UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

STU FEI
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

**DIPLOMA PROJECT**

*Master in Informatics Engineering*

# Anti-spoofing mechanisms in face recognition

*Research, implementation and comparison between methods for pupil detection in an image*

**REPORT AND ANNEXES**

**Author:** Sergi Bové Escribano

**Tutor:** Ing. Dominik Sopiak

**Course:** Summer semester 19-20

# Abstract

The objective of this work is to design and implement a pupil detection solution to be used as part of an anti-spoofing mechanism in face recognition. This is intended to improve the security and reliability of face recognition technology by minimizing its risks and, thus, provide confidence to users and companies that employ it. In addition, the project wants to focus on the efficiency and accuracy of the final solution, to do so, various alternatives based on different methods will be developed and compared.

To achieve this, the project follows a linear work methodology, that is to say, starting from the search of information, through the theoretical design phase, the code implementation and ending with the presentation of an operational solution. The results achieved after the whole process have been satisfactory and show the viability of the target set. Also, the work provides a series of possible future improvements while reflecting how the initial idea has been evolving and maturing to end up resulting in a product more solid and adjusted to the detected need.

*Last modification of the document: September 2, 2020.*

# Acknowledgements

This master thesis would not have been possible without the invaluable collaboration of several people, who have done a magnificent job with the aim of completing the project successfully. First of all, thank the tutor of this work, Ing. Dominik Sopiak, for his collaboration, both technically and personally for his guidance and advice throughout the time invested in supervising the development of pupil detection solutions. He has been key to not losing sight of the goals and helping to solve the problems that have arisen. I must also thank the FIB's Department of International Relations, especially to Carolina Martín and María Teresa Abad, for his good attitude and effort in always being willing to help in whatever is needed, without them much administrative problems would not have been solved successfully. Finally, a special thanks to Wolfgang Fuhl, researcher in the Department of Computer Science at the University of Tübingen and author of several pupil detection algorithms, who kindly provided multiple eye images and papers that have been really useful for the project.

Thanks to them all!

# Glossary

| | |
|---|---|
| AI | - Artificial intelligence |
| CNN | - Convolutional Neural Network |
| DL | - Deep Learning |
| GT | - Ground Truth |
| IDE | - Integrated Development Environment |
| IoU | - Intersection Over Union |
| LPW | - Labelled Pupils in the Wild |
| mAP | - Mean Average Precision |
| ML | - Machine Learning |
| OS | - Operating System |
| POC | - Point of Consumption |
| PX | - Pixel |
| RANSAC | - Random Sample Consensus |
| ROI | - Region of Interest |
| TF | - TensorFlow |
| THOLD | - Threshold |

# Figures list

# Tables list

# Index

# Preface

## Origin of the project

Nowadays, face recognition has already become a common technology in our daily life. Smartphones have played an important role in this trend by implementing applications and cameras able to use this technology that have become a must when buying a new device. Its practicality, smooth operation and comfort have put this relatively new technology in the media spotlight. Since its outbreak, it has left no one indifferent and sparked fascination from the press to governments and among the general public as no other technology has done. Despite of the promising panorama, face recognition has also its dangers, like is identity fraud.

## Motivation

Believing in a technology also implies helping to improve it and this, together with the interest that its potential arouses and the promising panorama described before, was what led me to accept this part of the project that Mr. Dominik Sopiak leads. Although the starting point of this work is when a technology weakness is found and you start looking for a solution to the detected problem, the project also offered the possibility of entering in new branches of the computer science such are machine learning and artificial intelligence, disciplines with a great future projection. This idea of taking part in a leading project that offers entering in a very dynamic and constantly changing world has been the trigger for the start of this work.

## Previous requirements

The desire to develop a pupil detection algorithm requires a set of prerequisites. These are basically theoretical knowledge about image treatment, machine learning and artificial intelligence, especially about neural networks. It is also required to have knowledge in programming languages, in this case Python, and skills to be able to handle with the computer science problems that arise along the work as well as practice on how to code and debug algorithms. In addition, good team work skills and the ability to be organized are also highly valuable.

# Introduction

This work aims to advance in the project of developing an anti-spoofing mechanism in face recognition but, to accomplish this goal, it is necessary first to understand the various tasks and stages that this encompasses. Most of them are purely computer engineering aspects such as selecting the software and frameworks to be used, deciding the techniques to detect the face and the pupil or obtaining useful datasets of labelled images while others require of smart ideas like designing the strategy to decide if there is spoofing or not.

However, as can be suspected, the extent and workload of the project far exceeds that of a master thesis project. This reason has led to the division of tasks into smaller projects, thus ensuring a better dimensioning and distribution of the work and the possibility of assigning them to a student of the most appropriate specialty according to the nature of the task.

## Project objectives

Based on the above, the aim of this work is to complete one of the necessary parts for developing an anti-spoofing mechanism in face recognition, in this case, the detection of the pupils in an image. More specifically, this project will design a strategy to detect the pupil by selecting, implementing and testing several algorithms used for pupil detection. This involves searching for information on various algorithms and technologies to be able to select the most appropriate ones in terms of accuracy and computational cost. Other important aspects to take into account are the implementation cost of the chosen detection methods and the selection of the indicators that will be used to test and compare the algorithms.

At the same time, the work aims to show in a linear and transparent way the process of selection, implementation and subsequent testing, indicating the criteria used to select the algorithm that best suits the needs of the project. To all this we must add the desire to show a wide and detailed explanation of each detection method as well as a solid comparison between them so the reader can realise about strengths and weaknesses of each algorithm and use this work as a base for deciding its own detection strategy. Finally, this project is intended to serve as a gateway to the artificial intelligence and deep learning fascinating world.

## Project scope

The present work is included, as mentioned before, in a larger project which is the design and implementation of anti-spoofing mechanisms for face recognition and is divided into several works. One of these works in which the project has been reduced is the one that precedes this one with the aim of capturing the image where to detect the pupils, either extracting a frame from a video or using a webcam. The other work that has to be taken into account is the one that follows the pupil detection, which is detecting the face orientation. Having both works preceding and following the current project limits the scope of this work significantly.

In this way, the scope of the work is to select several algorithms for pupil detection as well as implement them to be able to choose the one that fits better the project purpose. It is also responsibility of this work to ensure that the code is operational once it is completed and it is fully integrated in the already existing code from other parts of the project. In addition, as already stated in the objectives, a solid comparison between the tested algorithms will be provided to justify the selection.

# CHAPTER 1.
# A global view: pupil recognition as part of an anti-spoofing mechanism

When speaking about face recognition the vast majority of people references it to its fictional use in films like James Bond or series like CSI or Black Mirror. More or less the same happens when we talk about artificial intelligence, in the collective imagination, especially for older people, this is an almost magical technology with an incomprehensible operation. However, as is evident, it is not. And not only is it not, despite all its advantages, it also has its risks like spoofing.

In this chapter we will talk about the spoofing risk in face recognition and how we can solve it, or at least reduce it, to make it a safe and trustworthy technology. As a consequence, the following paragraphs are only directed to serve as a logical and ordered introduction to arrive to the real purpose of the project, the pupil detection, with the enough knowledge as to understand it and perceive the reasons that made it necessary.

## 1.1. Context and reasons for developing an anti-spoofing mechanism

Before planning how any anti-spoofing mechanism it is necessary to have an overview on how face recognition appeared and which are its problems so it is possible to design a proper solution. In the following sections a brief analysis on these points is done.

### 1.1.1. The artificial intelligence and face recognition growth

In the recent decade the world has experienced such a fast improvement on the face recognition technology. That, together with the rising acceptance of this new authentication methods by the society, maybe due to its advantages and practicality, maybe because humanity is getting used to it, this fact has allowed that nowadays almost everyone is aware of what face recognition is. Moreover, it has been already incorporated it in our daily life through, for example, smartphones. In other words, is easy to see that face recognition technology is here to stay and has yet a big potential of expansion in the upcoming years with new implementations and improvements.

But where all this success and expectations come from? Let's take a look on how the face recognition has risen since it was invented in 1960s. The following chronologically ordered list shows some of the most remarkable events in its progression.

*1960s: with funding from an unnamed intelligence agency, Woodrow Wilson Bledsoe creates first manual measurements using electromagnetic pulses.*

*1970s: Researchers Goldstein, Harmon, and Lesk establish 21 points of facial measurement*

*1988: Kirby and Sirovich establish normalized face image using fewer than 100 points of facial measurement using linear algebra.*

*1991: Turk and Pentland invent first crude automatic face detection from images.*

*1993: Defense Advanced Research Projects Agency (DARPA) creates the first basic database of facial images.*

*2002: A face recognition database of 856 people is used at Super Bowl XXXV. The experiment fails.*

*2003: DARPA database upgrades to 24-bit color facial images.*

*2004: National Institute of Standards and Technology creates the NIST test.*

*2009: Pinellas County Sherriff's Office creates forensic database.*

*2010: Facebook creates image identity auto-tagging using face recognition.*

*2011: Panama Airport installs first face recognition surveillance system.*

*2011: Body of terror mastermind Osama bin Laden positively identified using face recognition.*

*2013: FaceFirst reaches effective real time mobile match alerts over cellular connection.*

*2014: Automated Regional Justice Information System (ARJIS) deploys a cross-agency system in southern California, sharing criminal face recognition data across local, state and federal agencies.*

*2016: U.S. CPB deploys exit face recognition at Atlanta Airport.*

*2017: iPhone X becomes the world's top-selling phone with face recognition access control.*

*2018: FaceFirst achieves 150,000 facial points of measurement, including the ability to determine identity from 90-degree profile images.*

*2018: Japan announces that it will use face recognition to verify the identity of athletes at the 2020 Olympic Games.*

As can be seen, the evolution of face recognition has gone so fast and, since Apple incorporated face recognition on its devices many other brands started incorporating it as well resulting in a common feature of the nowadays smartphones, and thus, of our daily life.

### 1.1.2. Face recognition risks

Apart from the good expectations in the face recognition technology exposed before, there are also some dangers to take into account. While it becomes more and more popular, the dangers of spoofing increase such as identity fraud or the control of the masses by governments and large companies that have huge databases created over decades.

While the first one is the scariest and maybe the most exploited in films and novels, it cannot be prevented from an engineering point of view. It has to be controlled by effective laws and education of the society from an ethical point of view. Betting on privacy and the responsible use of data is the only way in which these technologies can contribute can contribute well-being to society and earn the trust of society.

Leaving the previous problem to lawyers, educators and sociologists, the second problem is much more suitable for the work of a computer engineer. And this is where this project is focused: the need of finding mechanisms to prevent identity fraud and make face detection a safe and reliable technology so it can be widely used in all the fields where it can be helpful without compromising people's and organizations' security.

## 1.2. A global view of the designed anti-spoofing mechanism

In the previous sections it has been justified the necessity of anti-spoofing mechanism. In this one it will be explained how the mechanism developed along this project will work and which is its strategy and basis.

Nevertheless, before start talking about this topic, it is a good practice to remember that this anti-spoofing mechanism is not designed by the author of this project. The idea comes from the tutor of the project, Ing. Dominik Sopiak, who is leading the whole project that contains different smaller parts. One of these smaller parts is this one, which, as explained in the introduction and scope sections is aimed at developing, implementing and testing of the necessary pupil detection algorithms and techniques.

### 1.2.1. Designed anti-spoofing mechanism strategy

The anti-spoofing mechanism developed in this project relies on the user cooperation. After some image processing to detect the pupils and gaze orientation the user will be asked to follow a random path with the gaze so it can be assured that is a real person. As repeated uncountable times before, this objective requires a lot of parts to be fulfilled, so this master thesis only aims to solve a small part of the whole task: the improvement of the pupil detection.

Below a brief schematic showing the whole project scope can be seen. The part to which this document is focused is marked in orange.

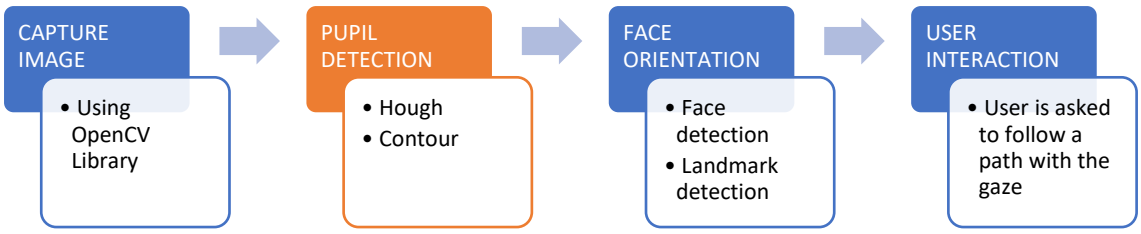



*Figure 1. Schematic overview of the different parts of the designed anti-spoofing mechanism to be developed in this project.*

Analysing a little bit deeper the previous schematic it is possible to see that the strategy on which the method is based is divided into four parts. The first part has been already solved by other students that have previously worked on the project. It consists on acquiring images that contain human eyes so the pupil detection can be performed on it. This part will be explained in detail in the following chapters as it is closely related with the purpose of the project and can have an important influence on the pupil detection results.

### 1.2.2. Pupil detection's role in the designed anti-spoofing mechanism

The pupil detection technical aspects will be skipped in here as it will have a whole chapter dedicated to it later. Nevertheless, it is important to understand the role of pupil detection in the whole project.

To be able to determine if the user is following the required path with the gaze it is necessary to know the pupil position so it is possible to track the gaze. Doing that requires a good accuracy as the eyes can be a small region on the image and there is the possibility that the environmental conditions makes the task even more difficult.

As a result, an accurate pupil detection algorithm is fundamental for the rest of the project as it is the basis for the following parts. If the pupil detection fails the algorithms that uses its output value as their input will also provide a wrong result. In consequence, to be able to provide a good enough solution, two different methods based on different technologies have been chosen. A computational method based on the Starburst algorithm and a solution based on the CNN developed by Google named Inception V4.

### 1.2.3. Future parts, face orientation and user interaction

The third and fourth steps are not explained in detail as they are not part of this thesis and also, because they are yet not clear as they require pupil detection as its basis. Despite of that a brief summary reviewing the main idea and the key parts of them is worth it to get a global view of the whole project.

Regarding the third step of the mechanism, the face orientation, it is required to be able to determine the gaze direction and its result is closely tied to a good functioning of the pupil recognition as if the centre of the pupil is incorrectly detected the gaze orientation cannot be calculated.

Finally, the fourth step consists on providing the user with an image on the screen containing a path that should be randomly generated. After that, the user is asked to follow this path with the gaze and while doing so the system is acquiring the image containing his or her face to calculate the gaze direction and, thus, calculate if the user is following the given path or not. The great point of the whole idea is asking the user for cooperation as it is a way to assure that is a real person if the path is followed correctly.

# CHAPTER 2.
# Basic theoretical concepts, general design lines and resources used

While the previous chapter gave a global view over the anti-spoofing mechanism to be developed and contextualized the world situation that makes it necessary, this chapter pretends to provide a list of theorical concepts used in the project and some general design lines that will be followed. In some way, this chapter supplies the questions and answers that were found at the very beginning of the project when starting to go deep in the face recognition field. A collection of all the knowledge that have been acquired along the time necessary to understand the resulting work and build the mental structure to be able to work on it.

## 2.1. Identifying the abstract idea of pupil

When detecting something in an image the first important thing that has to be clear is what has to be detected and how is it. To do so, is important to understand the abstract idea behind the mental image that we have about a pupil. If a simplification effort is made a pupil can be described as a black circle inside of another circle which, in turn, is inside the eye region between two eyelids. Thus, it is possible to affirm that no pupil exists outside the eye, and as a consequence any black circle that is outside of the eye can be anything else but not a pupil. In the following image can be seen a drawing containing the principal parts of an eye.

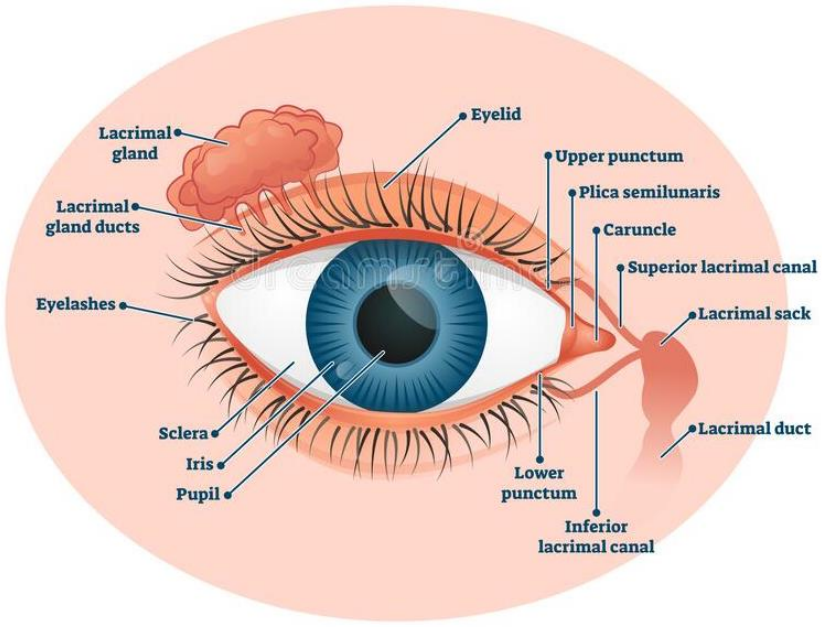



*Figure 2. Drawing of the eye parts (Source: https://dreamstime.com).*

From the previous reasoning, it is clear that detecting a pupil will be closely tied to detecting where the eye is in an image. Once the eye is detected detecting a pupil can be reduced to detect a circle, most of the times, with something partially overlapping it. Also, the intensity of the colour plays an important role when segmenting it from the iris but in a general way if this is clear detecting the pupil with the help of image processing methods and techniques is feasible.

It has to be said that the previous explanation goes more related to the eye detection using a traditional algorithm than to detecting it with a convolutional neural network as the last one works based on the learning acquired along the training process instead of following several conditions. Despite of that, it is worth it for the reader to think about the pupil in a more deep and abstract way and to get familiar with it.

## 2.2. An introduction to image processing

Most of the time detecting a pupil in an image is not easy at all. Black rounded regions easily confused with a pupil, dark iris, closed eyes or glints are frequent and complicate the task of detecting it. However, there are techniques that uses algorithms to modify the image and make the task of detecting a pupil much easier. This process is known as image processing.

According to Wikipedia:

> *"In computer science, digital image processing is the use of a digital computer to process digital images through an algorithm."*

Image processing usually involves one or more transformations of the image and, due to the matrix nature of an image, it is really close to mathematical matrix transformations. Some examples of it could be using algorithms to apply morphological transformations as open, dilate or close areas. As well as to change the colour scheme by merging channels. Other techniques can also involve reducing or increasing the matrix formed by the pixels, for example, to crop a region of interest from the picture. Also, many other techniques can be used as can be seen in the following image where a filter is applied.

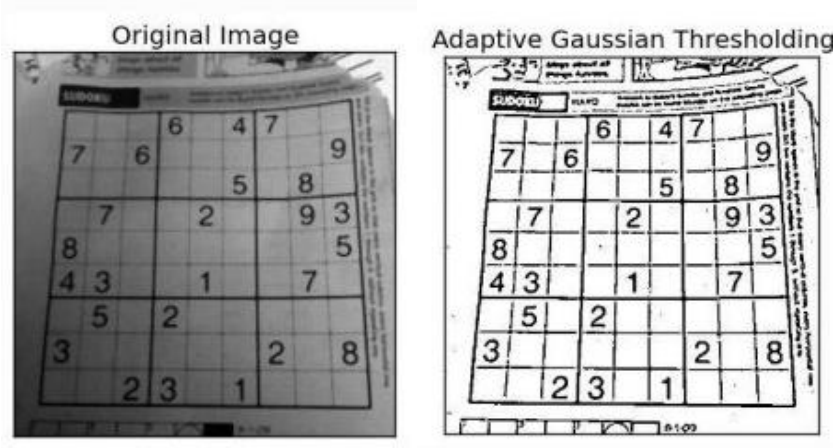



*Figure 3. Example from the OpenCV documentation where a filter is applied to an image.*

The previous image pretends to illustrate how image processing can make features in an image much more visible and accentuated and at the same time make the process of detecting them much easier for the detection algorithms. In general, it is worth applying image processing techniques, since due to their computational cost, they often save processing time later. Indeed, they can make the difference between whether the detection algorithm is able to provide a correct or incorrect result. For further information on this topic a section regarding the image processing techniques used will be find in the chapters where each detection solution is explained.

## 2.3. An introduction to traditional algorithms

Probably, if someone is stopped on the street and asked how a pupil can be detected on an image, after few seconds of thinking he or she will start enumerating a series of steps to do it as if it were a recipe. Unconsciously, what this person is creating is a traditional algorithm. In this section this concept will be explained in further detail and show how it works.

According to the Encyclopaedia of Mathematics:

> *"A traditional algorithm is realized in the form of a computational process, i.e. as a finite sequence of states of a real computer, discretely distributed in time, the real computer having a restricted rate of performance of the operations, a restricted number of digit places to form a number and a restricted storage capacity."*

In this project a traditional algorithm, understood as a finite sequence of steps, will be used for detecting pupils and is based on the Starburst method. Further information on its functioning will be provided in the chapter dedicated to it.

## 2.4. An introduction to Convolutional Neural Networks (CNN)

Until relatively recently, the most widespread solution to try to solve a problem or a need that required great computing power through software, in our case detecting a pupil, was the use of a traditional algorithm that works as explained before. It was not until the 1990s, despite its origin dating back to 1943, that a new alternative solution was consolidated: the famous neural networks. In this section a brief introduction to neural networks, specifically a subset of them known as Convolutional Neural Networks, will be done so the reader can better understand the following chapters.

### 2.4.1. Brief history of Neural Networks

The history of how neural networks have achieved current fame and generated enormous expectations among the society has highs and lows throughout the more than 75 years since the first ideas that inspired them were published. In the following graph you can see the main events of its short but intense history.

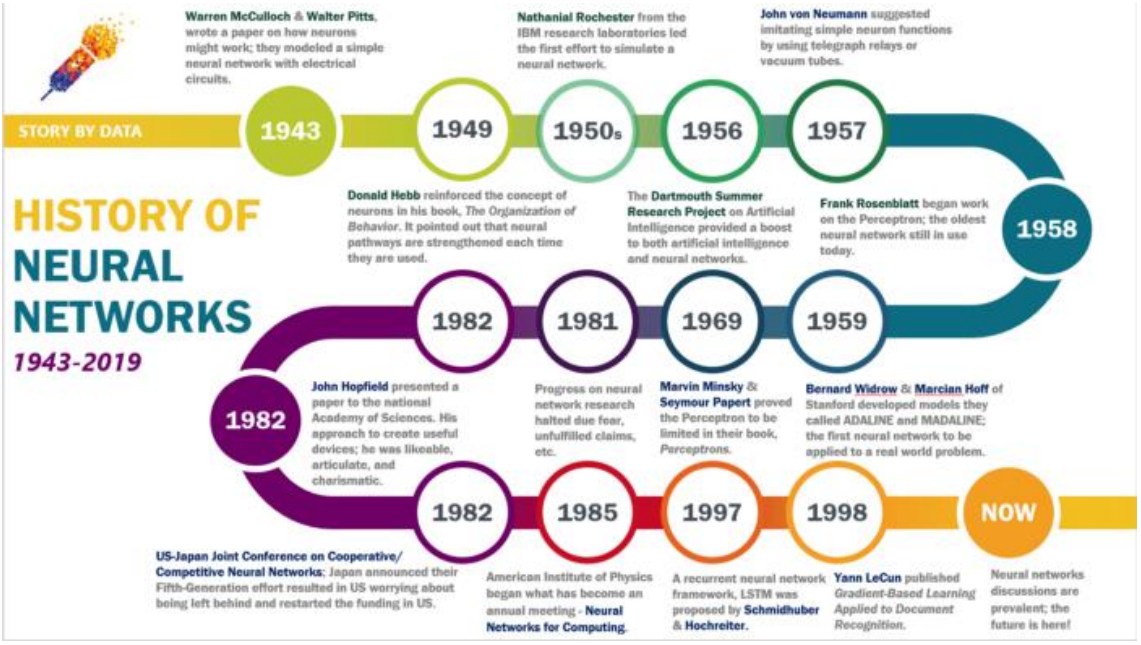



*Figure 4. Graphic that shows the main events in the history of neural networks. (Source: https://medium.com/analytics-vidhya)*

### 2.4.2. Understanding the structure of a Neural Network

To understand why neural networks have generated such those big expectations it has to be understood how they work and why are they able to provide undeniably good results in many fields of application. In this section the parts of a neural network will be explained in detail.

#### 2.4.2.1. The neuron concept

As the name suggests, neural networks are inspired by the neural architecture of a human brain, and like in a human brain the basic building block is called neuron. As the following drawings show, the functionality of a neuron is similar to the human one, it takes in some inputs and provides an output.

