, the dataset was collected from multiple devices so that it can be generalized for other desktop setups.

3. Materials and Methods

This section first describes the data collection methodology for the conducted research. The following section provides information about data preprocessing steps that are necessary for deep learning. In the third section, information about the dataset used in this research is presented. Finally, the last subsection presents a detailed overview of different architectures tested in this study.

3.1. Data Collection Methodology

This section presents the data acquisition method, which plays a significant role in deep learning methods. The quality of the acquired data, with the help of which the neural network is trained, has a paramount influence on the process itself because any image that deviates from the others can negatively affect the entire process of training the neural network.

Data were collected using the *EyeTrackerDataCollector* application, which is a desktop application registering images from a camera. No specific hardware requirements were made. The only hardware requirement was to have a webcam, preferably built into the top of the laptop screen (see Figure 1). This assumption allowed for the acquisition of images under real-world conditions and not typical laboratory conditions. These measures were designed to answer the question of whether anyone owning standard webcam equipment could use a classifier capable of determining where on the screen a person is focusing their attention. Test subjects were asked to pay attention to the lighting during image acquisition. A problem with the lack of robustness to this factor was noted at the outset. Failure to adjust the lighting resulted in very low-quality, dark even black, or completely overexposed images. The subject sat centrally facing the screen with the built-in camera at the height of the horizontal axis of the eyes at a distance of about 35 to 50 cm from the monitor. During every session, the subject was required to click 20 points displayed subsequently on screen in different evenly distributed locations (in a five by four grid). After each click, a series of three camera images was taken and stored together with information about the point's location.

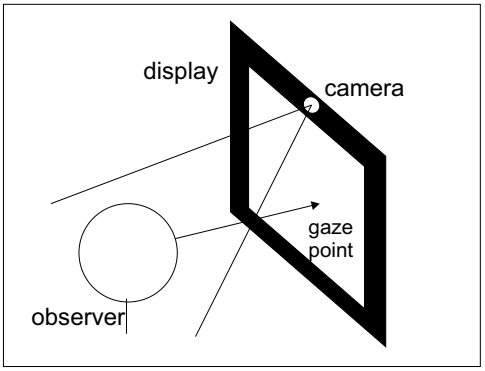

Figure 1. The experimental setup.

3.2. Data Preprocessing

Figure 2 presents the image processing steps. The original images as taken from the camera are presented on the left side of the figure, while the right side presents the images that were additionally sharpened using a function available in the OpenCV library. Step (1) is to take a picture with a webcam. Step (2) is to convert the image to grayscale. Step (3) is to cut out the face detected by the Viola–Jones classifier using a proper Haar cascade [44]. This method was presented by Paul Viola and Michael Jones in 2001. The proposed solution is based on machine learning and uses Haar-like features. The cascade feature is trained on a large number of positive and negative images. It is then used to detect objects in other images. In the first stage, the algorithm needs a large number of positive images, i.e., images that will be targeted for detection, such as images containing a face, and a large number of negative images that do not contain the object to be detected in a later stage. It is important to keep the size of the images the same. To achieve this goal, the images must be scaled appropriately to equal size. It allows the algorithm to omit any unnecessary and meaningless factors that would also have to be considered during learning. Once trained, such a classifier returns a value of "1" when an object is found or "0" if an object is not found. To search for an object across the image, the classifier, in the form of a window, moves across the image in search of the object by checking each location. The classifier is designed to be easily expandable and modifiable. It allows searching for objects of different sizes. Such a solution translates into the classifier's efficiency because the size of the input image is not required to change. To find an object of unknown size in the image, the scanning procedure is repeated several times, and the scaling factor of the classifier is modified each time. We used the cascades available in the OpenCV library.

The right and left eyes are found and marked by using another Haar cascade in step (4). The face is detected first, and then the eyes are detected in the face image. This approach reduced the susceptibility of misclassification of eyes in the image by detecting objects that are confusingly similar to eyes. Step (5) is to precisely cut out the eye itself if possible. Step (6) (the final set) is to mask the eye from the fifth stage with a white ellipse with a black border. Performing the fifth and final sixth stages helps to reduce the influence of meaningless pixels surrounding the eye as much as possible. This approach increases the quality of the trained neural network.

The last step before using the data to train the CNN is to resize the images. For proper operation, the neural network requires that each of the input images should be of the same size. Therefore, all eye images were rescaled to 60×30 pixels and all face images to 227×227 pixels using methods available in the OpenCV library.