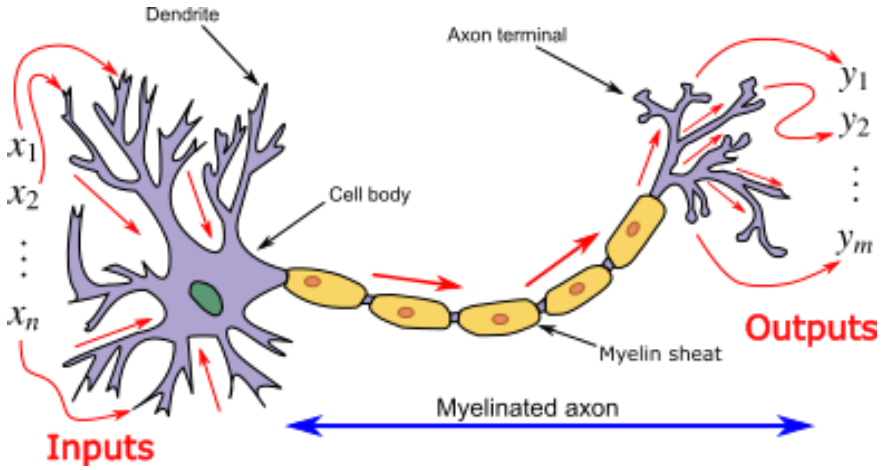



*Figure 5. Drawing of a human neuron structure showing its basic functioning.*

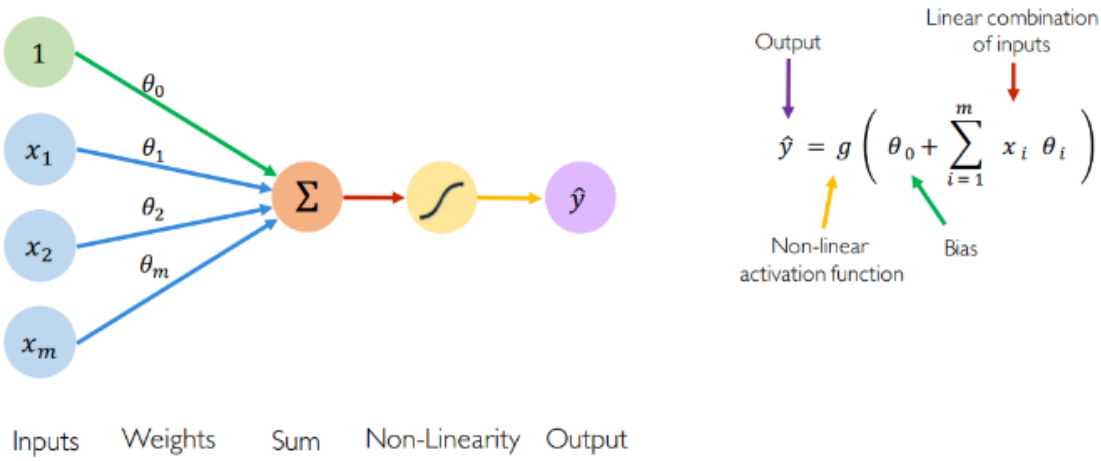



*Figure 6. Schematic of an artificial neuron showing its parts and the mathematical formula that rules its output value.*

In mathematical terms, a neuron in the deep learning world is a placeholder for a mathematical function, and its job is to generate an output by applying the activation function with the provided inputs. In a more simplified way, a neuron is a variable that holds a numerical value that is set when training the network.

#### 2.4.2.2. Activation functions

Each one of the neurons in the neural network, as seen before, holds a value. This value is calculated using the provided inputs using a function which is commonly termed as activation function. As one can foresee, this function is very important as it defines the value that the neuron stores. As a consequence, choosing the type of function is an important decision. Over the time many different functions have been tested to evaluate which offers the best performance, in this section the most common ones will be explained.

##### 2.4.2.2.1. The Step function

One of the first activation functions proposed in the beginning of neural networks was the step function. A function that provides a binary output, the maximum or minimum value. Its mathematical function is represented below.

$$f(x) = \begin{cases} 0\ if\ 0 > x \\ 1\ if\ x \geq 0 \end{cases}$$

Using this activation function the output is 1 if the value of x is greater than equal to zero and 0 if the value of x is less than zero. As can be seen a step function is non-differentiable at zero. Modern neural networks use back propagation method along with gradient descent to calculate weights of different layers. Since the step function is non-differentiable at zero hence it is not able to make progress with the gradient descent approach and fails in the task of updating the weights.

2.4.2.2.2. The Sigmoid function

To overcome the Step function's problems the sigmoid function was introduced. It is considered as one of the traditional activation functions, nowadays obsolete but necessary to know.

***Sigmoid function formula:***

$$f(x) = \frac{1}{1 + e^{-x}}$$

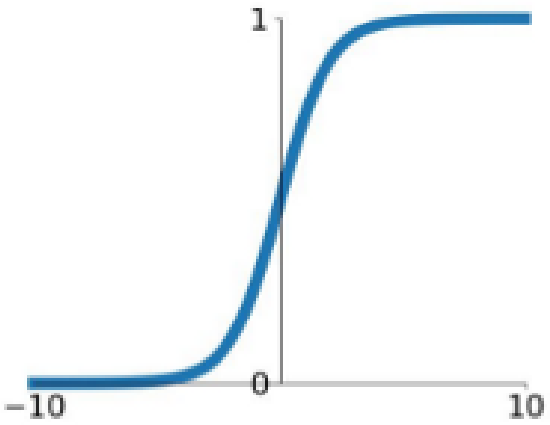



*Figure 7. Mathematical formula and graphical representation of a Sigmoid function.*

The virtues of this function are that it captures non-linearity in the data. Albeit in an approximated form, but the concept of non-linearity is essential for accurate modelling. Also, the sigmoid function is differentiable throughout and hence can be used with gradient descent and backpropagation approaches for calculating weights of different layers. Finally, a sigmoid function inherently assumes a Gaussian distribution for the independent variable which is a general distribution we see for a lot of randomly occurring events and this is a good generic distribution to start with. Although the described advantages, the sigmoid function also suffers from a problem of vanishing gradients.

2.4.2.2.3. The Hyperbolic tangent function

The second traditional activation function is the Hyperbolic tangent. Below its mathematical formula and graphical representation is shown.

***Tanh function formula:***

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

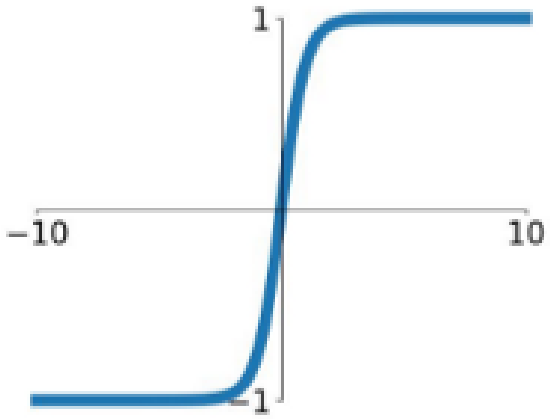



*Figure 8. Mathematical formula and graphical representation of a Hyperbolic Tangent function also known as Tanh.*

The reason for using the Tanh activation function in some cases as a replacement of the Sigmoid function is that since data is cantered around 0, the derivatives are higher and it has been observed that a higher gradient helps in a better learning rate.

#### 2.4.2.2.4. The ReLU function

The Rectified Linear Unit or just ReLU function is one of the named new functions widely used nowadays and solves many problems that traditional ones had. Below the mathematical formula representing it and its graphical shape are displayed.

***ReLU function formula:***

$$f(x) = \max(0, x)$$

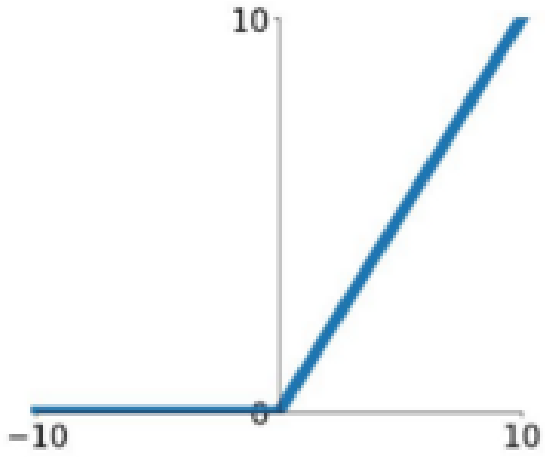



*Figure 9. Mathematical formula and graphical representation of a ReLU function.*

The function is as simple as it looks like, it returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Its simplicity makes the training of the neural network easier and, as a consequence, faster.

Many alternatives to it have been tried but generally there is no sufficient benefit to justify using anything other than ReLU. In general, ReLU has found to be performing better than Sigmoid or Tanh functions.

2.4.2.2.5. The Leaky ReLU function

The Leaky ReLU activation function is a slight variation of the original ReLU. In the following illustration its representation and formula can be found.

***Leaky ReLU function formula:***

$$f(x) = \max(0{,}1x, x)$$

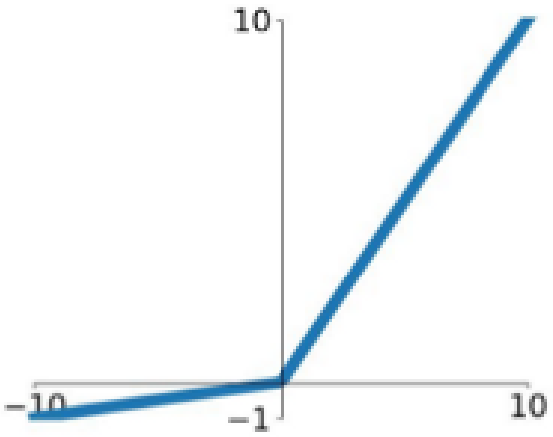



*Figure 10. Mathematical formula and graphical representation of a Leaky ReLU function.*

The Leaky ReLU is one of the most well-known alternatives to ReLu. It is the same as ReLU for positive numbers but, instead of being 0 for all negative values, it has a constant slope which is always less than 1.

Other modern alternatives to the previously overviewed activation functions could be ELU, Softmax, SELU or GELU among many others.

2.4.2.3. Layers

Now that the neuron concept is explained and its value calculation is clear let's place the focus on how they combine and interact with each other to form neural networks as if it were a human brain.

The chosen structure to place the neurons inside a neural network is in layers. A layer is just a matrix that holds a group of neurons. Every layer is fully or partially connected to the next layer with weighted connections. These weights are the values that are modified during the training process and determine how the values in the previous layer influence in the next one. Below, an illustration showing the matrix operations performed to calculate the values in the next layer using weights and biases is shown.

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

*Figure 11. Matrix operations that represent how layers are connected between them in a neural network from a mathematical point of view.*

There are 3 type of layers that form a Neural Network: input, hidden and output layers. The first type is the input layer, where the raw data for the neural network is injected. For example, in case that the input data are the pixels of an image, this layer will have the same dimensions as the picture. The second type are the hidden ones, intermediate layers between input and output layer. The number of hidden layers can vary depending on the network and it is the place where all the computation is done. Finally, the third type of layer is the output layer, the one that produce the result for given inputs. In the next illustration a representation of a neural network and its layers structure can be found.

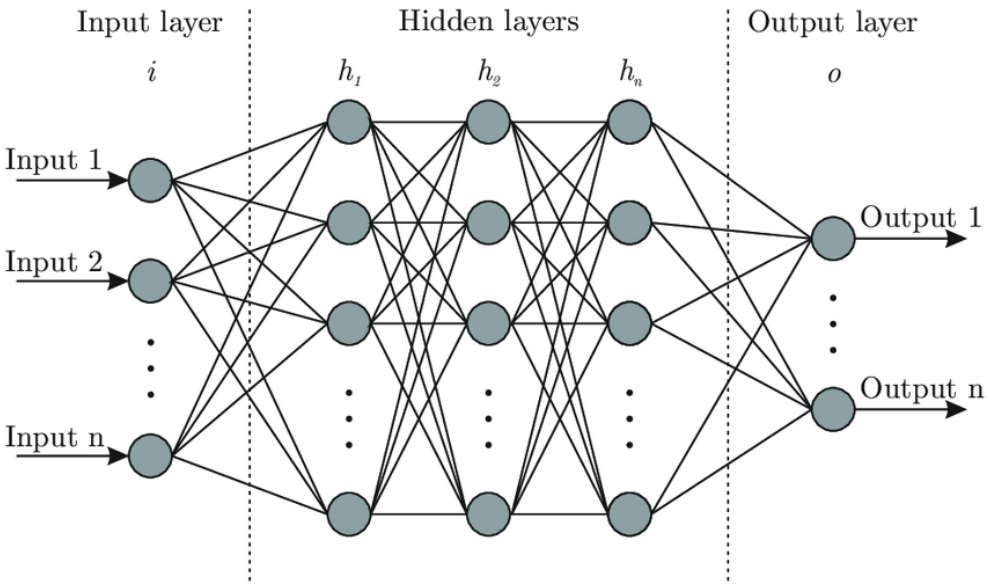



*Figure 12. Simplified schematic of a neural network architecture and its parts.*

As seen before, activations on the previous layer determine the activations on the next layer in a way that depends on the value of the connection's weight, bias and neuron's value. When looking at the whole network, it may contain hundreds of thousands of neurons and when a value changes in one of the affects to an exponential number of other neurons. This is why it is difficult to realise what each layer does and what the values stored in it represent.

What input and output layer do is easy to understand but, at the first glance, hidden layers may seem like an indecipherable black box that only contain a bunch of numbers. After several decades of work, it has been seen that what these layers may represent, for example when recognizing a digit, is the different features that form it. And by extension, each neuron represents the probability that this feature is in the image. Below there's an example with number 9 and its parts and edges.

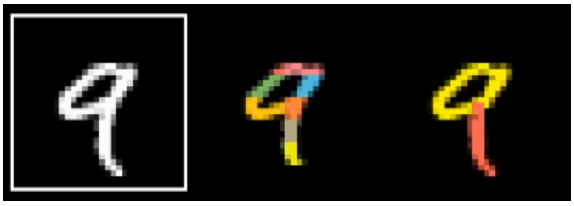


*Figure 13. Parts of a handwritten number nine painted in different colours.*

Following the same idea, the next illustration shows a really simple neural network created using the TensorFlow playground. This graphical tool allows to create a network able to classify group of points that contains two classes. As can be seen, the hidden layers detect geometrical shapes and edges that allow to classify the two types correctly.

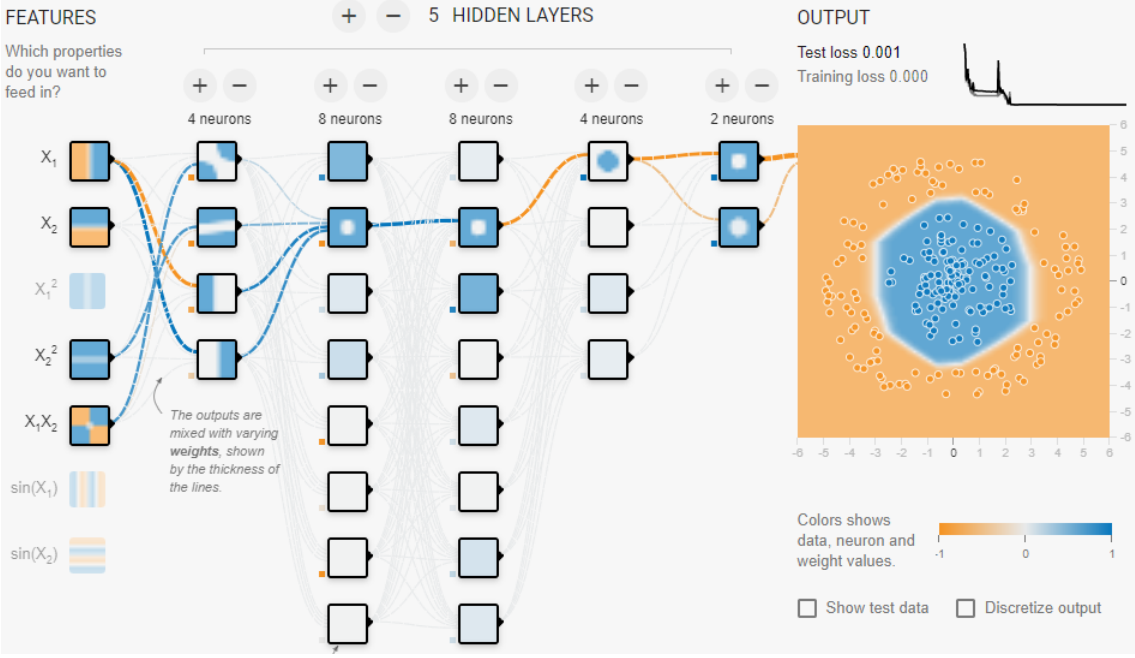



*Figure 14. TensorFlow playground capture that shows what do the hidden layers detect.*

### 2.4.3. Convolutional Neural Networks

A convolutional neural network, usually referred as CNN, is a type of neural network that contains, at least, one convolutional layer. Below an example illustration of how they look like is attached.

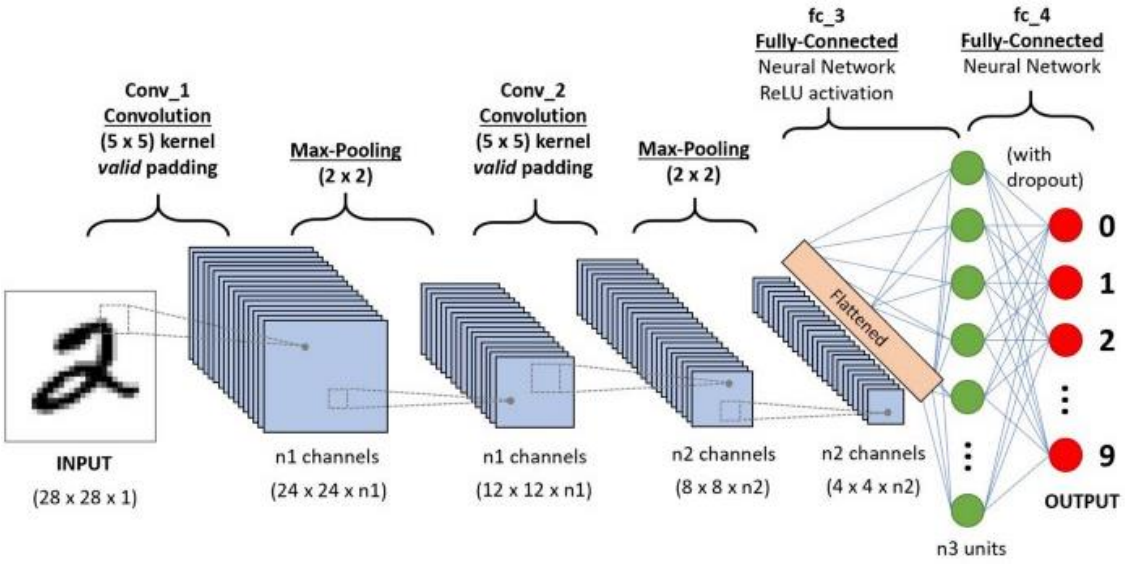



*Figure 15. Illustration showing how a CNN looks like. (Source: https://www.digitalvidya.com)*

Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Thanks to this convolutional operation, the network can be much deeper but with much fewer parameters. Moreover, these networks are able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

Due to those virtues, convolutional networks provide very effective results in image and video recognition, natural language processing, and recommender systems and are also applied in signal processing and image classification. For all the previous reasons, a CNN has been the network chosen for pupil detection in this project.

2.4.3.1. Convolutional layers

When working with a neural network that has to process images the number of parameters necessary becomes a problem, especially with large images, as it depends on the number of pixels. It is easy to realise the enormous amount of operations required if the network contains several layers. To overcome this problem the convolutional operation can be used.

The objective of the convolution operation is to extract the high-level features such as edges, from the input image. Conventionally, the first convolution layer is responsible for capturing the low-level features such as edges, colour or gradient orientation. With added layers, the architecture adapts to the high-level features as well, resulting into a network which has the wholesome understanding of images in a dataset. Below a convolution operation is shown.

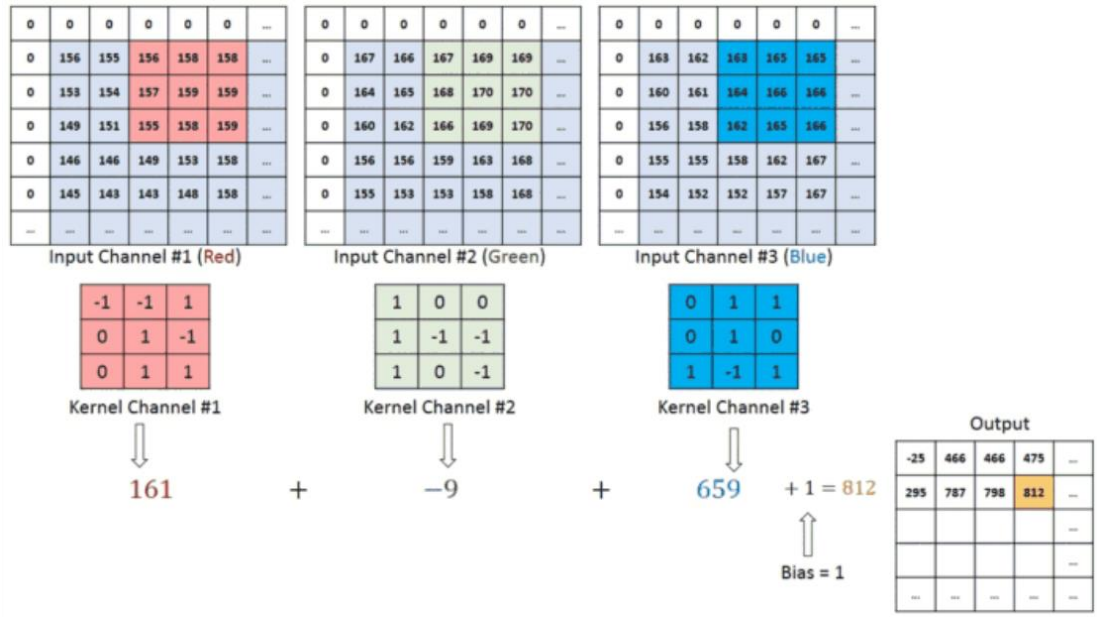



*Figure 16. Convolution operation applied to the three channels of an RGB images represented by matrixes. In the right corner can be seen how the operation results in a smaller matrix. (Source: https://towardsdatascience.com)*

During the convolution operation, the kernel moves to the right with a certain stride value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same stride value and repeats the process until the entire image is processed. In the case of images with multiple channels, for example RGB, the kernel has the same depth as that of the input image.

There are two types of results to the operation, one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying a padding value around the image. The first type of result is really useful in the case of processing images as it reduces the number of parameters in the network and, thus, reduces the computational power required to process the data and train the CNN.

2.4.3.2. Pooling layers

The pooling layer is responsible, similarly as the convolutional layer, for reducing the size of the convolved feature and, as a consequence, helps in decreasing the computational power required to process the data. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

Two types of pooling are available, max pooling and average pooling. The first one, returns the maximum value from the portion of the image covered by the kernel while average pooling returns the average of all the values from the portion of the image covered by the kernel. It can be said that max pooling performs a lot better than average pooling.

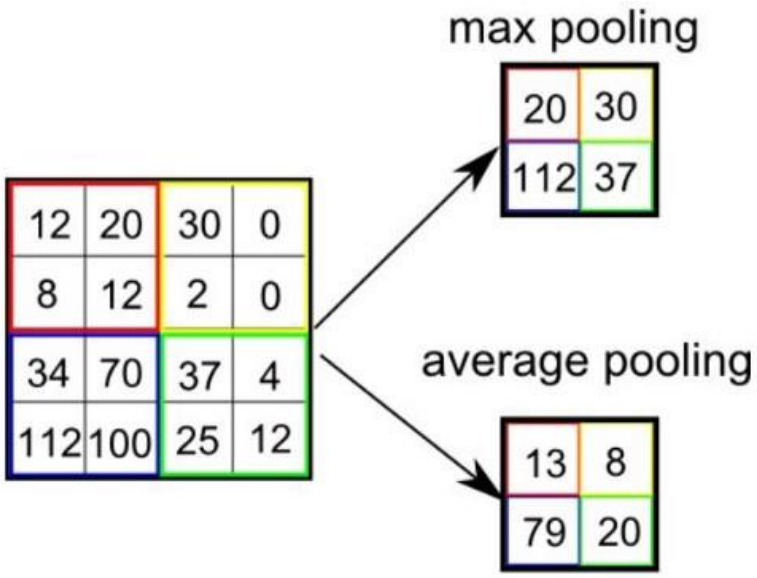



*Figure 17. Max pooling and average pooling example. (Source: https://towardsdatascience.com)*

### 2.4.4. Training process of a neural network

Once that the parts of a neural network are clear and its functioning explained let's focus on the last step before the network is ready to be used for inference, the training process.

#### 2.4.4.1. How the neural network is fed with training data

When speaking about training one has to imagine a process really similar to the way in how humans learn. In the case of this project, to train the CNN a lot of pictures, containing or not pupils, will be shown to it together with the real coordinates of the pupil in them. These pictures are grouped in what it is called batches, a bunch of pictures that are passed to the network together. The size of these batches can be any value and they are passed to the CNN several times. The number of times that the data passes through the network is referred as an epoch and training a neural network to obtain satisfactory results usually takes a large number of epochs.