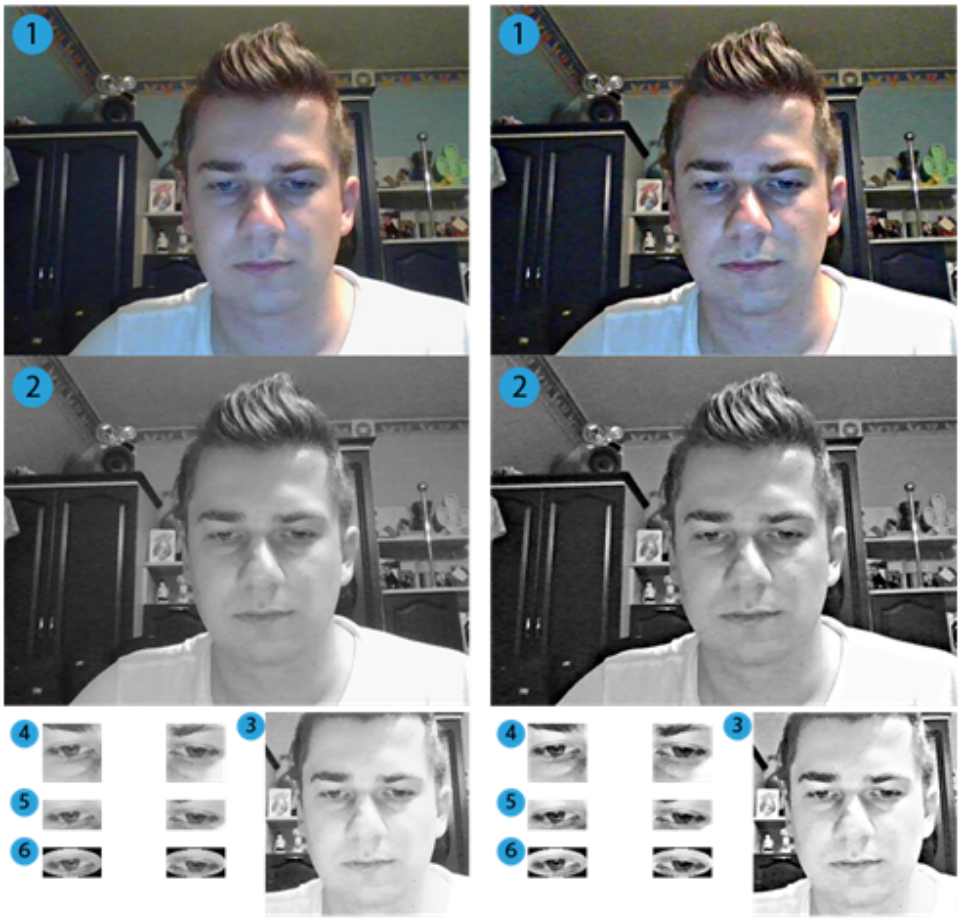

Figure 2. Stages of image processing.

The data prepared with this method must be manually verified. It consists of checking all collected images for their correct content. There are situations when the Haar cascade classifier used for eye detection localizes another element deceptively resembling the eye on the image. There are also cases when the person performing the examination has looked at another place. These cases should not be taken into consideration. There is also a possibility that the person blinks when the picture is taken, and a nearly closed eye is detected. Such deviating images should be manually excluded from the dataset and not subjected to the neural network training process. In the future, we plan to develop more automated methods to identify deviations. Examples of images that have been excluded from the dataset are marked in red and shown in Figures 3 and 4. The first figure shows the situation when the subject blinked when taking the image, while in the second figure, the Haar cascade classifier mistakenly detected an additional element in the image deceptively similar to the eye feature. The detected element is a fragment of the subject's hair.

The final step of pre-processing is image value normalization, which amounts to normalizing the pixel values of an image. The images belonging to the dataset used for the neural network training and testing process are normalized by dividing the values by 255, resulting in the value of each image pixel between 0 and 1.

3.3. Datasets

Two datasets were created and used for the neural network training process. The first dataset consisted of images acquired from a single person. There are approximately 6000 images in this dataset, which is the sum of all images acquired for all 20 points. It implies that there are about 300 sets of images for each point consisting of left and right eye images and a face image. The second dataset consists of images acquired from four

subjects, where one of the subjects additionally performed the tests twice—the first time without glasses (with lenses) and the second time with glasses. In this set, the number of images is more than quadrupled because each tested person was required to provide the same set of data.