The training process can be seen as an optimization process as, what happens during the training is the optimization of the weights of the connections between neurons. What it is done during this process is similar to what happens with a kid, the network has been taught how to identify pupils in a picture by showing to it a lot of labelled pupil images, this is to say, by progressively tuning the weight. The training ends when the network is optimized for the intended purpose and the measure used to quantify if this has been achieved or not is the loss function.

#### 2.4.4.2. Network initialization

An important aspect to take into account when training a neural network is the initial value that is given to its weights. As one can suspect, depending on the initial value the network its training will have a different computational cost or, even more, the model resulting from the training can have a slightly different performance.

Several discussions about this topic have been hold and nowadays there are many options available. Typically, zero initialization, which consists on setting all the weights to zero to start training and, random initialization, that proposes to give them a random value at the beginning were used. Nevertheless, in the recent years, more sophisticated techniques have arisen such as Xavier initialization or Kaiming initialization, methods that take into account the number of input and output connections in the network and offer such a better performance.

The objective of all these calculations is to obtain the best model possible, which implies finding the global minimum and skip the local ones. To achieve that is not always easy and sometimes depends on the previously explained about the initialization values as shown in the graphic below.

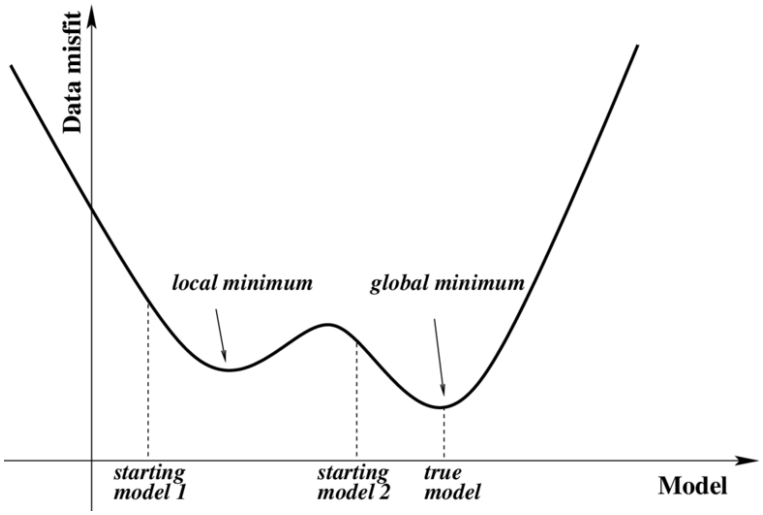



*Figure 18. Graphic showing how the starting model conditions the result of the training.*

2.4.4.3. Back propagation

So far it has been explained that neural networks have to be trained and how to do it. However, it has not been discussed yet through which mechanism they learn, it is known as back propagation.

Back propagation is an algorithm that uses gradient descent to optimize the weights of a neural network. Given an error function, the method calculates the gradient of the error function with respect to the neural network's weights.

To explain it in a simple way, the idea of this method is to feed an image to the network and check how far the result is from the true one. For example, in case of digit classification, if a picture of a two is provided to the network and the generated result says that the picture contains an 8. What back propagation would do is to calculate how the weights have to be changed so the output fits with the real number on the image.

In the following illustration a neural network being trained for digit recognition can be seen. In the output layer each neuron contains its weight when it is fed with a picture of a number two. Nevertheless, the output is not a two so there are arrows in red and blue indicating which weights have to be decreased and which ones have to be increased in order to activate the correct neuron and, hence, to provide the correct solution for the given picture. Repeating this kind of output correction hundreds of thousands of times is, in essence, what training is about.

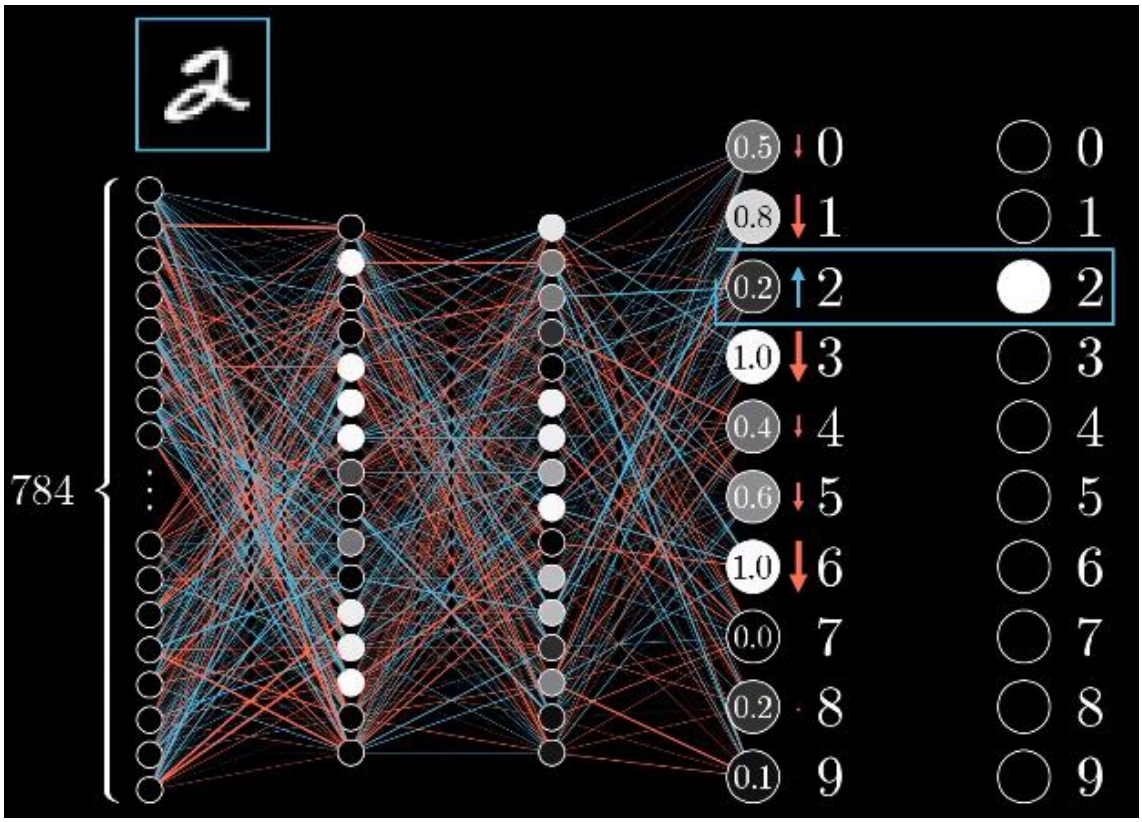



*Figure 19. Illustration of a neural network being trained to identify digits in a picture. On the right can be seen the expected result together with the given one and some arrows indicating whether the weights have to be increased or decreased.*

It is important to remember that to modify how the output layer is activated all the neural network's weights have to be modified as they are all connected between them. This, as can be expected, requires a lot of calculations and iterations and is one of the reasons why the training process takes such a long time and is directly related to the dimensions of the CNN.

#### 2.4.4.4. Training problems

Many issues can arise when training a neural network. Fortunately, there are also many techniques that help to reduce them and allow to train the desired network successfully.

These most common setbacks during the training process are normally connected with three aspects: problems in the dataset, problems with the neural network structure and problems during the training. The first type of issues can be related with noisy datasets that make the training impossible, it is important to keep in mind that the input quality has an enormous influence on the result. Another possibility is that the dataset is too small and has not sufficient examples to train the CNN, in this case data augmentation can be helpful. This technique consists in making some transformations over an image such as rotations, blur addition or cropping it partially. In the following page, an example shows how data augmentation allows to obtain multiple images from an original one without losing the feature that wants to be detected, in this case, a cat.

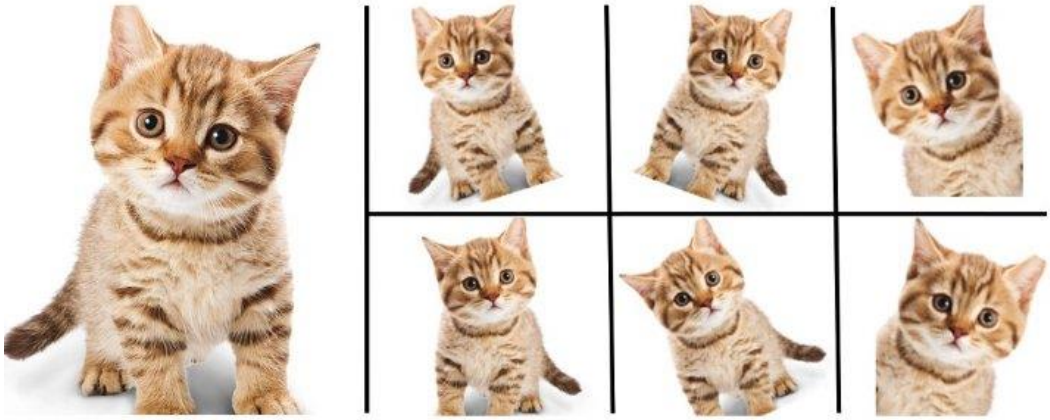


*Figure 20. Data augmentation example using a cat image.*

Regarding the problems with the network structure it can happen that the designed CNN is simply not adequate for the intended purpose, for example, due to an inadequate number of hidden layers or neurons. The solution is obvious in this case, the network structure has to be modified. Luckily many papers can be found where people share their knowledge on how a certain neural network behaves for a particular target. Taking a look on them and comparing different structure approaches before starting to implement a network is always a good practice.

Finally, concerning the problems due to the training process, is common to define an inadequate learning rate or not stopping the training on time which results in a degradation of it performance, although previously was good. Nevertheless, the most common of the issues, for which the data augmentation technique previously explained is also useful, is the overfitting and underfitting. Overfitting means generating a model that perfectly fits the input dataset but, due to its complexity, it is unable to generalize for unseen cases. In the opposite side, there is the underfitting, that occurs when a model generalizes too much and thus, it does not fit any dataset correctly. The correct approach consists in a balance of both extremes. An example of each one of these cases can be found in the graph below.

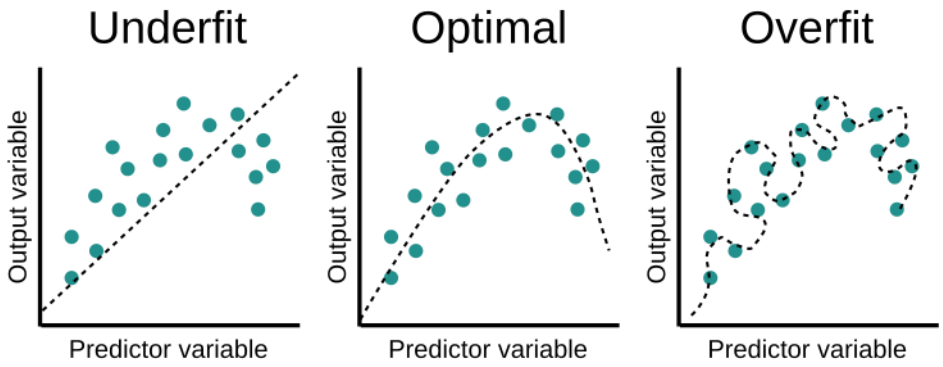



*Figure 21. Graphs showing one clear case of underfitting, overfitting and optimal fit.*

### 2.4.5. Performance evaluation

In any project that involves developing a software solution it is important to define how its performance will be quantifies. In the case of neural networks this is done using a metric called loss function. Configuring it is one of the most important steps to ensure the model will work in the intended manner. The loss function can give a lot of practical flexibility to your neural networks and it will define how exactly the output of the network is connected with the rest of the network.

Depending on the task that the neural network will develop a different type of loss function will be required since the output format will be different. This function has to be defined by de developer and will essentially calculate how poorly the model is performing by comparing what the model is predicting with the actual value it is supposed to output. If the prediction is very far off from the ground truth, the loss value will be very high. However, if both values are almost similar, the loss value will be very low. Hence, it is needed a loss function which can penalize a model effectively while it is training on a dataset.

If the loss is very high, this huge value will propagate through the network while it's training and the weights will be changed a little more than usual. If it's small then the weights won't change that much since the network is already doing a good job.

An example of how the loss value evolves along the epochs of training can be seen in the following graphic. It is worth it to analyse the graphic and realize that, if the training is working properly, the loss value decreases over time while the accuracy increases. Loss function should show similar results between the train and test sets to ensure there is not overfitting.

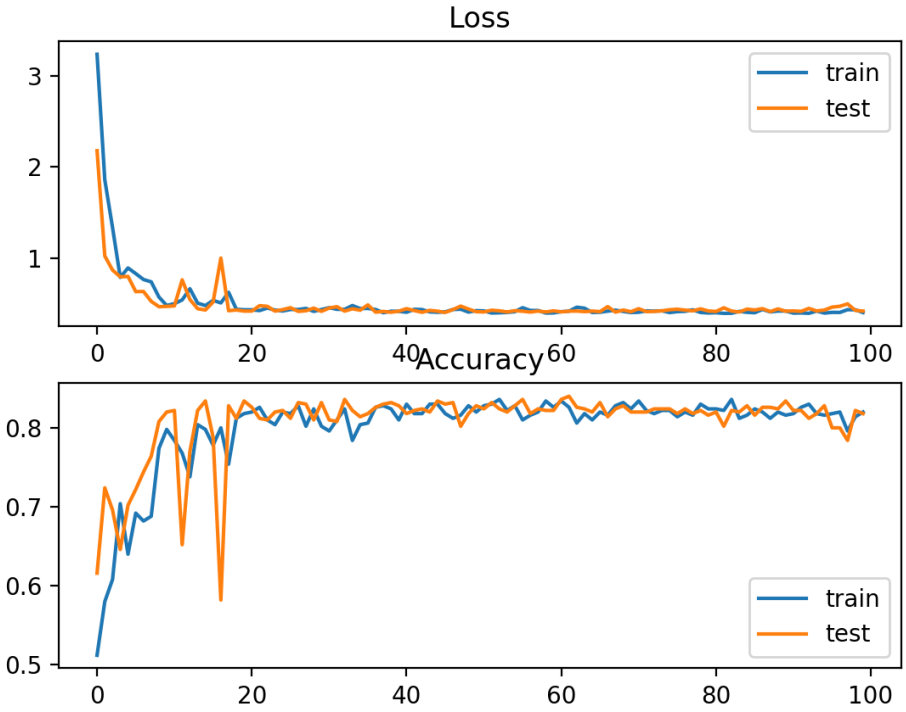



*Figure 22. Example of loss function and accuracy for train and test sets.*

### 2.4.6. Available deep learning frameworks to work with CNNs

When looking for a deep learning framework to develop a CNN the list of available options is long. TensorFlow, Keras, PyTorch, Caffe or DL4J are some of the names that appear everywhere when you start looking for the best one. Although this wide range of options, in this project the selected framework has been TensorFlow in its 2.0.0 version.

The reasons for choosing TensorFlow could sound less technical than expected but it has to be taken in mind that this project is not developing a highly complex CNN right from the start but an existing model, Google Inception V4, will be used. This makes the selected framework less relevant because, from a technical point of view, all of them allow to reach the target of the project without problems using the features they offer. Nevertheless, TensorFlow has been the selected one, basically, because there are enormous amounts of examples, tutorials and information all over the internet what makes the sometimes-difficult task of using it much easier. Also, its support for Python and the previous knowledge about it acquired in the Cloud Computing subject ended up consolidating the decision.

## 2.5. The labelled dataset concept

When working with convolutional neural networks is nearly indispensable to have datasets containing images, videos or whatever that can be used to train the CNN so it performs the desired detections. As said, a dataset is an essential part of the process to develop a CNN based solution so it must be clear what is a dataset. As a description, a dataset is a group of resources, basically images o videos in this case, that contain the object or feature that wants to be detected. Furthermore, each image or video is labelled with the true dimensions, tags, coordinates or whatever has to be detected. This allows the neural network to "learn" how to identify whatever is needed.

### 2.5.1. Labelled Pupils in the Wild (LPW) dataset

Labelled pupils in the wild (LPW) is a dataset of 66 high-quality, high-speed eye region videos for the development and evaluation of pupil detection algorithms. The videos in the dataset were recorded from 22 participants in everyday locations at about 95 FPS using a state-of-the-art dark-pupil head-mounted eye tracker. They cover people of different ethnicities and a diverse set of everyday indoor and outdoor illumination environments, as well as natural gaze direction distributions. The dataset also includes participants wearing glasses, contact lenses, and make-up. Five state-of-the-art pupil detection algorithms have been benchmarked on the dataset with respect to robustness and accuracy. It can be also useful for further study the influence of image resolution and vision aids as well as recording location (indoor, outdoor) on pupil detection performance.

Taking a look into how the dataset is labelled, each one of the 66 videos has a plain text file associated with the labels for each frame. The line where the label is placed in the file corresponds to frame number in the video. Only the centre of the pupil cartesian coordinates are labelled with a precision of two decimals which is more than sufficient for the purpose of this project.

In the following pictures can be seen several eye region images extracted from the dataset videos.

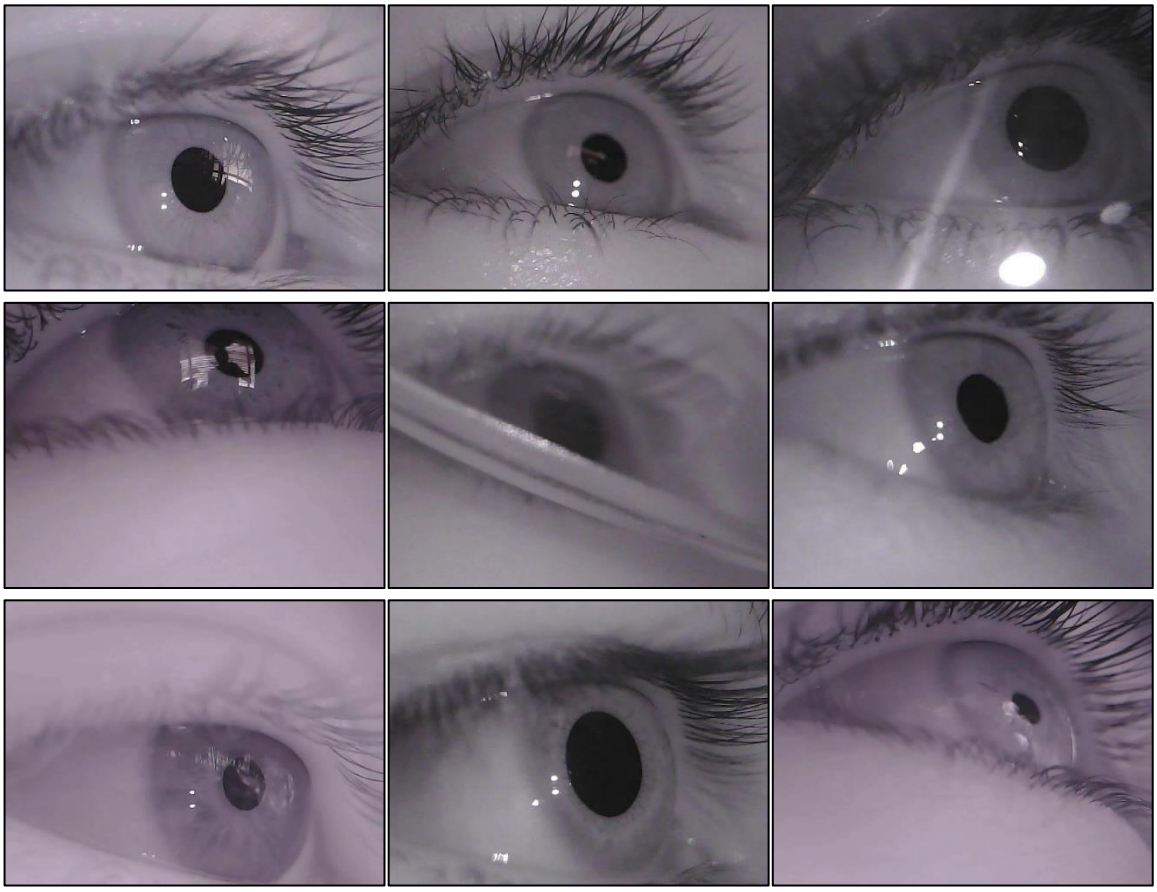


*Figure 23. Multiple eye region images extracted from videos of the LPW dataset.*

### 2.5.2. EMMA dataset

EMMA dataset is a dataset containing 17 smaller datasets of eye region images. It has been developed over the time to train and test the algorithms developed by the Human-Computer Interaction research group of the Universität Tübingen in Germany. It contains more than 38.000 images together with 17 plain text label files, one for each of the smaller datasets that compound it. The dataset has been used to test a large list of algorithms that are able to provide a really good performance such as ElSe, ExCuSe, PuRe or PuReSt, all of them developed in the same university.

Taking a look into how the dataset is labelled, each of the labels file contains the cartesian coordinates together with the image code. In this case the precision is less and only integer numbers are provided.

In the following pictures some images extracted from the dataset are shown.

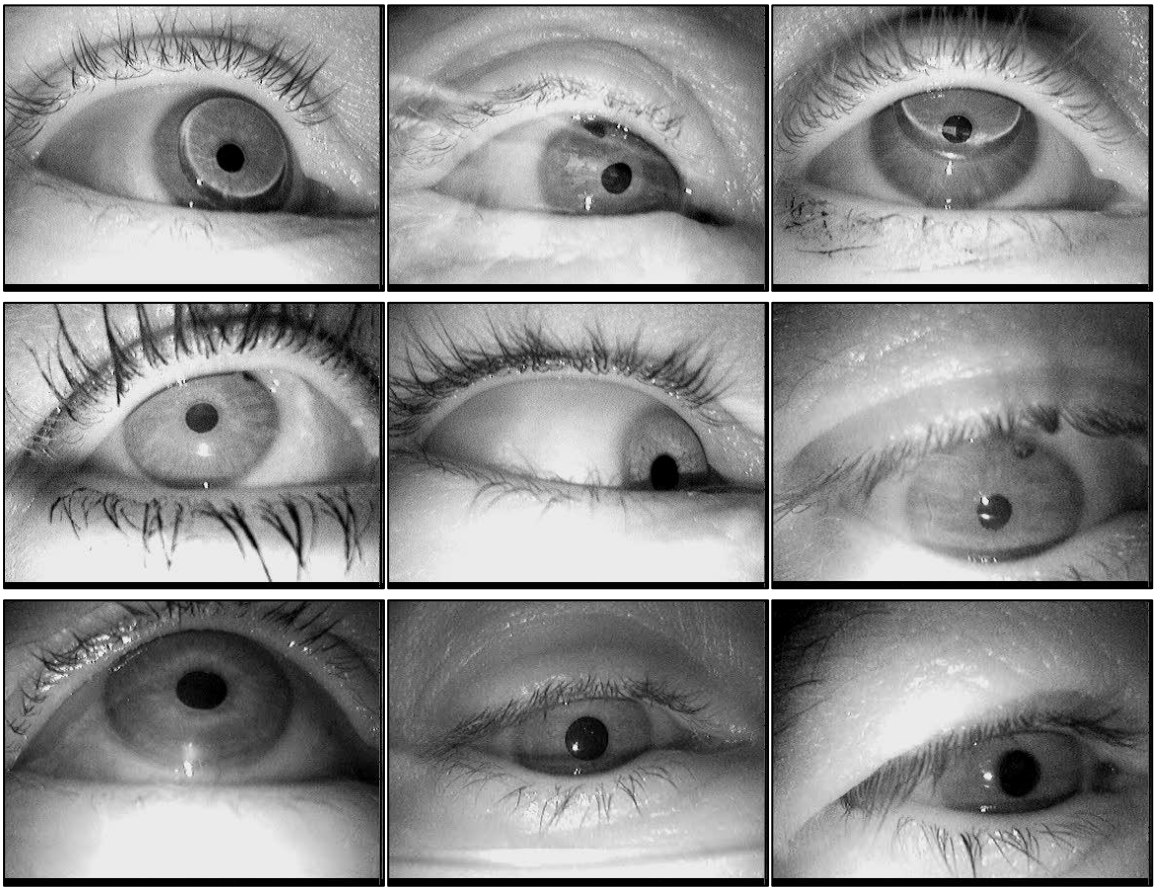


*Figure 24. Multiple eye region images extracted from videos of the EMMA dataset.*

## 2.6. Technical implementation decisions

When developing a software solution, apart from the programming aspects, other decisions have to be taken that can highly condition the results obtained from the developed solution. These important decisions, that may seem trivial sometimes, go from choosing the coding environment to selecting the programming language or electing the hardware to be used and they have to be taken in a conscious and reasoned way.

### 2.6.1. Selected programming language

Python was the selected programming language, in particular, Python 3.6. The reason to choose this language is simple, it was already chosen by the tutor on the very beginning of the project and everyone that took part on it also used the same language. As Python is compatible with the vast majority of libraries, is widely used and has thousands of libraries was, probably, the best choice nowadays.

Practically all the libraries for image processing and machine learning are available for Python, as well as most of the frameworks used to work with neural networks. All these advantages mentioned together with the enormous amount of information available on the internet made Python the language of choice.

### 2.6.2. Selected Integrated Development Environment (IDE)

PyCharm, the Python IDE developed by Jetbrains, have been the elected environment for the project. Specifically, the version used has been the Community 2020.1 as it is open source and free of charge. The reasons to choose this environment were basically two. The first one was that other students that worked in other parts of the project used this IDE so it was clear that the existent code would run without many problems on it. The second and more personal reason was that I had already experience in this environment and, thus, it was already familiar to me. Also, it offers all that is necessary for me, competent debug options, the possibility to install extra plugins, code completion, a Python terminal and a console so, in some way, was not necessary to make further research on this point as the existent IDE was already fine for the purpose.

### 2.6.3. Selected operating system and virtual machines

Once the programming language and IDE were clear the operating system was the next choice to do. After several attempts of working in Windows and facing continuous problems when installing the required libraries Linux, particularly Ubuntu 18.04, was the operating system of choice.

The reasons for this choice were that, first of all, the compatibility problems when installing libraries were solved with this change and, as the IDE is also available for Linux, moving to this new OS was not a big deal. This movement also opened the door to work with a virtual machine as the host OS on the laptop was Windows 10.

Working with a virtual machine really made a difference as allowed to work in a complete separated environment where making tests and installing new software was much less dangerous. Moreover, with virtual machines it is possible to work in parallel effortless as they can be duplicated easily, only further hardware resources are needed, option that has been useful in the latest steps of the project. The virtual machine software used was Oracle VirtualBox 6.1.

### 2.6.4. Working with virtual environments

In Python, a common strategy is to create a separated environment for every project where the libraries required for it are installed so let's focus on what a virtual environment is.

According to the Python website:

> *"A virtual environment is an environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in other virtual environments, and (by default) any libraries installed in a "system" Python, i.e., one which is installed as part of your operating system. A virtual environment is a directory tree which contains Python executable files and other files which indicate that it is a virtual environment."*

Also, it has to be said, that virtual environments include a set of commands to manage them. To work with a virtual environment this has to be activated, once it is no longer needed it can be deactivated and switch to another one.

An especially useful software to work with virtual environments and install big amounts of packages is Anaconda. According to Wikipedia:

> *"Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc."*

Anaconda presents a really good advantage, all the packages included in it are free of conflicts between them, a problematic that can make you to waste a lot of time as happened in the beginning of the project.

### 2.6.5. Working with GIT

In order to work in an efficient way some additional software has been used. The most relevant one has been GIT, in particular the GitLab tool that provides a repository manager and other associated services for free on the cloud. But first of all, what it has to be explained what is GIT used for.

According to Wikipedia:

> *"Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows."*

In this case, GIT has not been used so much to coordinate but for version control as only one programmer was working on the project. GIT offers, as said, an easy way to track changes in files and the possibility to store the files in the cloud so it acts, somehow, as a security back up. Finally, another advantage that can provide is the possibility to access the files from another device, this feature was useful to share the work advances with the supervisor.

# CHAPTER 3.
# Solution based on a traditional algorithm: Starburst method

The first solution tried for pupil detection in an image is based on a traditional algorithm. As explained in the previous chapter, a traditional algorithm is a finite sequence of states that perform a specific task, in our case, detect pupils in an image. In this section a detailed explanation of how the selected traditional algorithm works, its implementation and testing as well as an assessment of its pros and cons.

## 3.1. How the Starburst algorithm works

### 3.1.1. Global view of the algorithm functioning

As said above, the algorithm designed is based on the Starburst method. This it can be divided into four perfectly defined parts, each of which performs a specific function within the pupil detection process. As an introduction, below a graphic containing a brief explanation of each piece can be found and, later, in the following sections each one of them will be explained in detail.

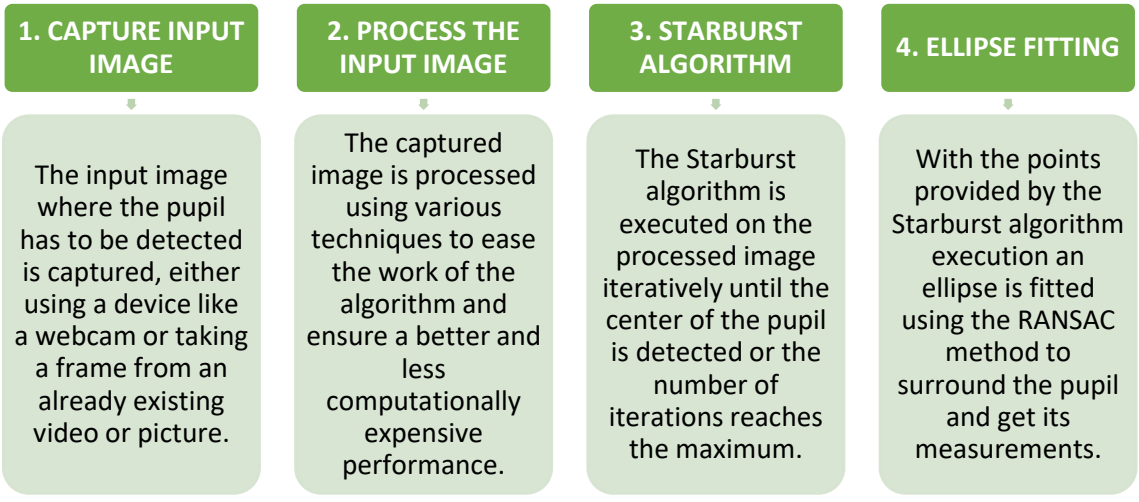



*Figure 25. Schematic showing the four stages of the designed traditional algorithm based on the Starburst method.*

### 3.1.2. Capture of the input image

As is obvious, to detect a pupil in an image first an image is needed. In our case, to obtain this image two different ways have been considered. The first one implies capturing the image from a device like a webcam. This method allows to obtain infinite images but cannot assure that the input has the enough quality unless the light and distance conditions are perfectly controlled which almost never happens in the practical if the capturing device is not head-mounted as in this case. Using a webcam is the indicated way once the algorithm is fully operational and it will be used for anti-spoofing detection as this project is intended to work in the real world.

The second method considered to obtain an image is retrieving it from an already existing multimedia file as a picture or a video. This way allows to provide to the algorithm a known image, this is to say, an image in which it can be previously verified that it contains a pupil and has a good quality. Moreover, this gives the programmer the opportunity to repeat the detection as many times as it is needed as the picture is saved in a file. This advantage is useful during the programming and tuning of the algorithm as makes understanding of what the code is doing easier and allows to find possible mistakes.

Also, it has to be said, that the algorithm is ready to work with a single image or a video, which is, in practice, a series of images one after the other.

### 3.1.3. Process the input image

Once an image is obtained using one of the methods explained in the previous point various image processing techniques are applied to it. These techniques are intended to ease the detection of the pupil or pupils in the image. The following graphic shows, in the correct order of application, the image processing methods applied in our case.

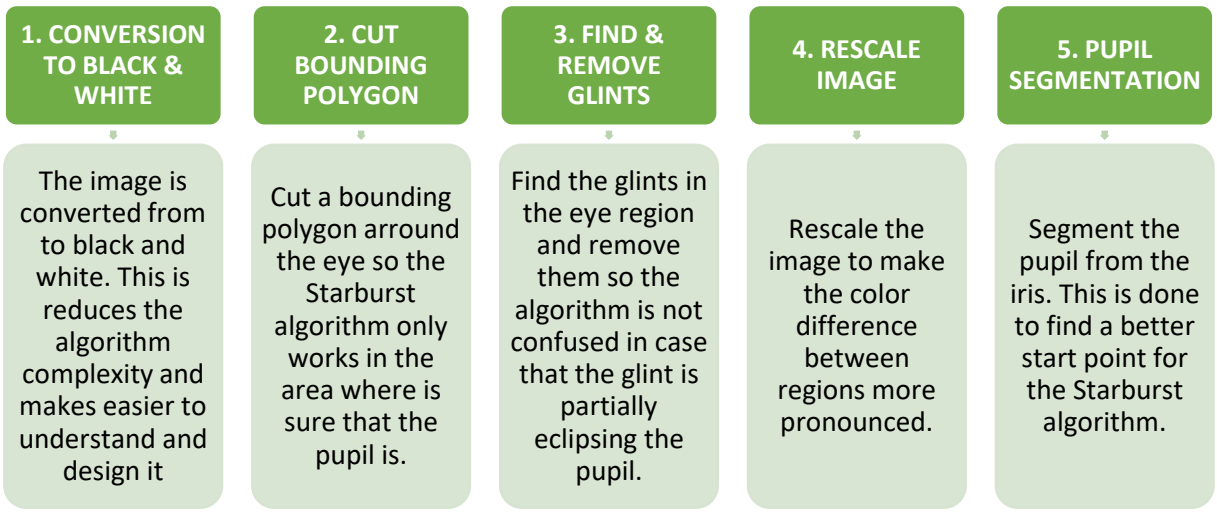



*Figure 26. Image processing techniques applied to the input image.*

3.1.3.1. Convert image to black and white

The first transformation applied to the input image is to convert it from a 3-channels colour scheme to a black and white scheme. This is done because the Starburst algorithm is based on the difference of value between one pixel and another one, then, only one value is needed per pixel instead of three values in the case of RGB.

Below two eye images can be seen showing how the same image changes when it is converted to black and white.

*Figure 27. Comparison between the original image (left) and the black and white one (right).*

And also, a detailed view of the values stored in the pixels of each of the pictures.

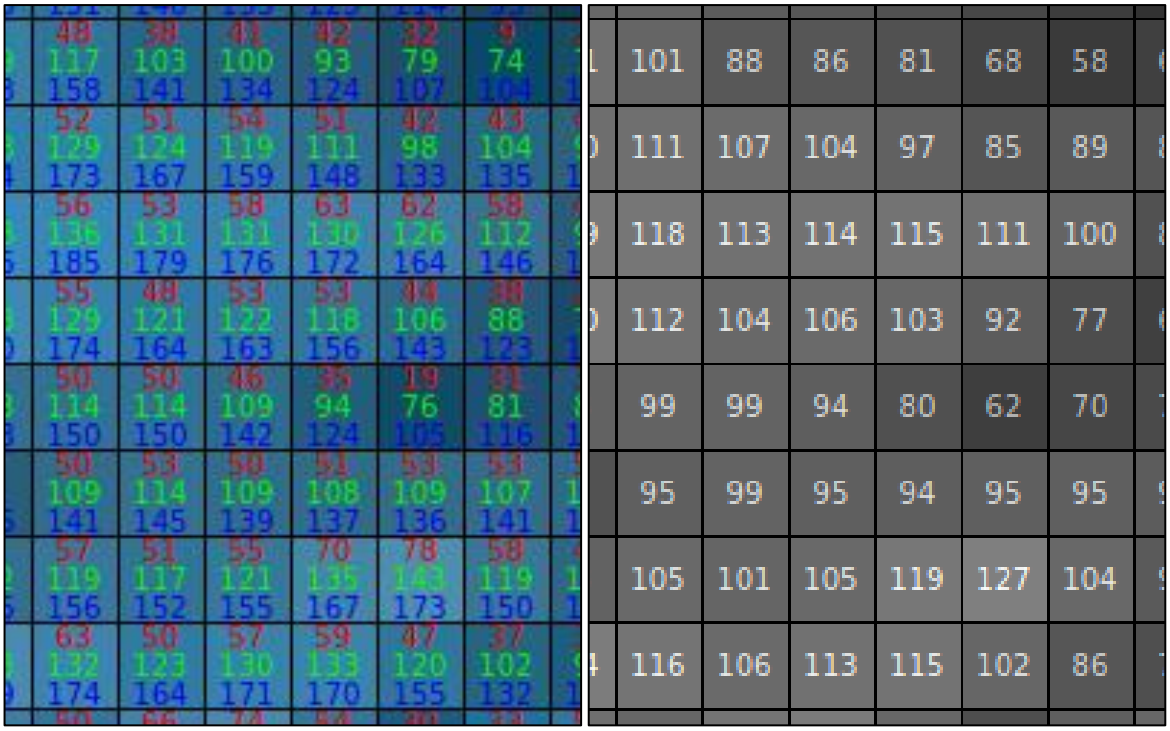



*Figure 28. Comparison between the values stored in a 3-channel colour image (left) and the pixel value in the same image converted to black and white (right).*

3.1.3.2. Cut bounding polygon encapsulating the eye region

The second transformation applied to an image is to cut a bounding polygon around the eye region. This is done because the input image can contain a whole face, which contains a lot of features that can make the pupil detection much more difficult and inaccurate or even prevent it. What is done then is to detect the eye region using the landmark method seen in the previous chapters and to take only this part of the image which will be named region of interest or ROI.

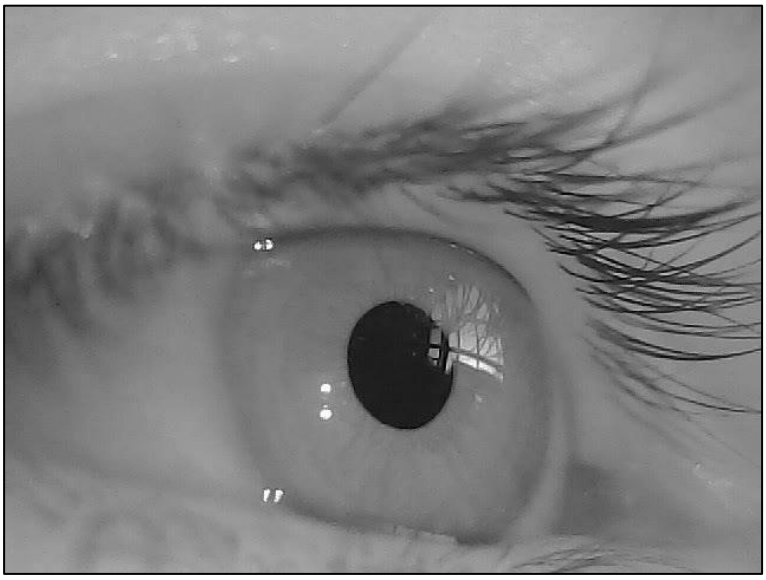


*Figure 29. Example image only containing the eye region.*

It is worth it to say that if the input image contains a face the algorithm will be executed twice, once per eye, as it is much easier to detect only one pupil than to attempt to detect both at the same time.

3.1.3.3. Find and remove glints

The third transformation applied to the image is aimed at eliminating glints that could hinder the pupil detection. This is done by identifying the bright regions of the picture. To be marked as a glint candidate the region of the image has to meet a group of conditions defined using a configuration file. First of all, the diameter of the candidate has to be bigger or equal to the 2% of the image width in pixel. Once this requirement is met, the area of the possible glint is checked, to be validated as a glint has to meet the following condition:

$$\frac{1}{3} * \left(\pi * \left(\frac{D}{2}\right)^2\right) < A_{glint} < 3 * \left(\pi * \left(\frac{D}{2}\right)^2\right)$$

Where the diameter D is the one specified before as the 2% of the image width.

After, a dilation with an elliptical kernel is applied to the bright blobs using a function from the OpenCV library to smooth the contour of the blobs and ensure that they are removed completely. Finally, all the candidates that have been found are measured and removed one by one if they have a circularity bigger than 50% by creating a glint mask that can be seen in the following figure.

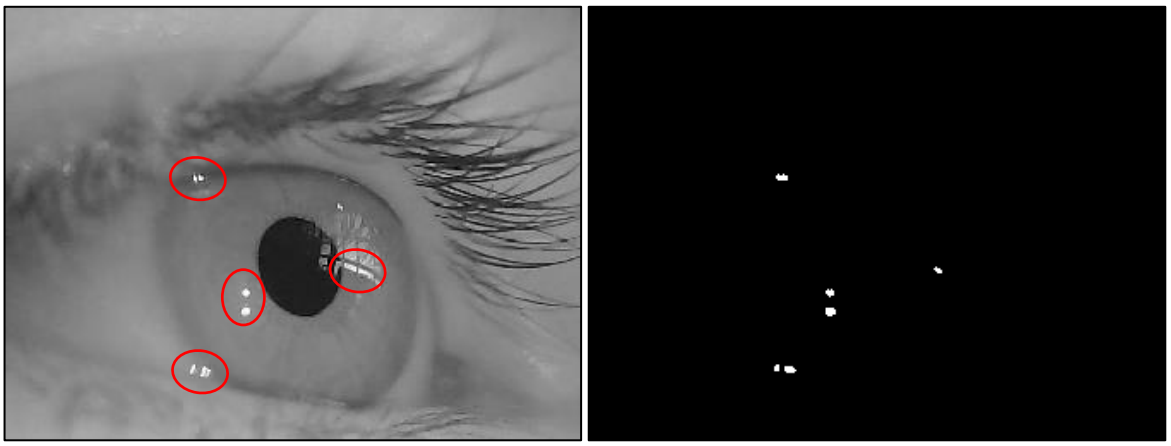


*Figure 30. Original image containing the eye region and the glint candidates circled in red (left) together with the glint mask generated from it showing in white the glint candidates that will be removed in the original image (right).*

The blobs that have been identified as a true glint by meeting all the conditions previously explained are then removed by using the inpaint technique. This method is. already incorporated in the OpenCV library and consists on replacing each glint pixel by normalized weighted sum of all the known pixels in the neighbourhood. In the following image can be seen how the glints are clearly removed.

*Figure 31. Comparison between the original image (left) and the same image after removing the glints detected (right).*

3.1.3.4. Rescale image

The fourth transformation that is applied to the eye region picture is the reescalation. This consists on modifying the intensity of the image to make the blacks more evident, this is to say, increase the difference of value between the pixels by maintaining the black ones as black and turning whiter the grey ones. This is done by using percentiles and the Scikit Image library, the result of the transformation can be seen below.

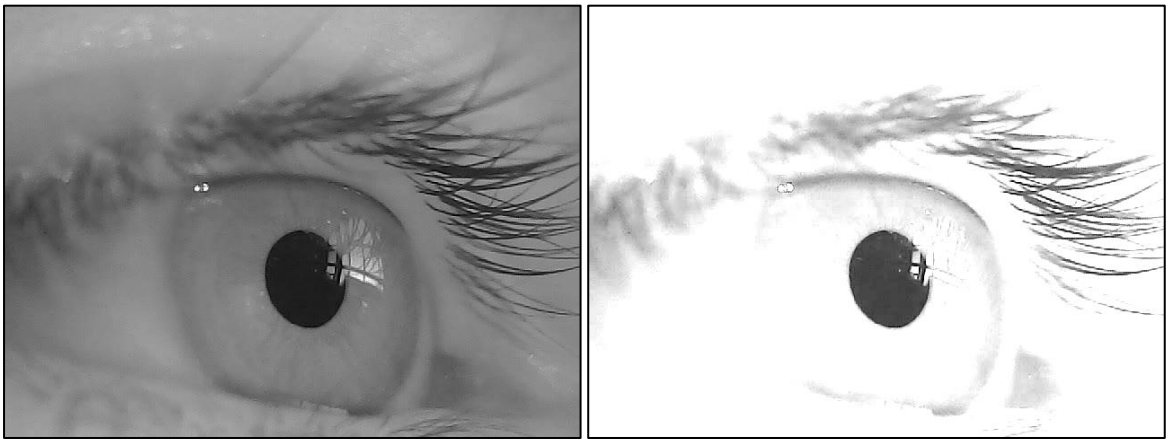


*Figure 32. Comparison between the original image (left) and the rescaled one (right).*

3.1.3.5. Pupil segmentation

The fifth and last transformation applied to the image is the pupil segmentation. This consists on limiting the pupil to know its centre. This is done once the picture is clean of glints and correctly rescaled to prevent the algorithm confusing a glint with the pupil.

The conditions used to determine if it is a pupil or not are the same than before but changing the target regions from white to black. Also, in this case, instead of choosing multiple regions as pupil candidates only one is selected as is impossible there are more than one in the same eye.

Although this transformation provides good results in general, sometimes, as some of the configuration parameters are added manually, it is not able to detect the pupil. For this reason, the Starburst algorithm is the one selected for detecting the pupil instead of pupil segmentation as the first one is able to provide better results more often. Due to that pupil segmentation is only applied to provide to the Starburst code an initial point and boost its performance. Below the result of this method are shown.

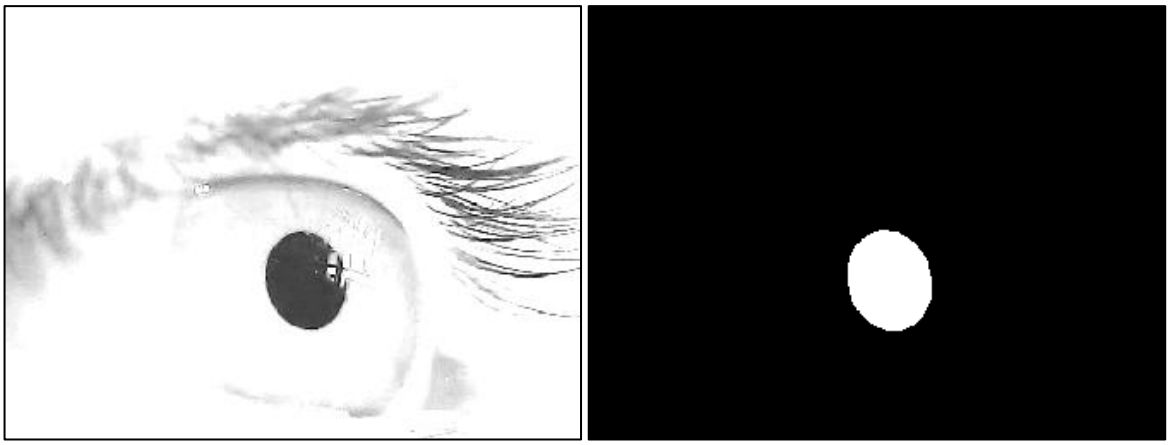


*Figure 33. Input image rescaled (left) together with the pupil mask with the true pupil detected (right).*

### 3.1.4. Execution of the Starburst algorithm

Once the input image is processed using the techniques previously described the Starburst is ready to be executed. The following points explain in detail all the functioning.

#### 3.1.4.1. Parametrization of the Starburst algorithm

In order to make easier to parametrize the designed algorithm a configuration file has been created. In the following table some of the parameters defined on it can be seen.

| PARAMETER | DESCRIPTION | SELECTED VALUE |
|---|---|---|
| method | Method used to fit an ellipse to the pupil using the points provided by the Starburst algorithm. | ROBUST_LSQ |
| starburstparams | Calculation method of the input parameters to the algorithm, manually or calculated by the algorithm itself. | manual |
| starburstedgethres | Threshold required to consider a point as an edge point, this is to say, the minimum colour difference between one pixel and the previous one. | 22 |
| starburstraylen | Length of the ray. In practice this means the number of pixels between evaluations of whether the edge threshold has been reached or not. | 1.0 |
| starburststartvar | Minimum variation in pixel between consecutive start points to keep iterating. | 5 |
| maxiterations | Maximum number of iterations performed independently if the variation between start points is small enough or not. | 5 |
| maxrefinements | Maximum ellipse inlier refinements performed. | 5 |
| maxinlierperc | Maximum inlier percentage of total points for convergence. This means the maximum number of points in the ellipse contour to prevent overfitting. | 95.0 |

*Table 1. Some of the configuration values used to parametrize the Starburst algorithm.*

3.1.4.2. Steps of the Starburst algorithm

First of all, to execute the Starburst algorithm the original eye image has to be passed to it. Then, with the parameters, either manually defined or calculated using a programmed function, the start point is calculated. From this point rays of the specified length are thrown in the calculated direction by defining the number of angles used. Every time the intensity derivative is calculated between the start and final points of the ray. The ray keeps the moving towards the direction until the derivative is greater than the designated threshold. When the derivative value is bigger than the threshold a feature point is placed and the march of the ray is halted. Once a feature point is placed, the ray is out the image or the maximum ray length has been reached the algorithm starts marching along the same ray but in the opposite direction returning towards the starting point. In this stage, also the intensity derivative is calculated and a feature point is placed in case that it is bigger than the threshold.

This process is repeated iteratively until the maximum number of iterations is reached or the calculated start point converges, with a small tolerance, with the previous start point. The final result of the execution of the Starburst algorithm will be a group of feature points that will be used in the last step to fit an ellipse using the RANSAC method and, and thus, obtain the dimensions and the true centre of the pupil.

All the process explained in the previous paragraphs can be understood easily in the following pseudocode.

***Input:*** *Eye image already processed (rescaled, glint removed…)*
***Output:*** *Set of feature points*
***Procedure:***
*Iterate (till maximum number of iterations defined)*
    *Stage 1:*
    *Follow rays extending from the starting point*
    *Calculate intensity derivative at each point*
    *If derivative > threshold then*
        *Place feature point*
        *Halt march along ray*
    *Stage 2:*
    *For each feature point detected in Stage 1*
        *March along rays returning towards the starting point*
        *Calculate intensity derivative at each point*
        *If derivative > threshold then*
            *Place feature point*
            *Halt march along ray*
    *Starting point = distance weighted mean of feature points*
*Until new starting point converges with the previous one*

*Figure 34. Pseudocode showing the functioning of the Starburst algorithm.*

The result of the algorithm can be seen in the following left image of a pupil after executing the code using it as input. The start point is represented in green, the feature points in red and the rays thrown from the start point in blue.