Figure 3. An example of blink eye images for the point (1 and 3).

Figure 4. An example of an extra element incorrectly detected as an eye by the classifier.

3.4. Network Architecture of the Tested Models

This section presents a description of three designed models, each of which takes a different set of input images. Each of the designed network architectures has fixed elements implemented: the output layer activation functions used, the hidden layer activation functions, the weight initialization strategy, and the loss function. The presented architectures are already refined models, which are the final ones that emerged after multiple tests of different parameters.

The output of each network is a vector of 20 values. Every value represents a probability that the input images are registered when a person looked at a given area in the 5 by 4 grid of 20 areas.

Every network consists of three kinds of components:

- Convolutional layer (Convolution): The layer's input is an image with some number of channels, and the layer creates another image with the number of layers equal to the number of filters. The number and the size of filters used to convert the image are two parameters of such layer;
- Pooling Layer (Subsampling): The layer's input is an image, and the output is the image reduced in both dimensions. Only Max Pooling layers were used, which reduced the image by representing the area of a given size by one pixel, which is the brightest. There is only one parameter for this layer—the size of the reduction. Only two sizes, 2×2 and 3×3, were used;
- Fully connected layer (FC): It is a classic neural network's layer that consists of some number of neurons, and every neuron received a weighted combination of all input values.

ADAM optimizer [45] was used to train the CNN network with the initial value of the learning rate set to a value equal to 0.00015. The Rectified Linear Unit (ReLU) [46] was used as the hidden layer activation function. Xavier initialization [47] was used as the initialization strategy for weights, while the output layer used an activation function in the form of a normalized Softmax exponential function (see Equation (1)):

$$\sigma(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}} \quad (1)$$

where y_i represents the output from the network, and $K = 20$ is the number of classes.

Negative Log-Likelihood (NLL) was taken as the loss function, which works well with the Softmax activation function for solving multi-class neural network learning and image classification problems considering a set of pixels as input. Equation (2) represents the mathematical formula for the NLL function:

$$l(\theta) = -\sum_{n=1}^{M} (Ylog(\sigma(\theta^T X)) + (1 - Y)log(1 - \sigma(\theta^T X)) \quad (2)$$

where θ is the parameter of the function, Y denotes the output values, X represents the features of the image, and M is the number of samples.

An identical naming convention for the neural network layers was adopted for each experiment. The LRE symbol designation refers to layers using the right eye images, the LLE symbol refers to layers describing the left eye, and LF refers to layers pertaining to face images. The *Convolution* layers always have information about the size of the filter, and the numbers of filters are provided in the text or tables. The *Subsampling* layers have information about the reduction size. The number of neurons for each *Full Connected* layer is explained in the corresponding text. The FC symbol indicates fully connected layers where, for example, FC-RE refers to the fully connected layer for the right eye.

3.4.1. One Eye Image as a Neural Network Input

For this experiment, a network architecture was prepared by accepting an image of one eye in its parameter. The architecture is shown in Figure 5. The neural network consists of three convolutional layers, one subsampling layer with a filter size of 2×2, and one fully connected layer. The first convolutional layer (LLE1) has 96 filters, the LLE3 layer has 384 filters, and LLE4 has 256 filters. The fully connected layer (FC1) has 256 neurons, and the last layer outputs the vector of 20 values. There were two models trained: one with left eyes images and one with right eyes images (further referred to as ARCH-LE and ARCH-RE).

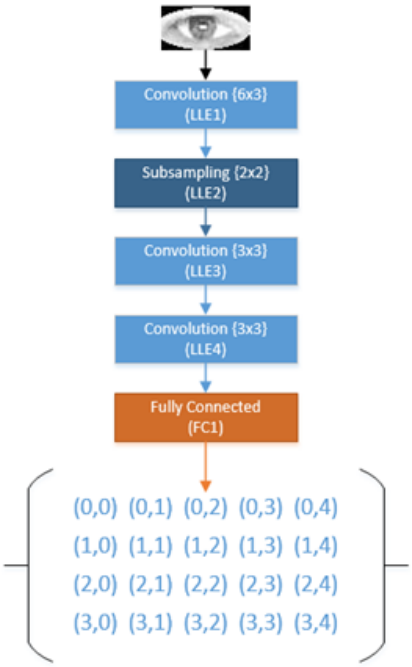

Figure 5. Network architecture for the network processing images with one eye (ARCH-LE and ARCH-RE).