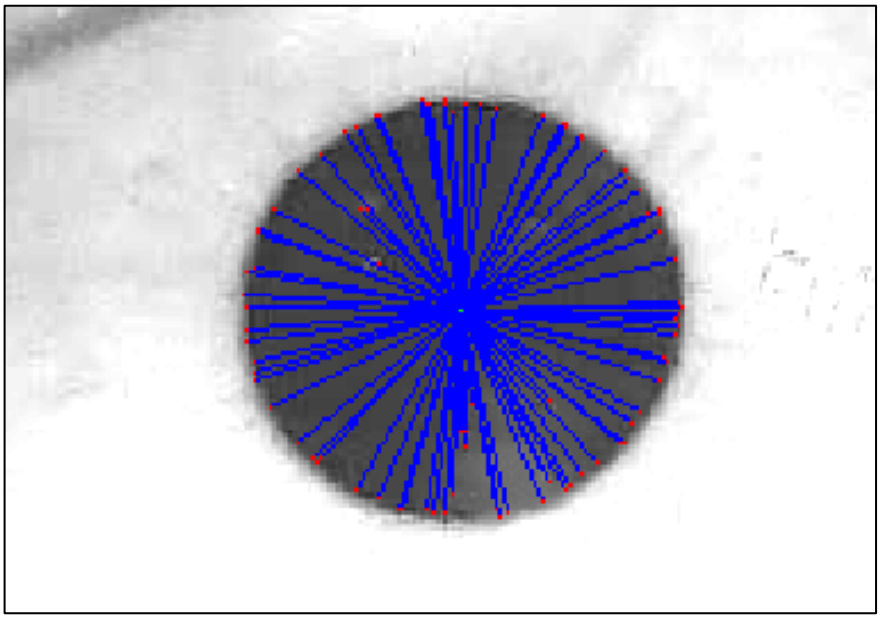


*Figure 35. Result of running the Starburst execution with the representation of the start point (green), the rays thrown from it (blue) and the feature points found (red).*

Below four iterations of the Starburst algorithm over the same eye image are represented.

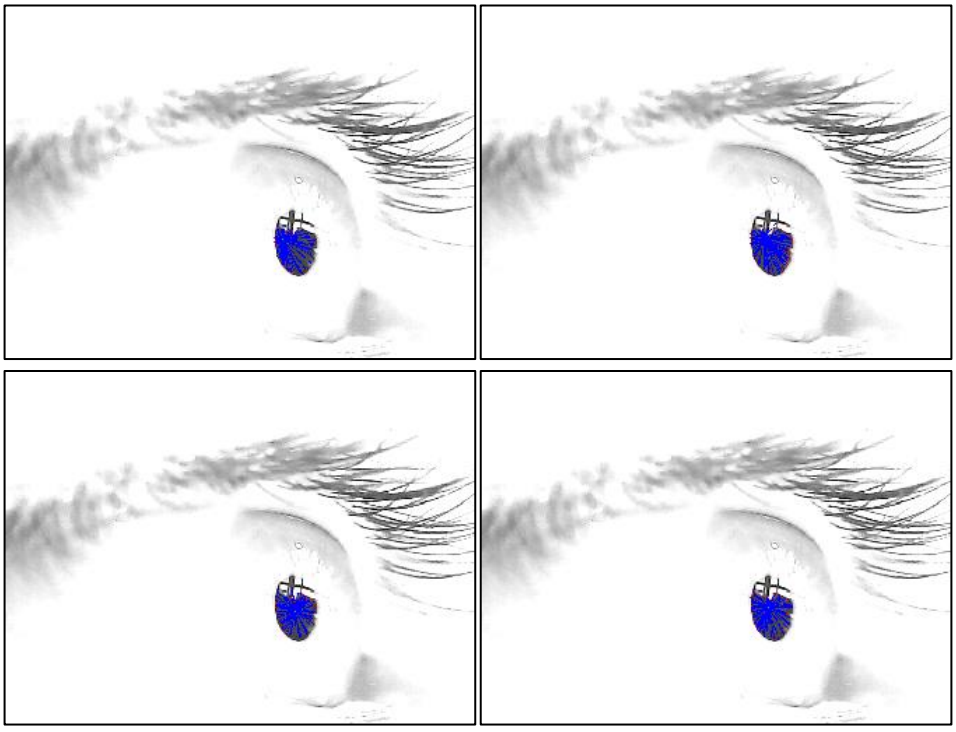


*Figure 36. Four iterations of the algorithm over the same eye image.*

Finally, the image below shows how the algorithm approximates more and more to the centre of the pupil with each iteration. The final iteration is represented in blue while the yellow colour represents the previous start points and, thus, see the improvement along the process.

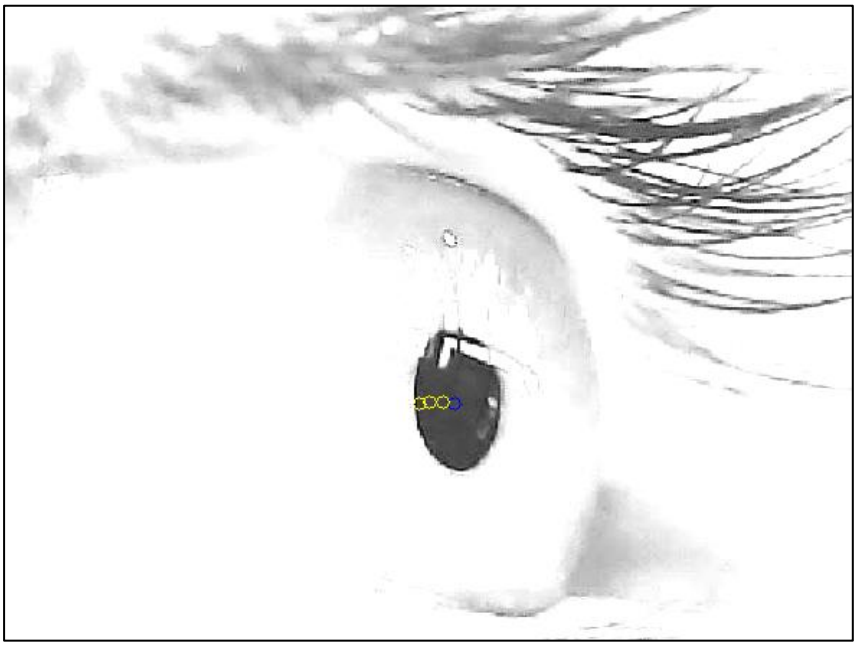


*Figure 37. Displacement of the start point along the iterative process in the same eye image.*

3.1.4.3. Contribution of image processing to the Starburst algorithm performance

It has to be said that the techniques showed in the previous point are not necessary to be performed before running the Starburst algorithm in an image but they make an important contribution to the performance of it. The following pictures show the difference of running the algorithm with and without glint removal where is easy to appreciate a worst result on detecting the true edge of the pupil when it is not applied.

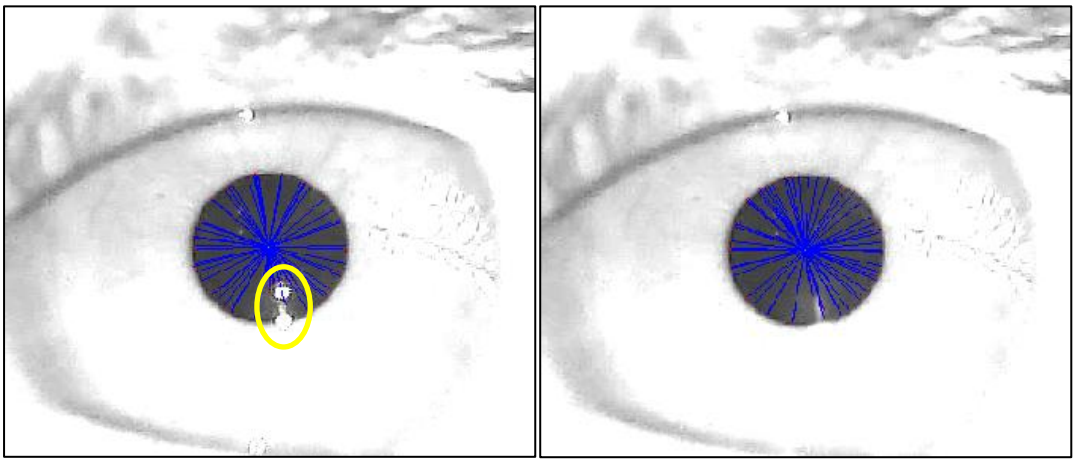


*Figure 38. Performance of the Starburst algorithm without glint removal (left) and performance with glint removal (right).*

As can be seen in the chart comparison above the Starburst algorithm do not require of a previous image processing process. Despite that, in the real-world applications, some image quality problems appear that, together with difficulties of parametrizing the algorithm to work well in most of the situations, make the use of them highly recommended.

### 3.1.5. Ellipse fitting using RANSAC method

The objective of running the Starburst algorithm is to obtain a group of feature points that represent the edges of the pupil. Once these points are determined the algorithm goes to the next step, the ellipse fitting using the RANSAC method.

#### 3.1.5.1. What is Random Sample Consensus (RANSAC) method

The method known as Random Sample Consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data containing outliers that are not intended to influence the model. In this case the mathematical model to be estimated is an ellipse. Below an example of RANSAC application to estimate a straight line is shown.

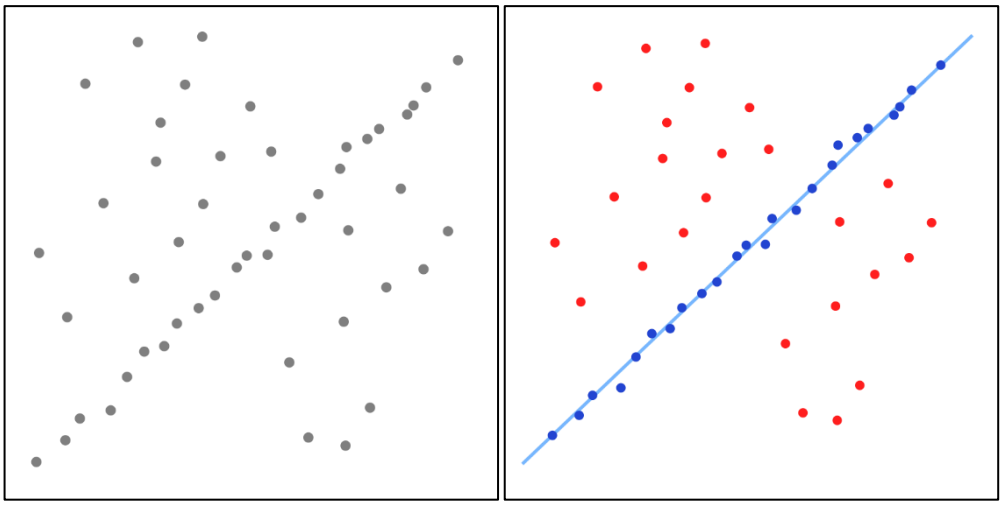


*Figure 39. Graphical representation on how RANSAC method works given a group of points. (Source: By Msm - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2071406).*

#### 3.1.5.2. Results of ellipse fitting using the RANSAC method

After applying the RANSAC method to the group of feature points, in the ideal case, an ellipse fitting the pupil perfectly should be provided. The ellipse can be represented mathematically using the following formula.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$where\ x, y = centre\ coordinates;\ a, b = semimajor\ and\ semiminor\ axis.$$

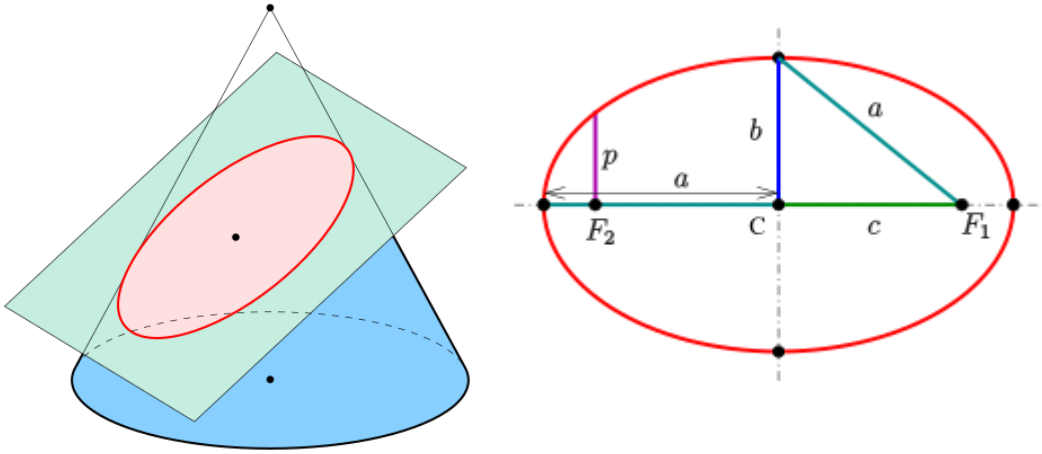



*Figure 40. Ellipse represented as a conic section (left) together with its parameters (right).*

Below can be seen the subsequent result of fitting an ellipse to a pupil in various eye images when the found points are, mostly, true feature points. The quality of the results here is considered as high due to the good fitting of the ellipse to the pupil.

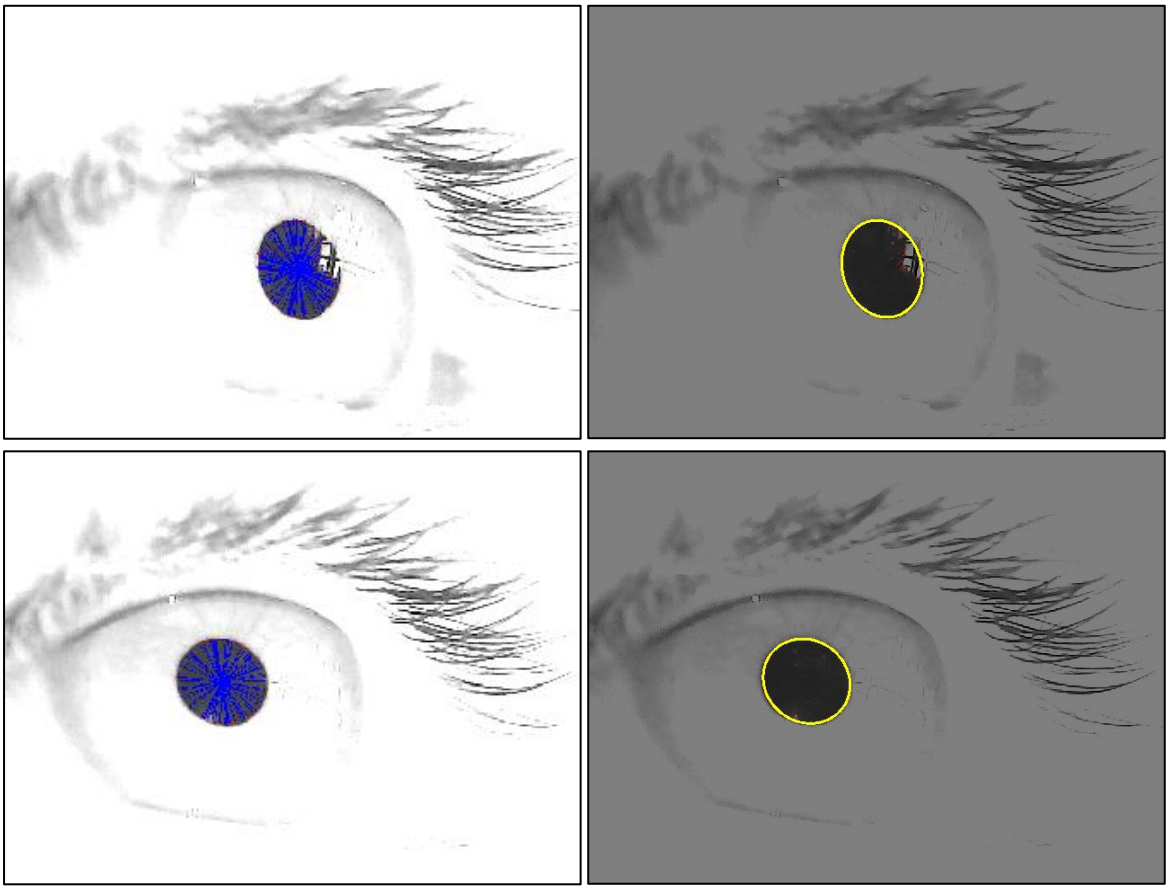


*Figure 41. Starburst execution result (left) together with the ellipse fitted using the RANSAC method in yellow and the feature points in red (right).*

Nevertheless, some times the algorithm does not perform a good work and incorrect results are generated as will be shown in the next figure. This is, in the vast majority of times, due to a bad finding of the feature points that leads to a bad ellipse fitting.

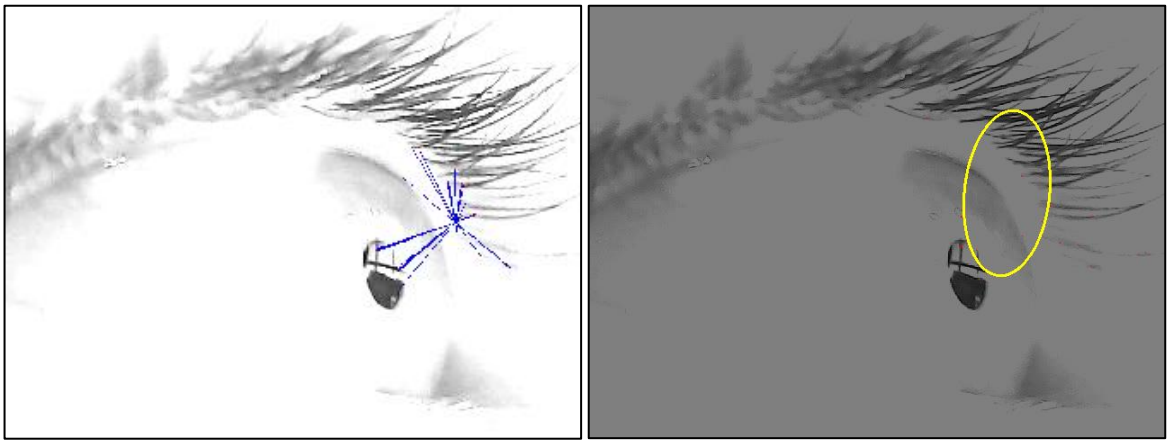


*Figure 42. Bad feature points provided by the Starburst algorithm (left) leading to a bad ellipse fitting (in yellow) using the RANSAC method (right).*

It is easy to see then that when the foundations are not solid enough, this is to say, the product of the Starburst algorithm, the ellipse fitting is also not correct. These results will be analysed and compared with further detail in the following sections in order to evaluate the designed solution.

## 3.2. Implementation of the Starburst algorithm

Now that the functioning of the Starburst algorithm and all the image processing techniques and ellipse fitting methods associated to it have been explained, the technical implementation aspects will be exposed in detail.

### 3.2.1. Libraries used to implement the algorithm

In this case, only a list of the principal libraries used to develop the pupil detection algorithm is attached as the programming language and software were already explained in the previous chapters. Also, in the annexes a complete list of libraries can be found to be able to run the whole project without any problem.

| LIBRARY | DESCRIPTION | VERSION |
|---|---|---|
| configparser | Configparser is a Python module that implements a basic configuration language. It is used to write Python programs which can be customized by end users easily.<br>In this case it has been used in the configuration file employed to parametrize the Starburst based algorithm. | 3.8 |
| openCV | OpenCV is a library of programming functions mainly aimed at real-time computer vision. It allows the user to read and write images or videos and to apply to them some image processing techniques as well to display them among many other functionalities.<br>In this project it has been used mainly for input/output of multimedia files, to visualize the pictures and apply basic transformations. | 4.1.0 |
| numpy | NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.<br>This library has been used basically to work with arrays and simplify some mathematical operations that are already built-in on it. | 1.17.0 |
| scikit-image | Scikit-image is an open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, colour space manipulation, analysis, filtering, morphology, feature detection, and more.<br>In the project it has been used for image transformations and image processing. | 0.16.2 |

*Table 2. Table containing a list of libraries used to implement the pupil detection algorithm based on the Starburst method with a brief description of each one.*

## 3.3. Results provided by the Starburst based algorithm

After the execution, the algorithm provides an ellipse for each eye image as described in the previous sections. Mathematically, once you know the parameters of the ellipse, you can easily extract its centre coordinates and, this value, is the interesting value in order to determine the gaze orientation. This value is the one that has to be used in the parts that follow this project, as explained in the beginning of this document, so it is needed to evaluate how good is this value. To do so, a mathematical criterion has to be defined.

In the next page, an eye image with the detections performed by the Starburst based algorithm can be seen. From a graphical point of view the detection has a good accuracy but to be able to say so it has to be evaluated in a numerical way. How this works is what will be explained in the next section.

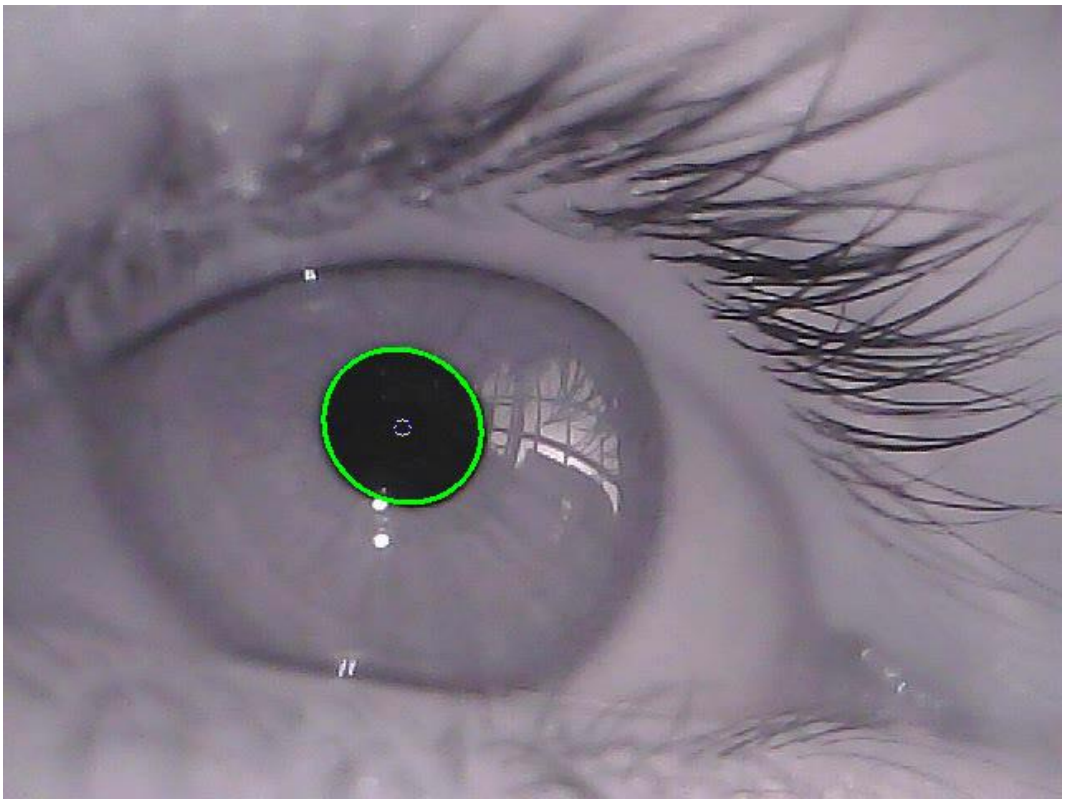


*Figure 43. Result provided by the Starburst based algorithm showing the fitted ellipse (green), the detected pupil centre (blue) and the ground truth from the dataset (yellow).*

### 3.3.1. Selection of the results evaluation criteria

A long search effort has been done to find a widely used method for evaluation and it provided a candidate method, Mean Average Precision (mAP). Although at the first view the mAP looked promising as it is a standardized method, widely used and able to provide good results evaluating accuracy of detections finally was discarded because it is too complex for the problem to solve and the available information.

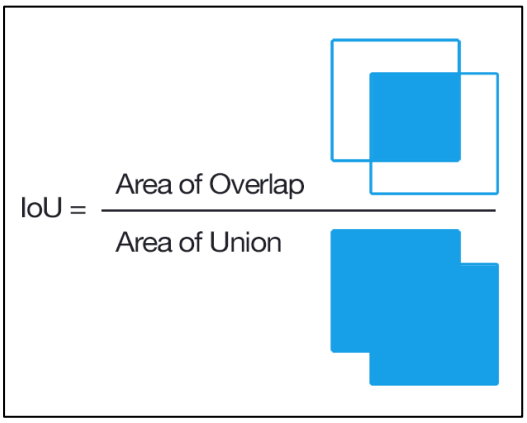



*Figure 44. Intersection over Union (IoU) formula which is the base of the mAP algorithm.*

The mAP method is based on the comparison of areas using the IoU, as shown in the image on the left, between the detected object and the ground truth. That would be suitable in case we could compare ellipse areas but the datasets available only have the centre of the pupil labelled so there are no areas to compare, only two points.

For the reasons explained before, a new criterion to evaluate the accuracy of the detection had to be selected, the Euclidean distance.

3.3.1.1. Euclidean distance method

The Euclidean distance method offers an extremely easy implementation, a practically inexistent computational cost and, the most important thing, a reliable way to calculate the accuracy. Its functioning is so simple, the accuracy is the distance in a straight line from the detected centre of the pupil to the ground truth or vice versa.

The formula used to calculate the metric, and for instance, the accuracy, is the following one.

$$Euclidean\ distance = \sqrt{\left(x_{det} - x_{gt}\right)^2 + \left(y_{det} - y_{gt}\right)^2}$$

$$where\ (x_{det}\ , y_{det})\ is\ the\ detected\ pupil\ centre\ and\ \left(x_{gt}\ , x_{gt}\right)\ is\ the\ ground\ truth\ labelled\ in\ the\ dataset$$

In the couple of pictures below is easy to see that the accuracy of the detection on the left picture is much better than on the right one. In the same way, the distance between the detected centre of the pupil and the ground truth, represented by the blue and yellow points respectively, is greater on the right picture than on the left one.

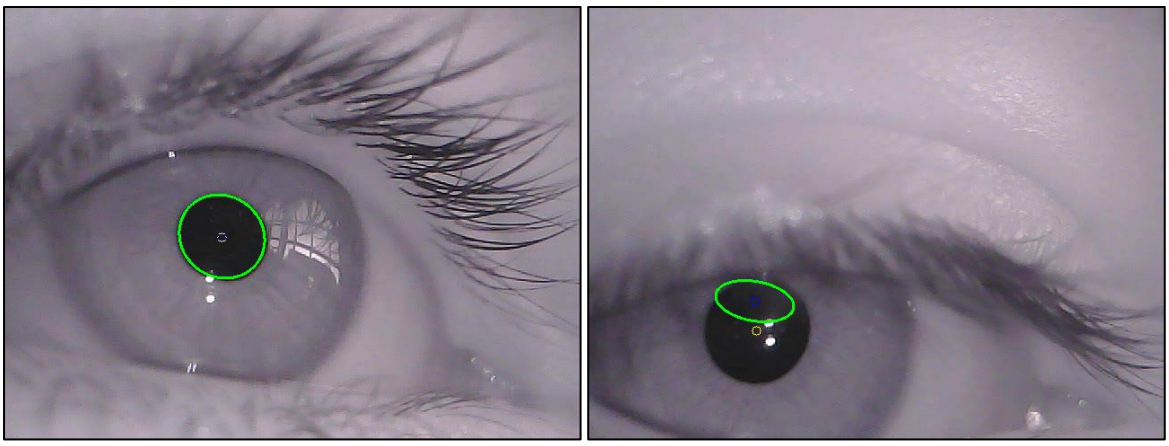


*Figure 45. Comparison between a detection with good accuracy (left) and one without (right). It is easy to see that distance is bigger between pupil centres when the accuracy is not good.*

Regarding the classification of the detections it is clear that one detection can be only correct or incorrect, it is thus, a binary classification. To classify them a threshold has been defined. In this way if the distance between detected pupil centre and ground truth is bigger than the threshold the result will be considered as incorrect and, if it is lower, as correct. This method adds a little bit of tolerance as it is really difficult that both pints are coincident but when that does not happen does not have to mean necessarily that the pupil was not detected goodly enough, maybe it is just not exact but still good enough for the purpose.

The previous paragraph can be summarised in the following condition.

$$correct\ detection = euclidean\ distance \leq threshold$$

In this stage it is obvious that the threshold makes a big difference as it is too big all the detections can be marked as correct and while if it is too small all may be incorrect. To solve this problem the solution is simple, each eye image is evaluated using several thresholds within a range equally spaced between them.

To finish with the explanation the image below is used to give a more graphical view of the threshold concept. Imagine that around the ground truth, coloured in yellow, several circles are drawn. Each of these circles has a known radius which is the threshold. When the circle drawn around the ground truth also contains the detection it is said that the detection is correct, otherwise it is incorrect. In the following illustration only the third circle (blue) contains both, the ground truth and the detection meanwhile the other two (red) do not. Thus, the selected threshold has such a big influence in the evaluation of the algorithm's performance and has to be selected carefully and in a reasoned way.

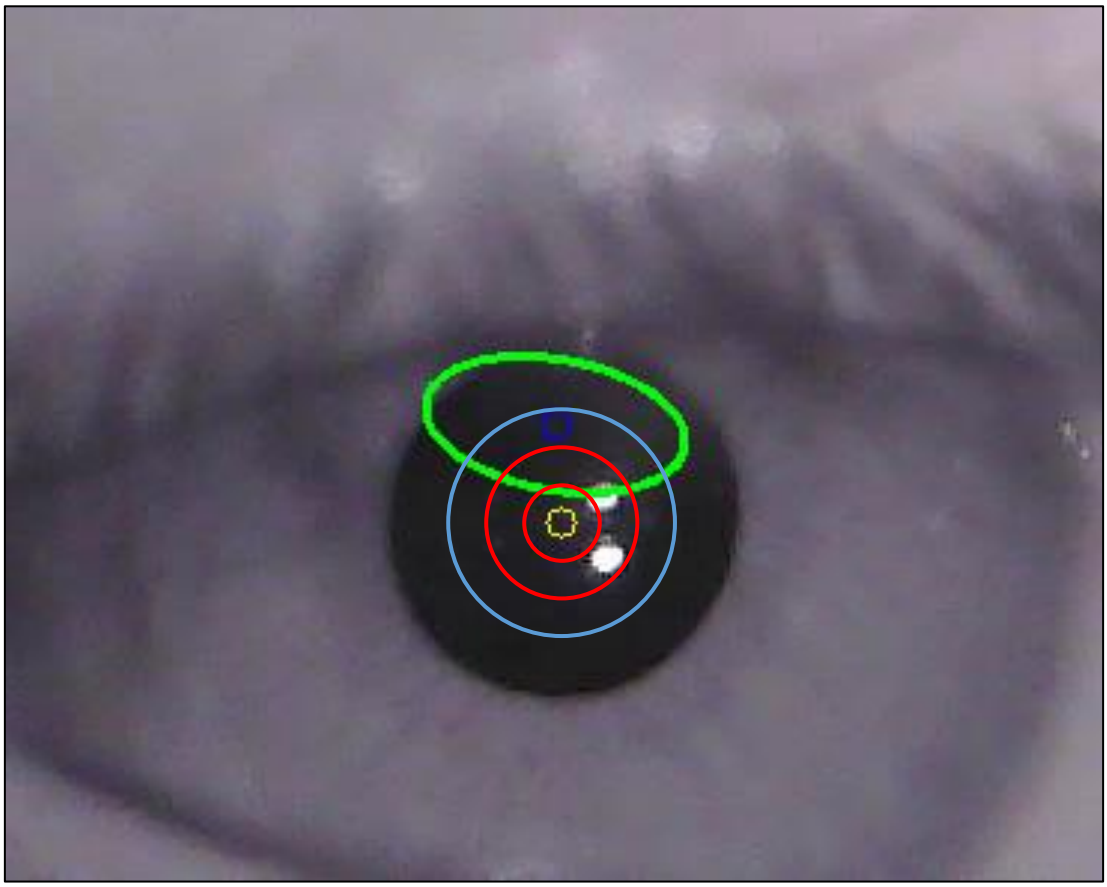


*Figure 46. Graphical representation of the threshold concept using circles whose radius matches with the selected threshold. In red thresholds that result in an incorrect detection, in blue thresholds that result in a correct detection.*

3.3.1.2. Detection rate

After calculating if a detection is correct or incorrect depending on the threshold selected it is needed to go further and find a metric that quantifies how good is the algorithm when working with a set of images as could be one of the videos from LPW dataset.

The set of images can be completely selected randomly, have nothing in common between them or be the frames from a video but it is needed a numerical value that tells us if the algorithm is valid to detect pupils on it or not.

This numerical value is a metric called detection rate. The detection rate is a really simple mathematical calculation that indicates the number of correct detections over the total amount of detections performed. As a result, its value will always be comprised between 0 and 1 and the bigger the better. Its mathematical formula is the following one.

$$detection\ rate = \frac{correct\ detections}{total\ detections}$$

It can also be represented as a percentage by multiplying it by 100.

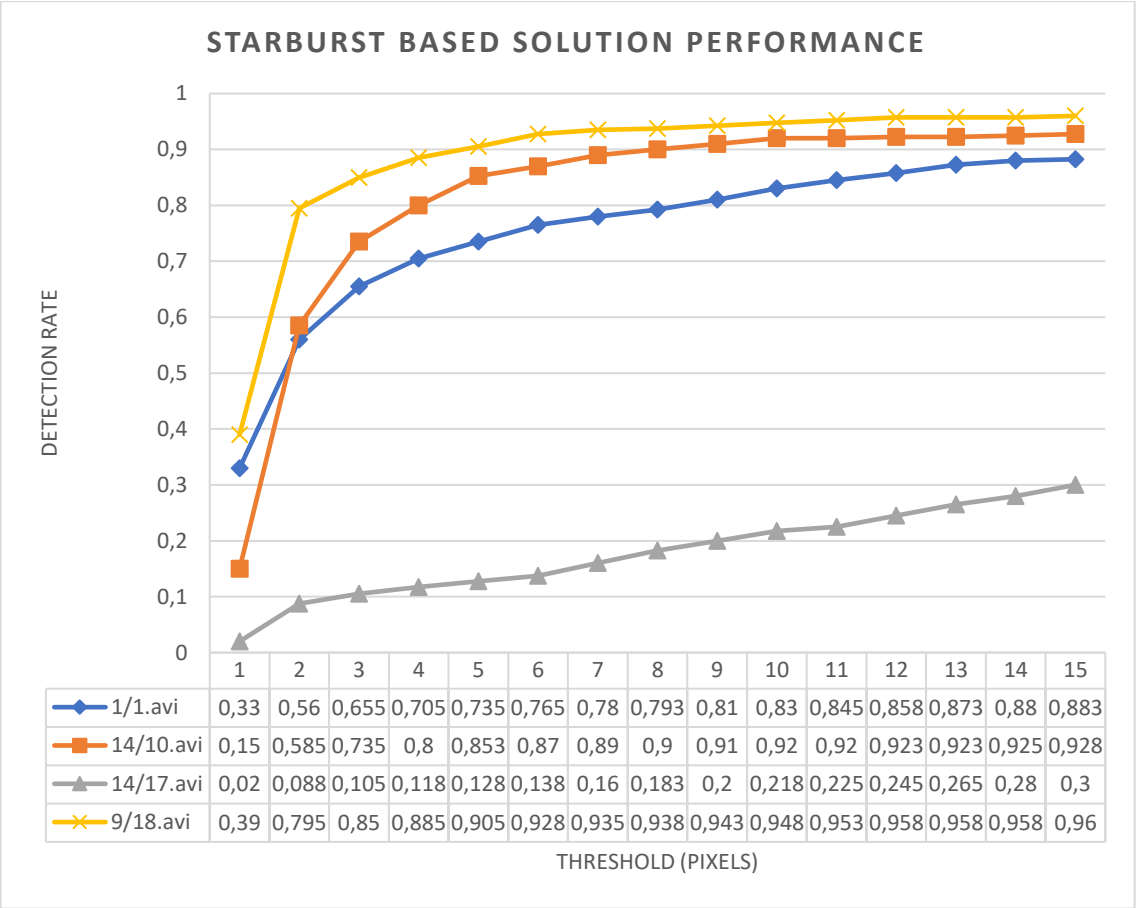
$$detection\ rate\ (\%) = \frac{correct\ detections}{total\ detections} * 100$$

This metric has been used, as in the case of the threshold, to evaluate the performance of pupil detection algorithms in many other works such as ElSe, ExCuSe, PuRe or PuReSt that have been cited in previous sections. This fact gives solidity to the works and allows to compare the implemented solution with these other algorithms which will be done in the following chapters.

### 3.3.2. Evaluation of the algorithm over an eye image dataset

In this section, as a continuation of the above, the Euclidean distance metric will be used to evaluate the performance of the algorithm with different thresholds in different videos extracted from the LPW dataset.

In the following page, a graphic and a table can be seen. They display the result of running the Starburst algorithm over four randomly selected videos contained in the LPW dataset. The vertical axis represents the detection rate, this is to say, the amount of correct detections over the total number of detections performed. The horizontal axis represents the threshold used to determine if a detection is correct or incorrect.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/1.avi | 0,33 | 0,56 | 0,655 | 0,705 | 0,735 | 0,765 | 0,78 | 0,793 | 0,81 | 0,83 | 0,845 | 0,858 | 0,873 | 0,88 | 0,883 |
| 14/10.avi | 0,15 | 0,585 | 0,735 | 0,8 | 0,853 | 0,87 | 0,89 | 0,9 | 0,91 | 0,92 | 0,92 | 0,923 | 0,923 | 0,925 | 0,928 |
| 14/17.avi | 0,02 | 0,088 | 0,105 | 0,118 | 0,128 | 0,138 | 0,16 | 0,183 | 0,2 | 0,218 | 0,225 | 0,245 | 0,265 | 0,28 | 0,3 |
| 9/18.avi | 0,39 | 0,795 | 0,85 | 0,885 | 0,905 | 0,928 | 0,935 | 0,938 | 0,943 | 0,948 | 0,953 | 0,958 | 0,958 | 0,958 | 0,96 |

*Figure 47. Starburst based solution detection rate when running it over four randomly selected videos from LPW dataset with different thresholds.*

The main purpose of the previous graphic is to show how the detection rate changes depending on the threshold used. As can be seen the first one increases rapidly with small changes of the threshold but as the threshold increases it tends to stabilize in one value.

Apart from the previous representation taking into account only four videos to make it easier to display the data, the detection method based on the Starburst algorithm has been executed over all the 66 videos of the LPW dataset. With the results a detection rate average has been calculated and it is displayed in the following graphic.

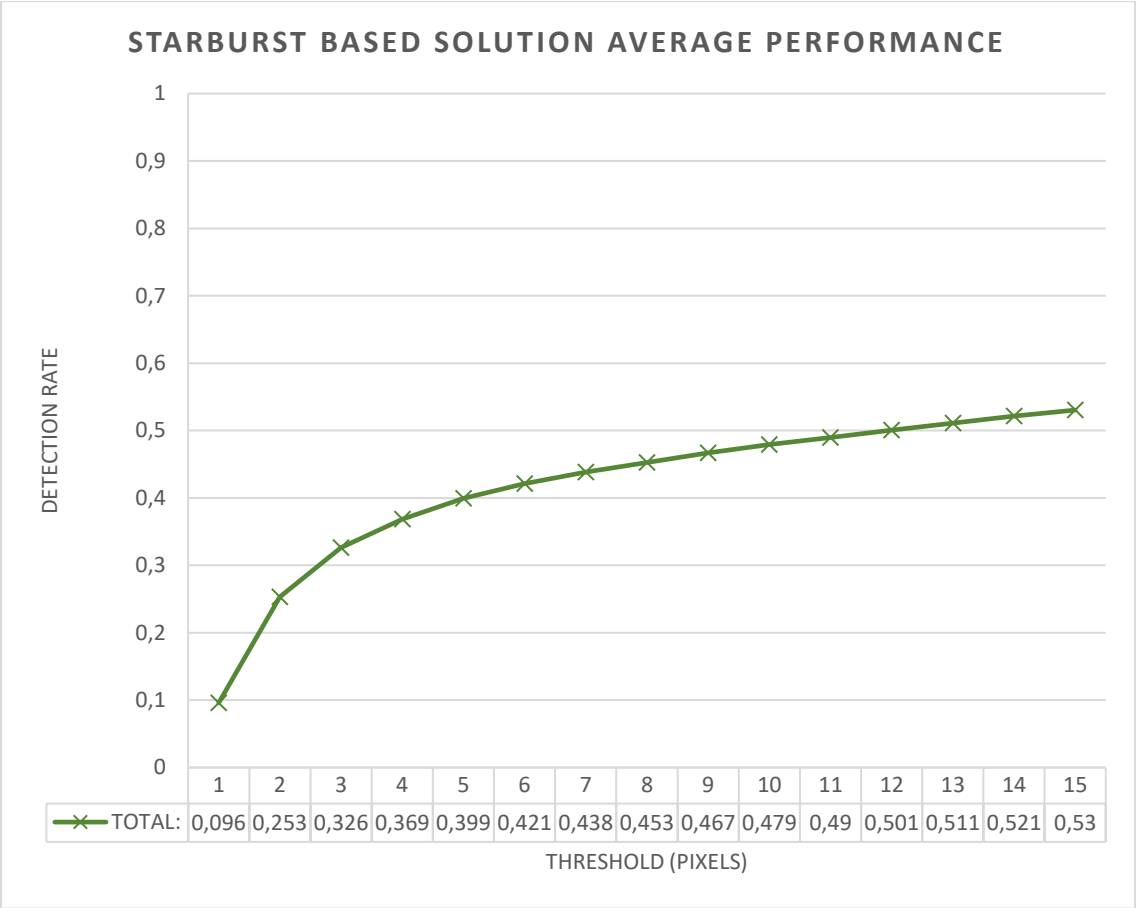



*Figure 48. Starburst based solution average detection rate when running it over all the videos from LPW dataset with different thresholds.*

Although the detection rate in the previous graphic with only four videos could seem good enough for three of them it is clear that, in average, the results are not so good. This is, as said before, largely due to the influence of the algorithm parametrization and the difficulty to provide a general solution with the designed method unless the environmental conditions are controlled.

It is also worth it to remember that the detection rate experiments big changes depending on the set of images as has been seen with the four-video example. Hence, as more sets are averaged the more reliable is the value obtained. The full table containing the results for each one of the 66 videos can be found in the annexes as it has been considered as too big to be displayed in this chapter.

## 3.4. Strengths and weaknesses of the Starburst algorithm

After the design, implementation and testing processes exposed along this chapter, it is already possible to make an assessment of the results achieved and determine the strengths and weaknesses of the developed solution based on the Starburst algorithm.

On the one hand, the selected solution showed that is able to detect pupils which is already a favourable point considering that it has been started from scratch. Additionally, it has been seen that it is able to provide a really good accuracy in some of the images, although not in all of them, this suggests that with a better parametrization strategy could provide really promising results so it still offers a margin of improvement. Another good aspect to take into account is that the Starburst algorithm can be considered as an elegant solution since its functioning is clearly understandable and is easily explained through really colourful and visual images. This logical structure and transparent functioning provoke that it is an easy to test algorithm as every line of code can be debugged, which is not the case in a CNN. This characteristic makes it suitable as a training for someone that is starting in the artificial intelligence and machine learning world.

On the other hand, the Starburst algorithm presents an accuracy that cannot considered really bad but that is clearly improvable, perhaps using a better parametrization as said in the previous paragraph. However, this could not be achieved during in this work and, as a consequence, the algorithm is not constant in its performance providing, sometimes, awful results that dim the accuracy average. Furthermore, it has to be said that this approach is still valid for many purposes and provides a solid performance in many fields but, regarding pupil detection, it can be said that is an outdated solution. If a look in the state of the art is taken it will be seen that many other solutions as convolutional neural networks, which will be seen in the following chapter, or techniques based on image processing provide a far better performance with less computational cost. In the object detection world where many times the problems require a real-time solution, the fact of not being efficient, unfortunately, can be a reason enough for an algorithm to be discarded, and this is the case. The times of processing for an eye image are too big to be a valid algorithm for real-time, which was not a specifically required target in this part of the project, but it is highly desirable for the near future.

All the conclusions that emerge from this section, have led to decide to test a solution based on a convolutional neural network and compare it to the Starburst based one. Hence, in the following chapter a solution based on a CNN is proposed and explained in detail.

# CHAPTER 4.
# Solution based on a convolutional neural network: Inception v4 CNN

After trying a traditional algorithm and find that its performance is not good enough, a new solution, based on convolutional neural network, has been developed. In this chapter, an explanation about how the CNN based solution works, how it has been trained and its performance results as well as an assessment of its pros and cons can be found.

## 4.1. How the CNN based solution works

In the first section how the selected CNN model, Inception-V4, has been built and what are the functions of each one of its parts will be analysed. Despite that designing and building the model is not inside the scope of this project, it has been considered as a requirement to collect all the information about it here so the reader can understand how it is structured and how it works.

### 4.1.1. Why Inception-V4

As already suggested, the selected convolutional network for this project has been Inception-V4. The following graph gives the key to the reason for this decision: its performance and size.

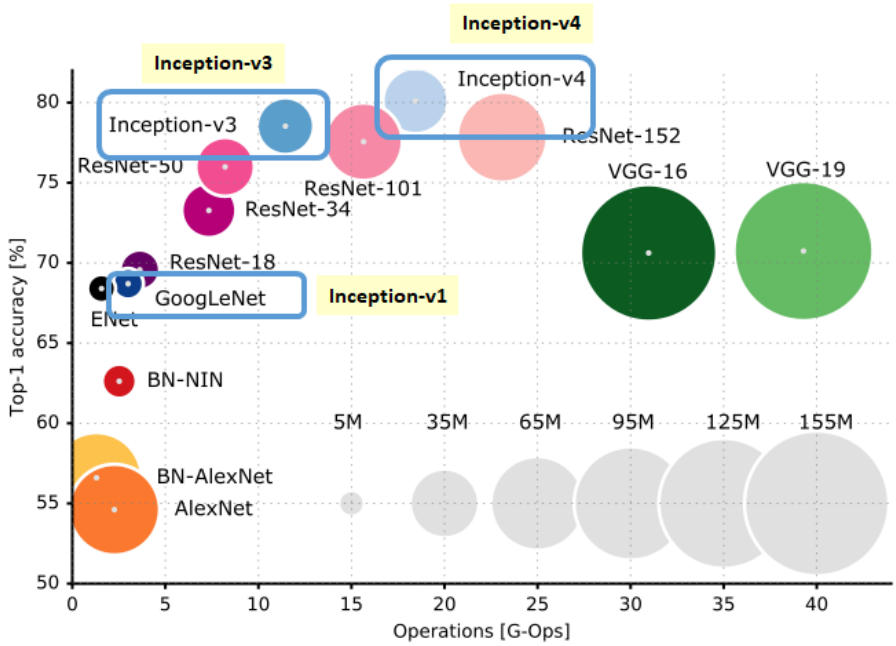



*Figure 49. Graphic that comparing the accuracy (%) of several CNN models to its size (G-Ops). (Source: https://towardsdatascience.com/)*

As can be seen, Inception-V4 boasts the best accuracy among the other neural networks shown in the graph. Consequently, it was the logical choice as the purpose of the project is detecting pupils with the highest possible accuracy. Although of its promising results shown in the graphic, it has to be said that its accuracy will depend largely in the quality of the data and the training process. However, starting with a good base, that is, with a good model, seems to be the best way to not compromise the CNN performance right from the start.

Another important reason for taking the decision of selecting Inception-V4 was the coding advantages as an implementation of it in TensorFlow was available on the internet. This, made a big difference on the time spent coding as many times, although the programmer knows the framework, small setbacks appear and cause important delays. Starting from the already implemented model only few adjustments and minor changes had to be done in order to adapt it for pupil detection.

Finally, one extra point to use Inception-V4 was the enormous amount of information about it on the internet as the developers from Google have been working on the Inception family for several years and different versions have been released.

### 4.1.2. Global view of the Inception-V4 neural network model

To work with a model, it is important to understand its structure and the function of the blocks that form it. In this way, during the training the programmer can better understand what is happening in the layers and try to improve the results obtained in a well-founded way. In this case, the Inception-V4 convolutional neural network presents the structure that can be found in the illustration below.

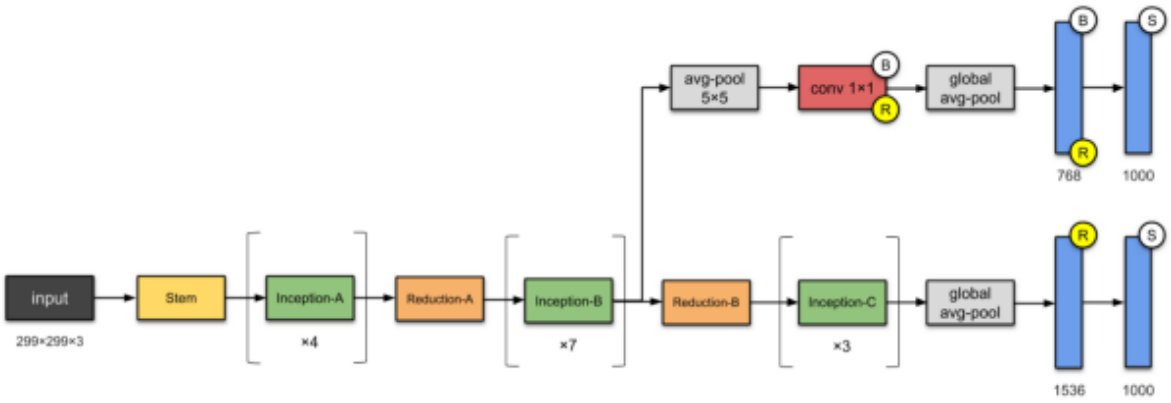



*Figure 50. Schematic representation of Inception-V4 convolutional neural network model developed by Google showing its main blocks and branches. (Source: https://towardsdatascience.com/)*

As can be observed in the previous schematic Inception-V4 contains several blocks including Stem block, Inception blocks, Reduction blocks, dense layers and average pooling block. Each one of these main parts will be detailed in the following points together with graphical representations.

The illustration below displays the legend that will be used in this chapter to represent the blocks and layers forming the CNN structure.