3.4.2. Face Image as Neural Network Input

The neural network architecture prepared for the second experiment takes face images as its parameters. The schema of the architecture is shown in Figure 6. The developed neural network was composed of five convolutional layers, three subsampling layers with filter size 3×3, and two fully connected layers FC1 and FC2. The information of the neural network layers is presented in Table 1. The number of neurons of the fully connected layers was 4096 for FC1 and 1000 for FC2. This solution will be referred to as ARCH-F in the rest of the paper.

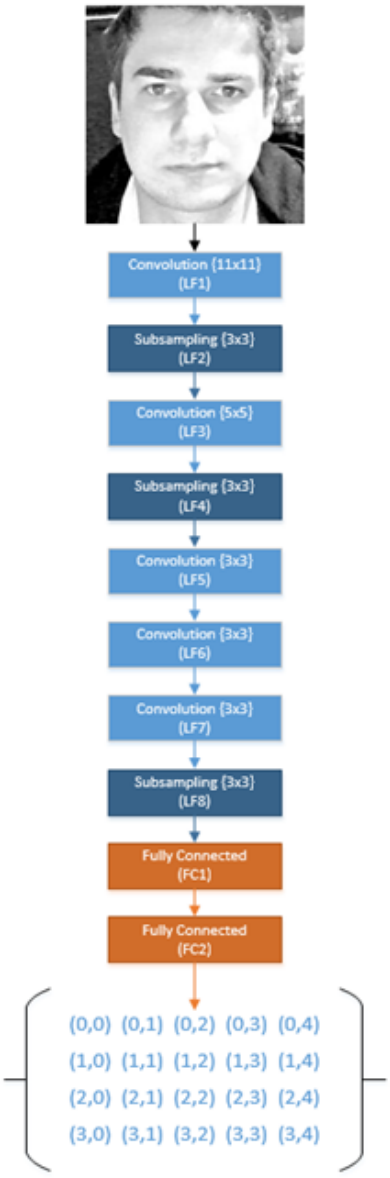

Figure 6. Network architecture for the network processing face images (ARCH-F).

Table 1. Characteristics of convolutional layers for an experiment using face image as neural network inputs.

Layer	Number of Filters	Filter Size	Step
LF1	96	11 × 11	4
LF3	256	5 × 5	1
LF5, LF6	384	3 × 3	1
LF7	256	3 × 3	1

3.4.3. Both Eye Images as a Neural Network Input

For this experiment, a network architecture was prepared to accept left and right eye images as its input parameters. The schema is shown in Figure 7. The neural network consists of three convolutional layers, one subsampling layer 2 × 2, one concatenation layer, and three fully connected layers. The fully connected layers consist of a layer for each input image, i.e., FC-LE1 and FC-RE1 with 1024 neurons, the concatenate layer, and two layers: FC1 with 2048 neurons and FC2 with 1024 neurons. The details of the convolutional layers of the neural network are presented in Table 2. The output of the developed neural network architecture is a set of twenty points representing the position of the area on which the subject focuses his/her gaze. This solution will be referred to as ARCH-LRE.

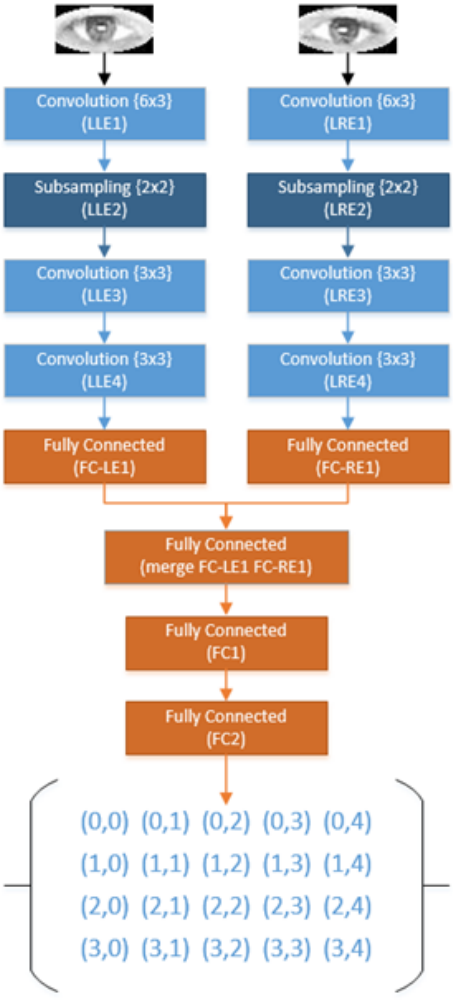

Figure 7. Network architecture for the network using both eyes images.