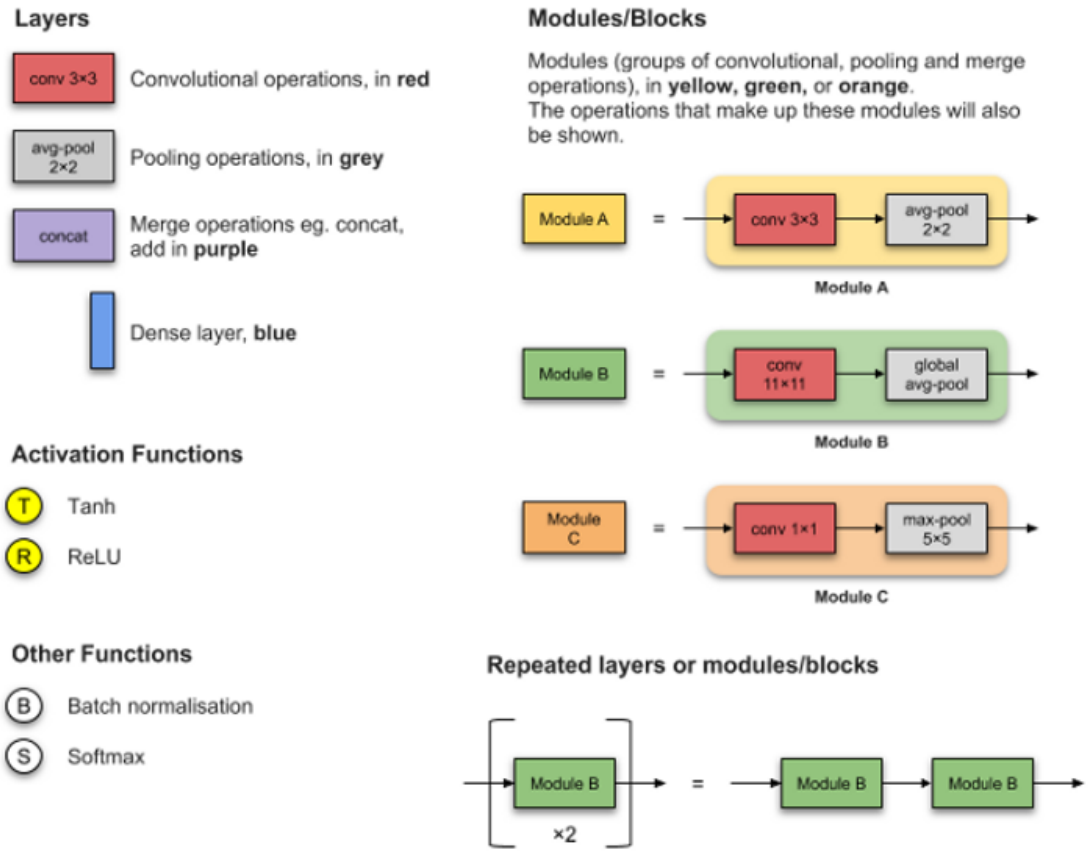



*Figure 51. Blocks legend used to represent the Inception-V4 CNN structure. (Source: https://towardsdatascience.com/)*

4.1.2.1. The neural network's input

Regarding the input of the network, despite that the original Inception-V4 expects a 299x299 three channels image, the model has been modified to accept 192x192 one channel images. This has been done because the available datasets contained only greyscale pictures.

As can be expected, using different datasets for training mean different image qualities and sizes. Because of that, a Python function has been created in order to homogenise all the pictures dimensions.

#### 4.1.2.2. The Stem block

The first block the input image passes through is the Stem block. This block contains several convolutional layers with different shapes. Some of these layers are connected in series while others are connected in parallel with concatenation layers. In the following schematic the stem block structure is represented.

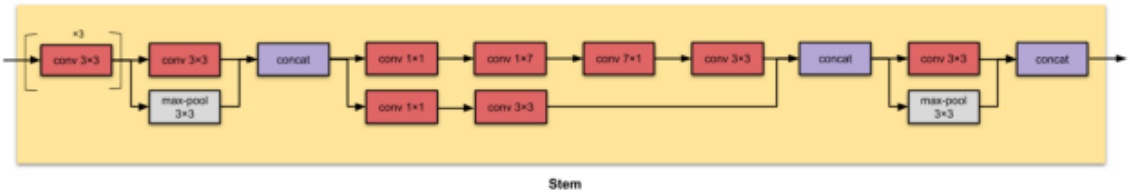



*Figure 52. Schematic of the Stem block from Inception-V4 CNN model with its blocks and branches. (Source: https://towardsdatascience.com/)*

#### 4.1.2.3. The Inception block

Inception blocks are used in CNNs to make the network more computationally efficient through a dimensionality reduction with stacked 1×1 convolutions. These modules, apart from solving the problem of computational expense, they help in preventing overfitting and other issues. The idea, in short, is to take multiple kernel filter sizes, and rather than stacking them sequentially, ordering them to operate on the same level.

The most simplified version of an inception module works by performing a convolution on an input image with three different sizes of filters (1x1, 3x3, 5x5). Also, max pooling is performed. Then, the resulting outputs are concatenated and sent to the next layer. By structuring the CNN to perform convolutions on the same level, it gets progressively wider instead of deeper.

To make the process even less computationally expensive, the convolutional neural network can be designed to add an extra 1x1 convolution before the 3x3 and 5x5 layers. By doing so, the number of input channels is limited and 1x1 convolutions are far cheaper than 5x5 convolutions. It is important to note, however, that the 1x1 convolution is added after the max-pooling layer, rather than before.

The following illustrations show the three types of inception blocks, named as Inception-A, Inception-B and Inception-C, used in the Inception-V4 CNN model. In this case, also asymmetric convolutions have been used.

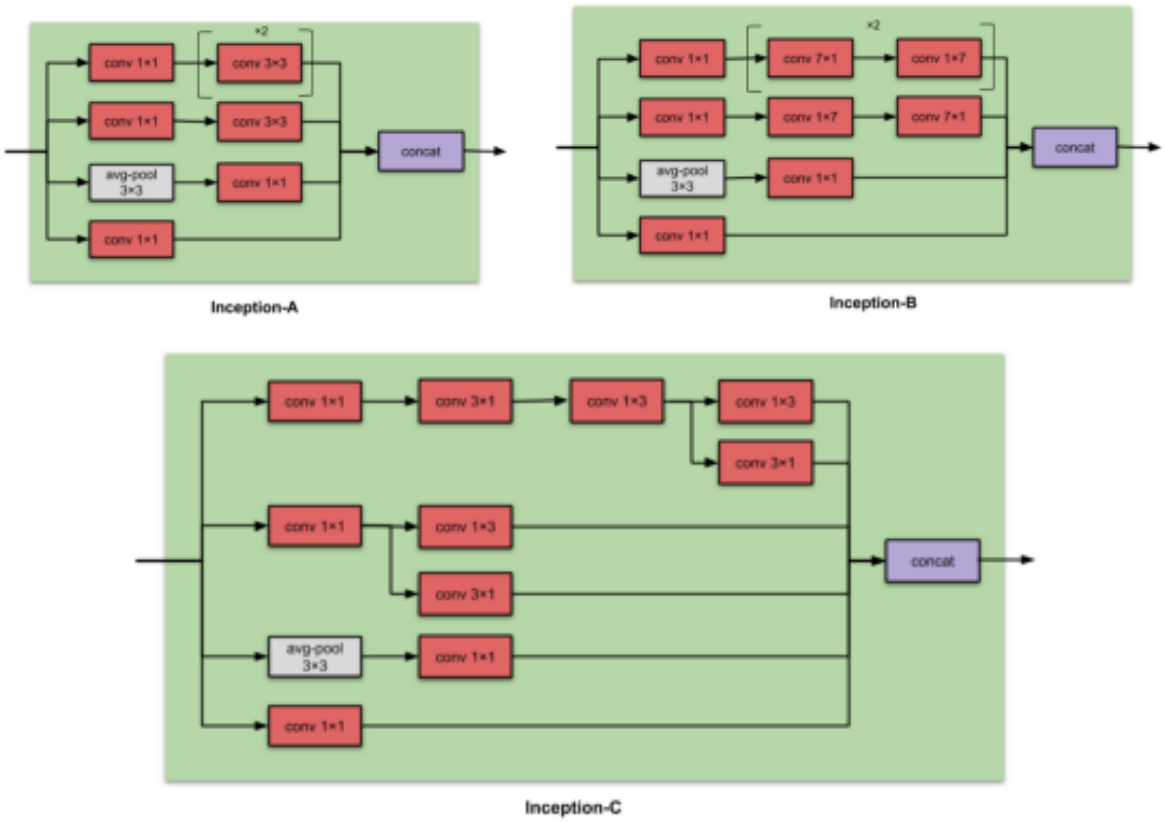



*Figure 53. Schematic representations of the Inception A, B and C blocks from Inception-V4 CNN model showing their blocks and branches. (Source: https://towardsdatascience.com/)*

#### 4.1.2.4. The Reduction block

After a group of inception blocks a reduction block is placed inside the Inception-V4 CNN. The reduction block has a similar structure to an inception block but with the purpose of reducing the number of parameters of the network.

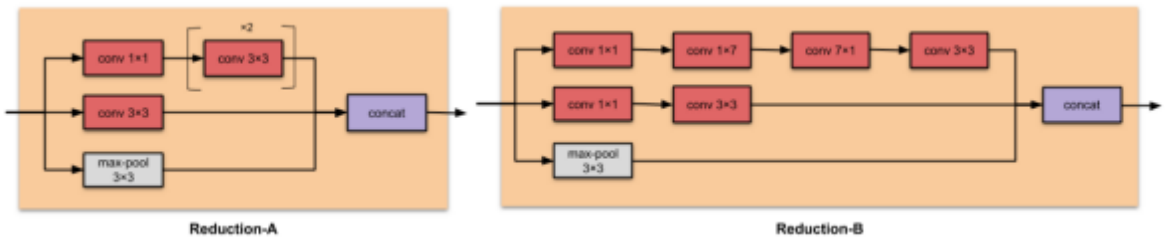



*Figure 54. Schematic representations of the Reduction A and B blocks from Inception-V4 CNN model showing their blocks and branches. (Source: https://towardsdatascience.com/)*

#### 4.1.2.5. The neural network's output

The neural network output is what reflects its performance on achieving the intended purpose, in this case, the detection of the pupil centre in an eye image. Thus, the implemented CNN, based on Inception-V4, has to provide two values, the X and Y coordinates of the pupil centre. To do so, the mean of all the elements in the tensor is computed and the network reshaped. Finally, the last convolutional layer is the one that provides two coordinate values.

The output of the network will be evaluated in the following sections in order to determine how accurate is it and its the average performance among the different videos of the LPW dataset. Below an image from the LPW dataset is attached. In the middle of the pupil, coloured in green, there is the detected pupil centre.

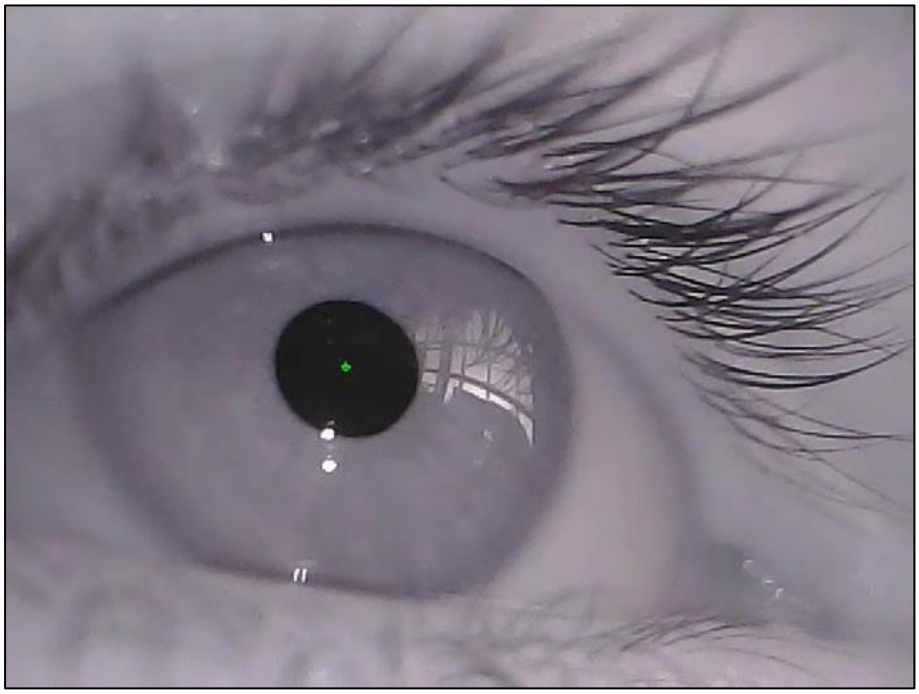


*Figure 55. Pupil centre detection in one of the frames of the LPW dataset using the trained CNN solution based on the Inception-V4 model.*

## 4.2. Implementation of the CNN based solution

Now that the Inception-V4 structure is clear and the modifications applied in order to the optimize it for pupil detection have been explained, the technical implementation aspects will be exposed in detail.

### 4.2.1. Libraries used to implement the CNN

In the table below a list of the main libraries used to develop the pupil detection convolutional neural network is attached. The complete list of libraries can be found in the annexes together with its version numbers. As can be observed some of them were used also for the implementation of the Starburst based solution.

| LIBRARY | DESCRIPTION | VERSION |
|---|---|---|
| openCV | OpenCV is a library of programming functions mainly aimed at real-time computer vision. It allows the user to read and write images or videos and to apply to them some image processing techniques as well to display them among many other functionalities.<br>In this project it has been used mainly for input/output of multimedia files, to visualize the pictures and apply basic transformations. | 4.1.0 |
| numpy | NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.<br>This library has been used basically to work with arrays and simplify some mathematical operations that are already built-in on it. | 1.17.0 |
| tensorflow | TensorFlow is library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning and deep learning applications such as neural networks.<br>It has been used to create the CNN structure using tensors, a container which can house data in N dimensions. The library contains numerous resources already prepared for neural networks implementation such as initializers and activation functions. | 2.0.0 |

*Table 3. Table containing a list of libraries used to implement the pupil detection convolutional network based on Inception-V4 with a brief description of each one.*

### 4.2.2. Training of the Inception-V4 CNN model

As already pointed in previous sections, implementing a CNN based method for pupil detection requires a training process. To do so a dataset containing hundreds of thousands of labelled pictures has been used. In the following paragraphs the technical aspects of the process will be detailed in order to understand how the training has been done.

#### 4.2.2.1. Training and validation data

The dataset used was formed by exactly 169191 pictures, all of them containing a pupil. To create it the 66 LPW videos have been split into frames and joined to the EMMA dataset pictures.

A basic method from Machine Learning to train neural networks is to split the available dataset into two different sets, the training and validation sets. This is done to preserve some unseen image to validate the results obtained with the training. In this case, the 80% of the eye pictures have been reserved for training and the 20% left for validation. These percentages left a training dataset with exactly 135352 pictures while the validation one contained 33839 images, all of them assigned randomly.

The percentages employed are more or less the typical ones although, depending on the number of pictures available or the criterion of the developer, they can vary a little bit. The important thing is always to use more pictures for training than validation as, then, the CNN can see more cases and is likely to generalize better.

Below, a collage containing 20 pictures from the used dataset is displayed.

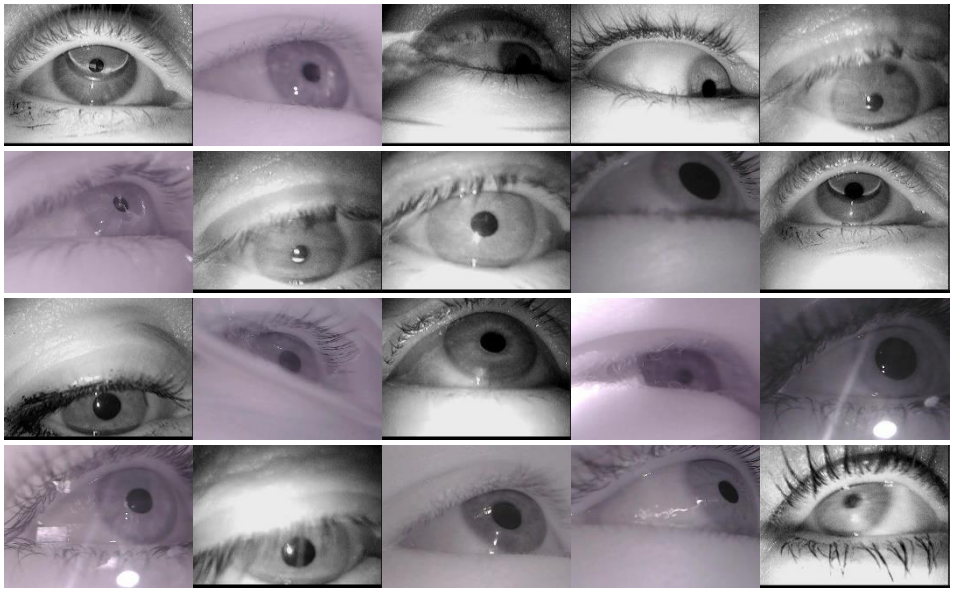


*Figure 56. Pictures containing a pupil from the training dataset.*

The purpose of the training set is clear, is the one used to update the weights between neurons in order to learn how to perform the detections required for the intended purpose, in this case, detect pupils. Nevertheless, the reason to keep a validation set may not be so obvious. One of the main reasons of its existence is the need to check, either at the end or during the training, that the CNN is not overfitting on the training set and, thus, is able to generalise well for unseen picture.

#### 4.2.2.2. Training process

Once the dataset is properly split into two sets, the training data has to fed to the convolutional neural network, this is done, by batches of a fixed size.

Many neural networks are trained using the batch gradient descent optimization algorithm. This involves using the current state of the model to make a prediction, comparing the prediction to the expected values, and using the difference as an estimate of the error gradient. This error gradient is then used to update the model weights and the process is repeated. The number of training examples used in the estimate of the error gradient is a hyperparameter for the learning algorithm called the batch size.

##### 4.2.2.2.1. The batch size and how it affects training

The batch size selected to train the CNN has been 64, this is to say, each time 64 images from the dataset are provided to the network. Numbers close to 64 samples are considered as small and usually work well. In some bibliography it is also possible to find recommendations to work with 32 samples. In general, these small batch sizes are used for two main reasons. The first one is that small batches are noisy, offering a regularizing effect and lower generalization error. And on the second place, they make it easier to fit one batch worth of training data in memory.

##### 4.2.2.2.2. Data augmentation as a mechanism to prevent training problems

Many times, in order to prevent overfitting and other problems that can occur during the training, data augmentation is used. This technique allows to slightly transform the images in a batch with different methods. These methods are applied randomly to each image assuring that the neural network difficultly will see the exact image twice.

In this case, the methods used involved rotating, rescaling, cropping o flipping the image. These common and simple transformations enable to create different images from the same one. Other methods used more specific for the pupil detection purpose have been adding fake pupils in the image, add blur, modify exposure or add partial occlusions so the detection is more challenging.

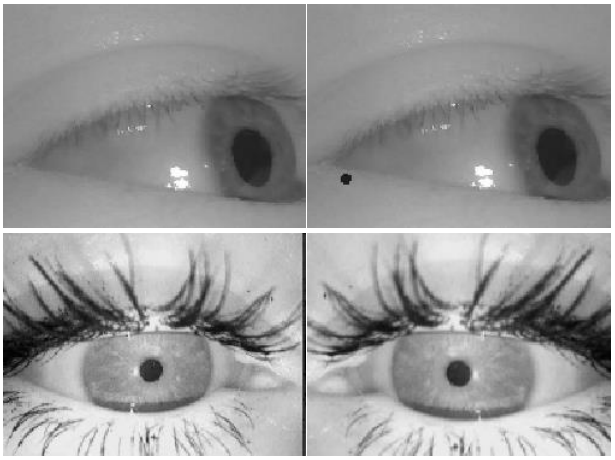


*Figure 57. Original images from the dataset used to train the CNN (left) together with the same image after applying to them data augmentation techniques (right).*

#### 4.2.2.2.3. Epochs in a training process

The training process is repeated several times, this is to say, once the whole dataset has been used the process starts again. This repetition is known as an epoch. In the current case the number of epochs has been 60, thus, the CNN has seen sixty times the same images organized in batches of 64 samples.

During an epoch the model is saved several times to prevent any progress loss in case that the training is stopped because of any problem. By doing that the model parameters can be reloaded from the saved file and start from the previous state. In the current CNN the model was saved every 3 batches.

#### 4.2.2.2.4. The learning rate effect on training

Apart from the structure in batches of the input data and calling the iterations over the dataset as epochs, there is another important parameter to take into account, the learning rate. This hyperparameter controls how much the model is changed in response to the estimated error each time the model weights are updated.

Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. For the reasons exposed the learning rate may is one of the most important parameters when configuring your neural network. The normal values are small positive numbers going in the range from 0.0 to 1.0.

In this case the selected learning rate is not the same along all the training process. The value starts being 0.001 and evolves as the training progresses and its minimum value ends up being 0.000001. This has been done to ensure that at the beginning the training is not too slow and, as the network gets closer to the optimal solution, the learning rate becomes smaller so the model is only being polished.

### 4.2.2.3. Validation process

As explained before, the training updates the weights of the CNN in order to optimize them for the intended purpose. Nevertheless, the training not always provides good results and some times too much training can lead to overfitting or other undesirable problems. A good practice to prevent that is to validate the model several times during the training.

This validation is done with the validation set which contains images previously unseen by the neural network. With the loss calculated with the validation it is possible to realise if the training goes in the right direction or not. Also, the validation loss is the value used to save the model. If the validation loss of the current batch is better than the stored one, the old is replaced with the new model file.

After completing the training, including the validations, a model is provided containing the optimized weights for pupil detection. However, this model does not have to be the one obtained in the las epoch but the model with the best loss, as said before.

## 4.3. Results obtained with the CNN based solution

After running the trained neural network, the centre coordinates are provided for each eye image containing a pupil. Unlike the Straburst based solution, this method already provides the coordinates instead of having to calculate them with the ellipse parameters. These coordinates are the ones that will be used by the other parts of this project, for this reason, evaluating their accuracy is fundamental. To do so, a mathematical criterion has to be defined.

In the following sections, the aspects that rule the evaluation criteria as well as its functioning will be explained in detail and graphical examples will be given.

### 4.3.1. Selection of the evaluation criteria

In this case, where the knowledge acquired with the development of the Starburst based solution is available, not much information research has been required. As in the previous chapter, Mean Average Precision (mAP) was discarded due to not meeting the problem to solve. Indeed, the evaluation to be done is practically the same as with the Starburst based method. For this reason, the same criterion has been chosen, the Euclidean distance.

#### 4.3.1.1. Euclidean distance method

The Euclidean distance method, as seen before, offers an extremely easy implementation, a practically inexistent computational cost and, the most important thing, a reliable way to calculate the accuracy. Its functioning is so simple, the accuracy is the distance in a straight line from the detected centre of the pupil to the ground truth or vice versa.

The formula used to calculate the metric, and for instance, the accuracy, is the one below.

$$Euclidean\ distance = \sqrt{\left(x_{det} - x_{gt}\right)^2 + \left(y_{det} - y_{gt}\right)^2}$$

$$where\ \left(x_{det}\ , y_{det}\right)\ is\ the\ detected\ pupil\ centre\ and\ \left(x_{gt}\ , x_{gt}\right)\ is\ the\ ground\ truth\ labelled\ in\ the\ dataset$$

In the subsequent pictures, extracted from the execution of the CNN on different videos of LPW dataset, can be seen how the detected pupil centre is much more accurate on the left picture than on the right one. In the same way, the distance between the detected centre of the pupil and the ground truth, represented by the blue and yellow points respectively, is greater on the right picture than on the left one. The pictures have a close similarity with the ones obtained with the Starburst implementation, however, the ellipse is missing as only the X and Y coordinates are provided.

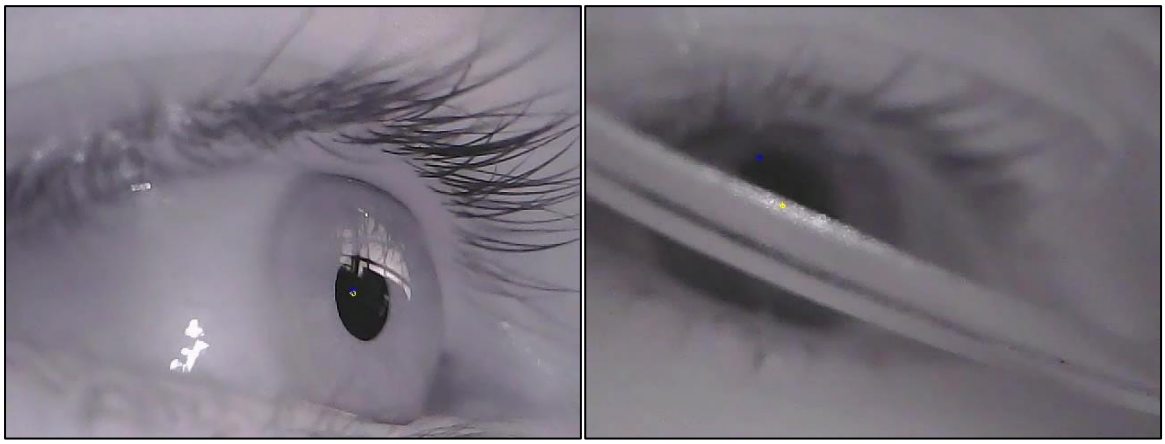


*Figure 58. Comparison between a detection with good accuracy (left) and one without (right). It is easy to see that distance is bigger between pupil centres when the accuracy is not good.*

Regarding the sorting of the detections, it is a binary classification as one detection can only be correct or incorrect. In order to do so a threshold has been defined. In this way if the distance between detected pupil centre and ground truth is bigger than the threshold the result will be considered as incorrect and, if it is lower, as correct. This method adds a little bit of tolerance as it is really difficult that both pints are coincident but when that does not happen does not have to mean necessarily that the pupil was not detected goodly enough, maybe it is just not exact but still good enough for the purpose.