Table 2. Characteristics of convolutional layers for an experiment using both eyes images as neural network inputs.

Layer	Number of Filters	Filter Size	Step
LLE1, LRE1	2 × 96	6 × 3	2
LLE3, LRE3	2 × 384	3 × 3	1
LLE4, LRE4	2 × 256	3 × 3	1

All three presented network architectures are a little bit different in terms of topology. For instance, one eye image architecture contains only three convolutional layers, one subsampling layer, and one fully connected layer (Figure 5). Face image architecture is more complicated and contains two sets of alternate convolutions and subsampling layers and then again three convolutional layers, one subsampling layer, and finally two fully connected layers (Figure 6). For both eyes architecture (Figure 7), the network has a similar architecture as one eye network, but the outputs from the left eye and right eye stacks are concatenated at the end and sent to two fully connected layers. All three architectures have a different number of parameters because they all have a different numbers of layers. The proposed architectures were chosen from a set of several possibilities that were initially studied. These networks are not generic, and the search for possible better architectures requires further studies.

4. Experiments and Results

This section presents a cross section of the research conducted and the results obtained. In the beginning, the methodology of the conducted research is presented, defining how the results were acquired. The following sections contain a description of each of the experiments conducted. The final section is a summary of the results obtained and conclusions of the research performed.

4.1. Research Methodology

The training set and the test set were determined automatically and randomly in a ratio of 70% for the training dataset and the remaining 30% for the test dataset. Numerically, this translates to approximately 4200 left and right eye and face images for the training set and the remaining 1800 images for the test set for Dataset 1.

Each neural network configuration was repeatedly changed in terms of architecture or hyperparameters and then retested. All experiments were performed on a laptop with Intel Core i7 and 16 GB RAM hardware specifications. The prediction quality was determined by the precision index built into the DeepLearning4j library calculated from the classification accuracy of the test set.

4.2. Results

This section presents the results of tests performed using the designed neural network architectures for the given data. The main goal of the conducted research was to find the optimal architecture capable of correctly classifying the gaze focus point. The resulting models, trained with training sets during the course of the experiments, were embedded in the specially designed *EyeTracker* application, allowing their performance to be tested in real-time.

Each experiment was conducted on two datasets. The first (denoted as D1) contained images acquired from one individual while the second (denoted as D2) contained images collected from four individuals, and one individual performed the test twice with contact lenses and with glasses, as mentioned in Section 3.3. Every network architecture was trained up to 70 epochs.

The first test was aimed at checking if the model will work better after specific preprocessing. The first subset consisted of original grayscale images, and the second of the images was additionally sharpened by using an algorithm available in the OpenCV library.

This experiment was performed based on images from the dataset containing data collected from one person (D1) and using ARCH-LRE architecture. The results of the experiment are presented in Table 3 (columns two and three).

Table 3. Results for original and sharpened images (the latter for both D1 and D2)—averaged accuracy for the range of training epochs.

Epoch	Original Images (D1)	Sharpened Images (D1)	Sharpened Images (D2)
1–10	26.01%	27.52%	16.25%
11–20	41.25%	45.92%	30.73%
21–30	56.48%	59.12%	43.04%
31–40	67.32%	68.34%	53.06%
41–50	73.62%	72.70%	54.46%
51–60	76.52%	77.68%	63.28%
61–70	77.24%	81.61%	73.28%

The second experiment aimed at comparing the results for both datasets D1 and D2. Since the first experiment already proved that the network works better when sharpened images are used, only sharpened images from D2 were used. The results are presented in the last column of Table 3.

The following experiment performed the neural network learning process using an architecture that takes entire face images as input parameters. The schema of the developed ARCH-F architecture used in this experiment is presented in Section 3.4.2. The results of that experiment for Datasets 1 and 2 are presented in Table 4.

Table 4. Results for ARCH-F architecture—averaged accuracy for the range of training epochs.

Epoch	Dataset 1	Dataset 2
1–10	11.80%	7.42%
11–20	15.20%	38.07%
21–30	22.74%	72.66%
31–40	39.88%	80.02%
41–50	61.81%	83.99%
51–60	77.00%	66.61%
61–70	81.81%	80.81%

The last two experiments aimed to check if it is possible to use images of only one eye and achieve results comparable to the more complicated network utilizing two eyes. The results of these experiments for left and right eyes and both datasets are presented in Table 5.

Table 5. Results for ARCH-L and ARCH-R architectures—averaged accuracy for the range of training epochs.

	Left Eye		Right Eye	
Epoch	Dataset 1	Dataset 2	Dataset 1	Dataset 2
1–10	34.72%	44.00%	35.93%	36.22%
11–20	68.51%	64.29%	63.86%	59.49%
21–30	80.86%	72.75%	75.85%	67.94%
31–40	84.71%	75.17%	78.65%	71.95%
41–50	87.15%	77.40%	80.58%	74.34%
51–60	87.77%	78.03%	81.51%	75.37%
61–70	88.55%	79.53%	82.30%	75.93%

4.3. Discussion

The experiments presented in Section 4.2 show that it is possible to achieve decent eye-tracking results even for unconstrained environments and using low-quality web cameras. Of course, the results are far from perfect—we checked only if one of the 20 areas could be determined based on eye image—but this outcome may be sufficient for many human–computer interface (HCI) applications.

The first experiment proved that sharpening the acquired image has a positive impact on further classification (accuracy increased from 77% to 81%). Therefore, all following experiments used the sharpened versions of the images. Not surprisingly, the results for the Dataset 2 were worse than for Dataset 1 for every experiment . However, it is worth noticing that these results are also quite good and had between 72% and 80% accuracy depending on the architecture. It should be remembered that random guessing for 20 classes provides about 5% accuracy, so the results are noticeably better than random ones.

Interestingly, the best results (over 88% accuracy) were achieved for the architecture in which only images of the left eye were used (ARCH-L) (Table 5). The main reason for this phenomenon probably was that the network that uses two eye images and concatenates layers (ARCH-RL) was more complicated than ARCH-L, and 70 epochs were not enough to train it sufficiently.

Another interesting finding was the good accuracy of the model that uses face images instead of eyes. The model obviously has more data to analyze, but a large percentage of this data is probably irrelevant. Due to more extensive input, this model was also more challenging to train; the training process lasted significantly longer than for other models. However, good results suggest that eye detection algorithms may be omitted in some applications.

5. Conclusions

In this study, a possibility to track a person's gaze using an unmodified camera has been investigated. Since commercial eye tracking devices are expensive, their availability is limited. In this research, the low-cost eye-tracking method was proposed which can be easily used for standard desktop or laptop computers without needing any additional equipment. The state-of-the-art CNN network with unmodified webcams commonly available with computers was used. The images acquired from the unmodified cameras are low quality and very sensitive to changing light conditions. Therefore, it was a big challenge to achieve reasonable results. However, this study shows that when the CNN network is used with a carefully designed topology and tuned hyper-parameters, it is possible to achieve results that may be usable in practical applications in a real-world environment.

Although this study achieved some significant results, there are some limitations associated with this study as well. For instance, calculating the exact gaze point was not treated as the regression problem but simplified into classifying gaze point as belonging to one of 20 areas, which makes it difficult to compare with other methods. Furthermore, the number of participants and the amount of data were limited, while it is well known that deep learning networks work and generalize better when trained using huge datasets. Moreover, this study did not include any publicly available datasets to perform the benchmark on them.

In our future research investigations, we plan to address the limitations mentioned above. We will consider extending our dataset and testing our model by using publicly available datasets of eye positions.

Author Contributions: M.F.A. was responsible for preparing the state of the art, elaborating the results, and preparing most of the text; P.K. was responsible for the initial idea, network architecture, results analysis, and supervision; M.O. was responsible for the implementation and performing all calculations. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Statutory Research funds of Department of Applied Informatics, Silesian University of Technology Grant No: 02/100/BKM21/0016.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used in the research will be available on GitHub repository upon acceptance of the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Networks;
PCCR	Pupil Center Corneal Reflections;
HCI	Human-Computer Interfaces;
VOG	Video Oculography.

References

1. Menges, R.; Kumar, C.; Müller, D.; Sengupta, K. Gazetheweb: A gaze-controlled web browser. In Proceedings of the 14th International Web for All Conference, Perth, Australia, 2–4 April 2017; pp. 1–2.
2. Huang, C.M.; Mutlu, B. Anticipatory robot control for efficient human–robot collaboration. In Proceedings of the 2016 11th ACM/IEEE International Conference on Human–Robot Interaction (HRI), Christchurch, New Zealand, 7–10 March 2016; pp. 83–90.
3. Chen, Z.; Shi, B.E. Using variable dwell time to accelerate gaze-based web browsing with two-step selection. *Int. J. Hum.-Comput. Interact.* **2019**, *35*, 240–255. [CrossRef]
4. Pi, J.; Shi, B.E. Probabilistic adjustment of dwell time for eye typing. In Proceedings of the 2017 10th International Conference on Human System Interactions (HSI), Ulsan, Korea, 17–19 July 2017; pp. 251–257.
5. Outram, B.I.; Pai, Y.S.; Person, T.; Minamizawa, K.; Kunze, K. Anyorbit: Orbital navigation in virtual environments with eye-tracking. In Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, Warsaw, Poland, 14–17 June 2018.
6. Patney, A.; Salvi, M.; Kim, J.; Kaplanyan, A.; Wyman, C.; Benty, N.; Luebke, D.; Lefohn, A. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph. (TOG)* **2016**, *35*, 1–12. [CrossRef]
7. Grillini, A.; Ombelet, D.; Soans, R.S.; Cornelissen, F.W. Towards using the spatio-temporal properties of eye movements to classify visual field defects. In Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, Warsaw, Poland, 14–17 June 2018.
8. Hoppe, S.; Loetscher, T.; Morey, S.A.; Bulling, A. Eye movements during everyday behavior predict personality traits. *Front. Hum. Neurosci.* **2018**, *12*, 105. [CrossRef] [PubMed]
9. Rutter, D.R.; Durkin, K. Turn-taking in mother–infant interaction: An examination of vocalizations and gaze. *Dev. Psychol.* **1987**, *23*, 54. [CrossRef]
10. Jacob, R.J.; Karn, K.S. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In *The Mind's Eye*; Elsevier: Amsterdam, The Netherlands, 2003; pp. 573–605.
11. Majaranta, P.; Bulling, A. Eye tracking and eye-based human–computer interaction. In *Advances in Physiological Computing*; Springer: London, UK, 2014; pp. 39–65.
12. Baltrusaitis, T.; Zadeh, A.; Lim, Y.C.; Morency, L.P. Openface 2.0: Facial behavior analysis toolkit. In Proceedings of the 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China, 15–19 May 2018; pp. 59–66.
13. Zhu, W.; Deng, H. Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 3143–3152.
14. Wu, X.; Li, J.; Wu, Q.; Sun, J. Appearance-based gaze block estimation via CNN classification. In Proceedings of the 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP), Luton, UK, 16–18 October 2017.
15. Yang, C.; Sun, J.; Liu, J.; Yang, X.; Wang, D.; Liu, W. A gray difference-based pre-processing for gaze tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Beijing, China, 24–28 October 2010; pp. 1293–1296.
16. Niu, C.; Sun, J.; Li, J.; Yan, H. A calibration simplified method for gaze interaction based on using experience. In Proceedings of the 2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), Xiamen, China, 19–21 October 2015.
17. Ji, J.C.Q. 3D gaze estimation with a single camera without iR illumination. In Proceedings of the 2008 19th International Conference on Pattern Recognition, Tampa, FL, USA, 8–11 December 2008.
18. Chen, Z.; Shi, B.E. Geddnet: A network for gaze estimation with dilation and decomposition. *arXiv* **2020**, arXiv:2001.09284.
19. Wang, H.; Pi, J.; Qin, T.; Shen, S.; Shi, B.E. SLAM-based localization of 3D gaze using a mobile eye tracker. In Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, Warsaw, Poland, 14–17 June 2018.
20. Brau, E.; Guan, J.; Jeffries, T.; Barnard, K. Multiple-gaze geometry: Inferring novel 3D locations from gazes observed in monocular video. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 612–630.

21. Zhang, X.; Sugano, Y.; Fritz, M.; Bulling, A. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *41*, 162–175. [CrossRef] [PubMed]
22. Zhang, X.; Sugano, Y.; Fritz, M.; Bulling, A. Appearance-based gaze estimation in the wild. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 4511–4520.
23. Sugano, Y.; Matsushita, Y.; Sato, Y. Learning-by-synthesis for appearance-based 3D gaze estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1821–1828.
24. Palmero, C.; Selva, J.; Bagheri, M.A.; Escalera, S. Recurrent cnn for 3D gaze estimation using appearance and shape cues. *arXiv* **2018**, arXiv:1805.03064.
25. Lou, Y.; Wu, R.; Li, J.; Wang, L.; Chen, G. A Convolutional Neural Network Approach to Predicting Network Connectedness Robustness. *IEEE Trans. Netw. Sci. Eng.* **2021**, in press. [CrossRef]
26. Lou, Y.; He, Y.; Wang, L.; Tsang, K.F.; Chen, G. Knowledge-Based Prediction of Network Controllability Robustness. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, in press. [CrossRef]
27. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
28. Sewell, W.; Komogortsev, O. Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*; Association for Computing Machinery: New York, NY, USA, 2010; pp. 3739–3744.
29. Baluja, S.; Pomerleau, D. *Non-Intrusive Gaze Tracking Using Artificial Neural Networks*; Technical Report; Carnegie-Mellon University: Pittsburgh, PA, USA, 1994.
30. Krafka, K.; Khosla, A.; Kellnhofer, P.; Kannan, H.; Bhandarkar, S.; Matusik, W.; Torralba, A. Eye tracking for everyone. In Proceedings of the IEEE IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2176–2184.
31. Huang, Q.; Veeraraghavan, A.; Sabharwal, A. TabletGaze: Unconstrained appearance-based gaze estimation in mobile tablets. *arXiv* **2016**, arXiv:1508.01244.
32. Weidenbacher, U.; Layher, G.; Strauss, P.M.; Neumann, H. A comprehensive head pose and gaze database. In Proceedings of the 2007 3rd IET International Conference on Intelligent Environments, Ulm, Germany, 24–25 September 2007; pp. 455–458.
33. Zhang, X.; Park, S.; Beeler, T.; Bradley, D.; Tang, S.; Hilliges, O. ETH-XGaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2020; pp. 365–381.
34. Fischer, T.; Chang, H.J.; Demiris, Y. Rt-gene: Real-time eye gaze estimation in natural environments. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 334–352.
35. Li, J.; Zhang, Y. Learning surf cascade for fast and accurate object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3468–3475.
36. Baltrušaitis, T.; Robinson, P.; Morency, L.P. Continuous conditional neural fields for structured regression. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 593–608.
37. Zhang, X.; Sugano, Y.; Fritz, M.; Bulling, A. 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, Honolulu, HI, USA, 22–25 July 2017; pp. 51–60.
38. Park, S.; Spurr, A.; Hilliges, O. Deep pictorial gaze estimation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 721–738.
39. Mahanama, B.; Jayawardana, Y.; Jayarathna, S. Gaze-Net: Appearance-based gaze estimation using capsule networks. In Proceedings of the 11th Augmented Human International Conference, Winnipeg, MB, Canada, 27–29 May 2020; pp. 1–4.
40. Xia, Y.; Liang, B.; Li, Z.; Gao, S. Gaze Estimation Using Neural Network And Logistic Regression. *Comput. J.* **2021**, in press. [CrossRef]
41. Wood, E.; Bulling, A. Eyetab: Model-based gaze estimation on unmodified tablet computers. In Proceedings of the Symposium on Eye Tracking Research and Applications, Safety Harbor, FL, USA, 26–28 March 2014; pp. 207–210.
42. Liu, G.; Yu, Y.; Mora, K.A.F.; Odobez, J.M. A differential approach for gaze estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 1092–1099. [CrossRef] [PubMed]
43. Lemley, J.; Kar, A.; Drimbarean, A.; Corcoran, P. Convolutional neural network implementation for eye-gaze estimation on low-quality consumer imaging systems. *IEEE Trans. Consum. Electron.* **2019**, *65*, 179–187. [CrossRef]
44. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1.
45. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
46. Xu, B.; Wang, N.; Chen, T.; Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv* **2015**, arXiv:1505.00853.
47. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.