As in the Starburst algorithm evaluation, the formula below represents when a detection is considered as correct.

$$correct\ detection = euclidean\ distance \leq threshold$$

Each eye image is evaluated using several thresholds within a range equally spaced between them to prevent the strong influence that has the threshold in the classification and provide a general view.

#### 4.3.1.2. Detection rate

After classifying the detection using the threshold selected, it is needed to go further and find a metric that quantifies how good is the trained CNN when working with a set of images as could be one of the videos from LPW dataset.

This numerical value is a metric called detection rate. As explained in the previous chapter, the detection rate is a really simple mathematical calculation that indicates the number of correct detections over the total amount of detections performed. As a result, its value will always be comprised between 0 and 1 and the bigger the better. The mathematical formula representing it is the following one.

$$detection\ rate = \frac{correct\ detections}{total\ detections}$$

It can also be represented as a percentage by multiplying it by 100.

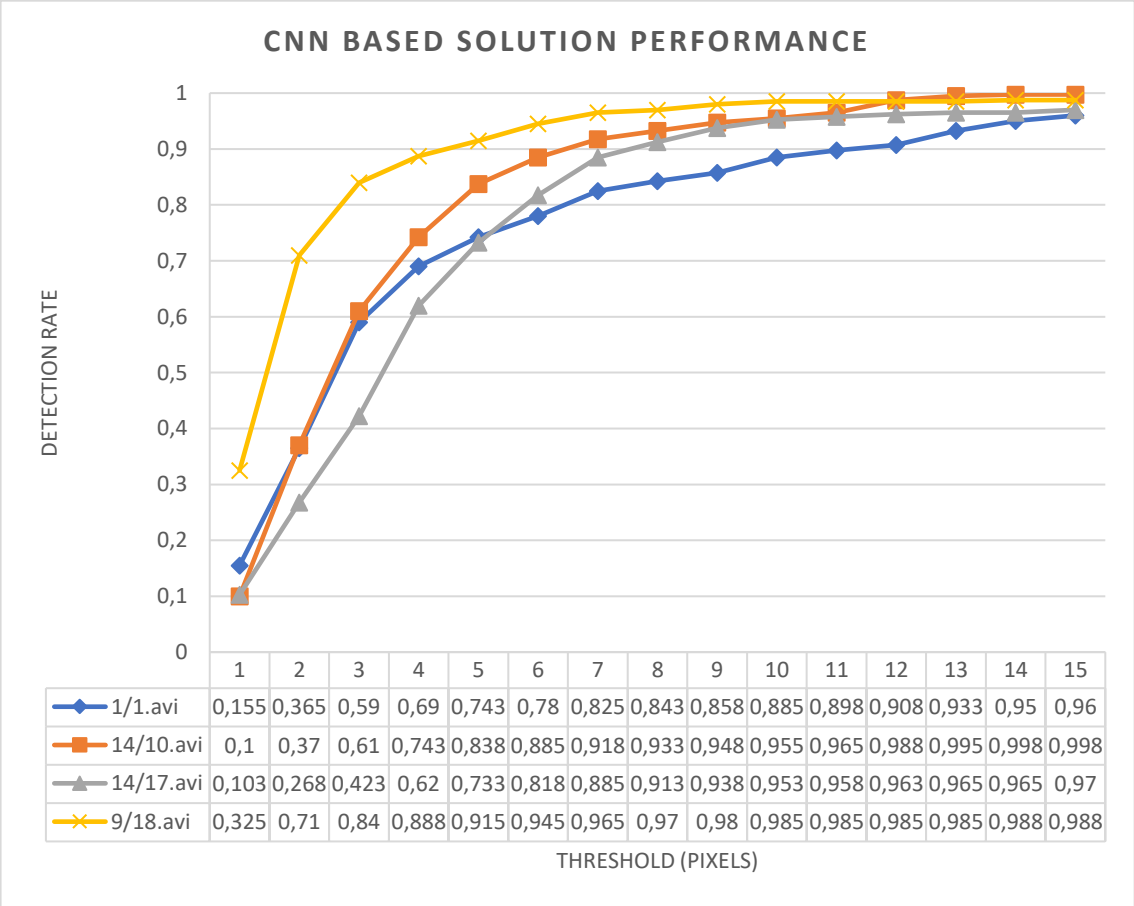
$$detection\ rate\ (\%) = \frac{correct\ detections}{total\ detections} * 100$$

Using the same metric greatly facilitates the performance comparison between the CNN based solution and the Starburst based one which has been an important reason to select it. A complete and detailed results comparison will be provided in the following chapter.

### 4.3.2. Evaluation of the CNN model over an eye image dataset

In this section, the Euclidean distance metric will be used to evaluate the performance of the trained CNN with different thresholds in different videos extracted from the LPW dataset.

In the following graphic displaying the result of running the CNN over four selected videos contained in the LPW dataset. The videos are the same ones used for the Starburst evaluation in order to ease the comparison. The vertical axis represents the detection rate, this is to say, the amount of correct detections over the total number of detections performed. The horizontal axis represents the threshold used to determine if a detection is correct or incorrect.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1/1.avi | 0,155 | 0,365 | 0,59 | 0,69 | 0,743 | 0,78 | 0,825 | 0,843 | 0,858 | 0,885 | 0,898 | 0,908 | 0,933 | 0,95 | 0,96 |
| 14/10.avi | 0,1 | 0,37 | 0,61 | 0,743 | 0,838 | 0,885 | 0,918 | 0,933 | 0,948 | 0,955 | 0,965 | 0,988 | 0,995 | 0,998 | 0,998 |
| 14/17.avi | 0,103 | 0,268 | 0,423 | 0,62 | 0,733 | 0,818 | 0,885 | 0,913 | 0,938 | 0,953 | 0,958 | 0,963 | 0,965 | 0,965 | 0,97 |
| 9/18.avi | 0,325 | 0,71 | 0,84 | 0,888 | 0,915 | 0,945 | 0,965 | 0,97 | 0,98 | 0,985 | 0,985 | 0,985 | 0,985 | 0,988 | 0,988 |

*Figure 59. CNN based solution detection rate when running it over four randomly selected videos from LPW dataset with different thresholds.*

The aim of the previous graphic is to display how the detection rate changes depending on the threshold used. As can be seen the first one increases rapidly with small changes of the threshold but as the threshold increases it tends to stabilize in one value. This can be understood as the incorrect detections are never added to the detection rate while the ones that are not exact but good enough finally are classified as correct by adding a little bit more of tolerance.

Apart from the previous representation that takes only into account four videos to prevent saturation of the graphic, the detection method based on the CNN trained model has been executed over all the 66 videos of the LPW dataset. With the obtained results, a detection rate average has been calculated and it is displayed in the following graphic.

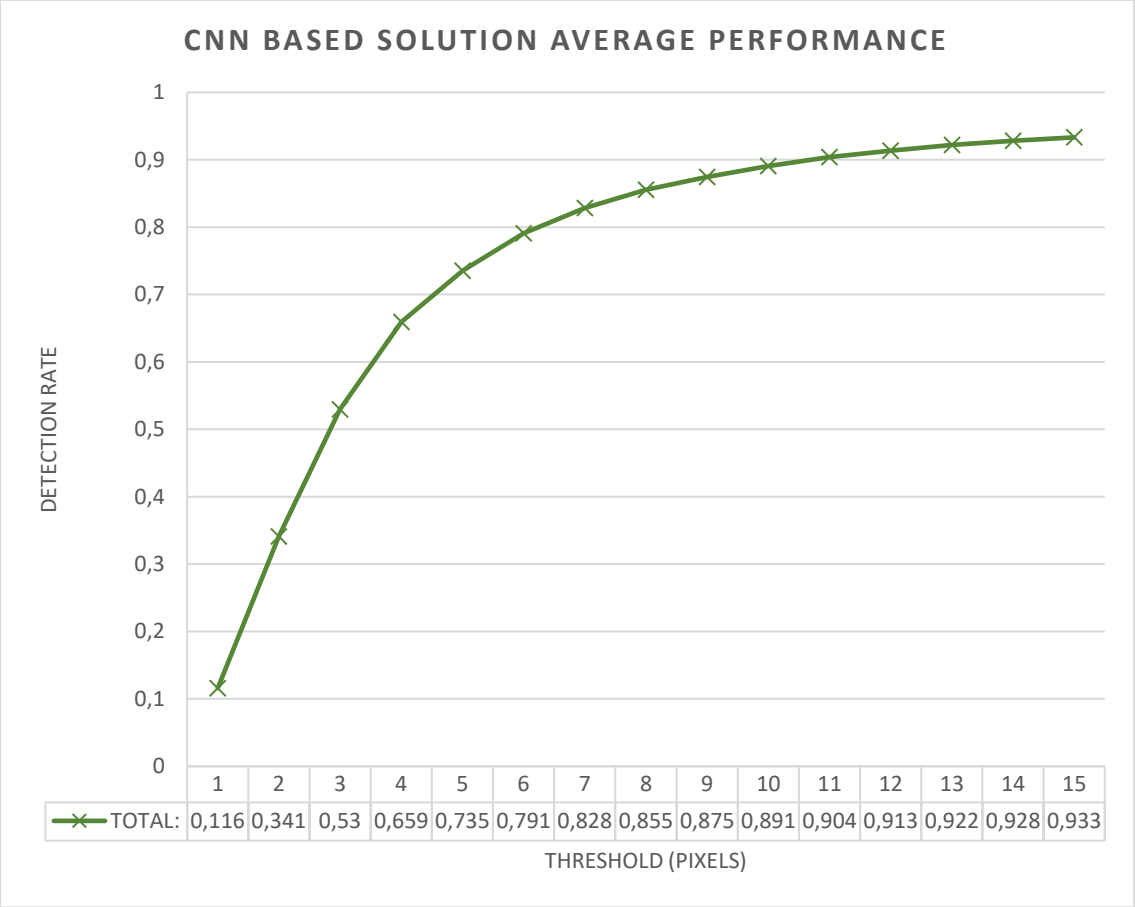



*Figure 60. CNN based solution average detection rate when running it over all the videos from LPW dataset with different thresholds.*

Despite the detection rate in the previous graphic with only four videos could seem nearly perfect when taking a look on the average graphic a slim decrease is observed. The fact that the decrease is so small illustrates the capacity of the trained neural network to generalise and provide good results indistinctly of the input.

As always, the detection rate is not exactly the same and it depends on the input. Hence, as more sets are averaged the more reliable is the value obtained. The full table containing the results for each one of the 66 videos can be found in the annexes as it contains to many data to be displayed in this section.

## 4.4. Strengths and weaknesses of the CNN based solution

After the model selection, its training and testing processes described on this chapter, it is now possible to analyse the obtained results and determine the strengths and weaknesses of the developed solution based on a convolutional neural network.

On the one hand, the solution implemented in this chapter showed its ability to not only detect pupils but identify them in the pictures with good detection rates. It has to be kept in mind that the solution was not started from scratch but it was created from an already existing neural network model what eased the implementation work. To this strengthens it has to be added the capability of the CNN to perform the detections quite fast which opens the door to its usage for real time applications. Additionally, the accuracy provided in the vast majority of frames suggests that the model works really fine and has still the possibility to be improved with a better training process. This could be done by adding better pictures to the training dataset, spent more time on it and analyse the training parameters that could be optimised. Finally, implementing a solution using a neural network makes the work a cutting-edge project as this type of neural networks are the ones starting to be used nowadays and they provide promising results.

On the other hand, a CNN based solution as the developed one presents serious problems in the training process. This is because the difficulties in finding datasets containing images labelled for the intended purpose. Finding eye image datasets with the pupil centre labelled with good quality became one of the biggest challenges of the project. It is really probable that for some other less common purposes there are not available datasets and they have to be created expressly with the cost in time and resources that this supposes. Additionally, the training process takes a lot of time and there is no guarantee that the result will be successful, indeed, many times it is not due to problems such as overfitting. Also, the difficulty to debug them does not help at all. Because of that, when there is a deadline approaching like in this project, training a CNN is a risky operation. Furthermore, the size of the generated files may compromise its application in small capacity devices.

All the conclusions that emerge from this section, suggest that the performance of the CNN solution is more desirable than the one from the Starburst algorithm. Hence, in the following chapter a detailed comparison between both solutions is performed.

# CHAPTER 5.
# Comparison between both solutions

Along this document two solutions for pupil detection that rely on two different technologies have been detailed. Now, with the information gathered with their implementation and testing processes, it is possible to determine which one is able to detect pupils more accurately spending the less possible amount of time, in short, decide based on the evidence which one is the most suitable for the intended purpose. Thus, this chapter contains a detailed comparison between both detection methods and provides the data to support it.

## 5.1. Comparison in terms of implementation

One of the first aspects to pay attention to is the implementation cost of each solution. This part affects more the programmer than the user but is important as the detection software has to be robust and, thus, easily implementable.

On the one side, the Starburst based method is easily programmable because it is a traditional algorithm and its debugging can be done without many problems. Then, it can be said that from the code point of view, once its functioning is clear, coding does not represent a big deal as only basic programming skills are required.

However, the difficulty comes when trying to parametrize it. The main hassle of this algorithm is the difficulty to generalize well as it relies on a saturation difference threshold between pixels and this is highly dependant upon the illumination. Several approaches have been tried in order to improve this aspect but the results have not been as good as expected. From the experience acquired this algorithm has to be parametrized for each situation and this means an important implementation setback.

On the other side, the CNN based solution does not represent an obstacle as coding the model using TensorFlow is easy but requires an additional knowledge of the framework. Nevertheless, as it has been done in this project, many models are already implemented on the internet and using them can be a solution for the coding problem as only few modifications have to be done to adapt them for the intended purpose.

Despite that, the training of the convolutional neural network represents a big challenge due to its complexity and risk of obtaining insufficient results after spending a lot of time or facing unexpected problems such as overfitting which are difficult to solve due to the poor debugging. To overcome the training problems extra knowledge is required on machine learning and specific techniques already defined to solve them as could be data augmentation.

To sum up, in this section the implementation costs of both solutions have been compared and from the experience obtained in this work it can be said that, in terms of implementation, the Starburst algorithm is the preferable one. If the code is parametrized for each situation, the rest of the problems that could appear are easily solvable and its knowledge requirements are far lower than the ones required to implement the CNN one.

## 5.2. Comparison in terms of computational cost

Another aspect to take into account when comparing both designed solutions is the computational cost of developing and using them. This is to say, how powerful has to be the equipment where the code will be executed. This comparison will be done only in terms of time as it is considered the important value if the code has to be executed in real time.

Regarding the Starburst algorithm, the computational cost of this option has revealed to be elevated during the execution. On average, it took 2,05s to process a single frame. This metric shows that, considering that a normal video has tens of frames per second, it clear that this method would not be suitable for real time applications as the time to process a frame is higher than the seconds per frame.

In comparison, the CNN based solution has revealed that its computational cost during the execution process is perfectly affordable. Specifically, the time spent per frame once the model is loaded was, on average, 0,14s. This good result makes this method based on a neural network suitable for real-time applications and means that it has a low computational cost respect to the previous one.

All the times referred in this section have been calculated using the OpenCV library. The process used has been as simple as obtaining the time before starting to process the LPW video and when finished. The average has been calculated by subtracting the starting time from the final time and dividing this number by the number of frames. The subsequent formula reflects the time calculation.

$$t = \frac{t_{final} - t_{start}}{number\ of\ frames}$$

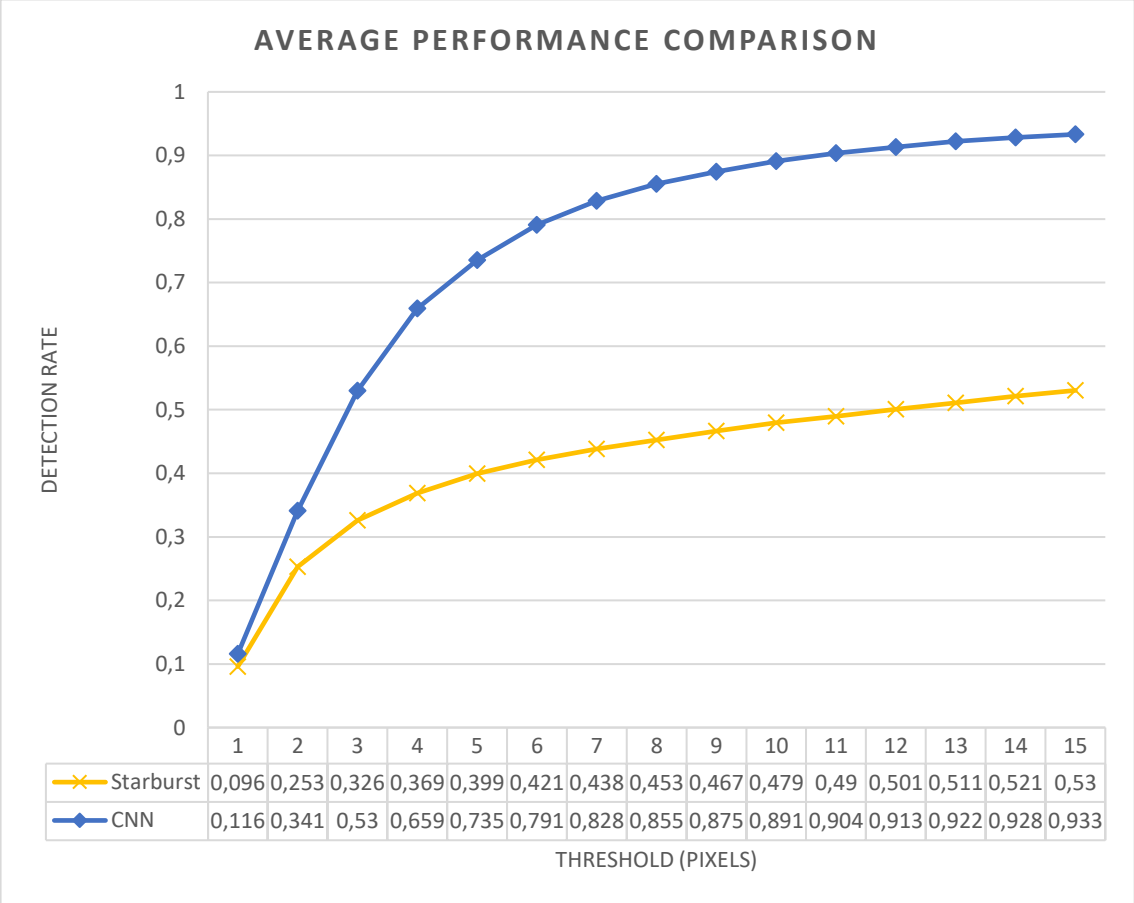
Also, it has to be taken into account that all the times reported are closely tied to the device that is running the code. This means that, in another machine, the time cost per frame will vary in direct relation to its power. Despite that, this measure is enough for the purpose of this section as it only intends to show which algorithm has a better performance over the LPW dataset. To sum up, it can be said that, computationally, the solution based on a CNN is approximately 11,5 times better than the Starburst based solution.

## 5.3. Comparison in terms of performance

After comparing the two solutions in terms of implementation and computational cost, it is the turn to do it in terms of performance. This comparison represents the detection accuracy achieved by each one. Consequently, as detecting a pupil with the best possible accuracy is the target of the project, this comparison is the most important. It also represents how well the objective of the work has been achieved and the remaining improvement margin.

### 5.3.1. Results with the LPW dataset

To make this comparison, both algorithms have been executed over the 66 of the LPW dataset. Doing it over the same videos assures that the results are comparable and provide a clear view of what accuracy is able to achieve each solution. The following graphic shows the average detection rate considering different thresholds for each one of the solutions.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Starburst | 0,096 | 0,253 | 0,326 | 0,369 | 0,399 | 0,421 | 0,438 | 0,453 | 0,467 | 0,479 | 0,49 | 0,501 | 0,511 | 0,521 | 0,53 |
| CNN | 0,116 | 0,341 | 0,53 | 0,659 | 0,735 | 0,791 | 0,828 | 0,855 | 0,875 | 0,891 | 0,904 | 0,913 | 0,922 | 0,928 | 0,933 |

*Figure 61. Comparison between the average detection rate of the Starburst based and CNN based solutions when executing them over the 66 videos from LPW dataset taking into account different thresholds.*

From the previous graphic one explicit conclusion can be extracted, the CNN based solution has a performance nearly two times better than the Starburst based one. In fact, the first one has a really good performance achieving, for example, detection rates of nearly 75% with a 5px threshold. On the contrary, the traditional algorithm is narrowly able to exceed a detection rate of 50% with a 15px threshold which is not enough to be considered a good result.

Below, another graphic is attached representing the detection rate for each of the 66 videos contained in the LPW dataset together with each solutions average.

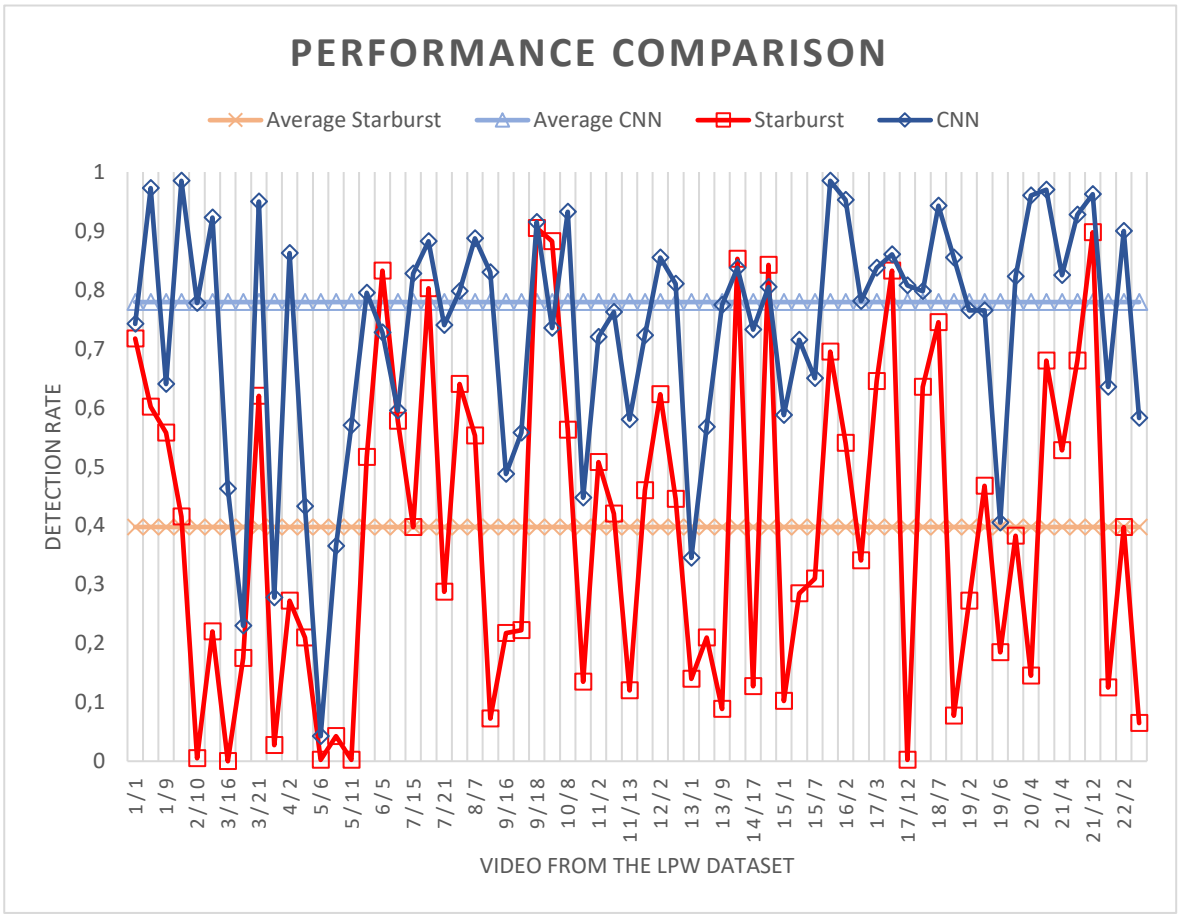



*Figure 62. Average detection rate for each one of the videos from LPW dataset when using a 5px threshold. The CNN solution results are represented in blue and the Starburst based ones in red. Also, the average for each solution is plotted.*

From the graphic several impressions can be extracted. First of all, the performance of both algorithms varies a lot depending on the input. And the second one, reinforcing the first graphic conclusions, it can be seen how, nevertheless both solution's detection rate fluctuate directly related to the input, the CNN has a better performance in the vast majority of the videos. Thus, this is reflected on its average being nearly two times better for a 5px threshold.

### 5.3.2. Results compared to other available solutions

To finish with this comparison chapter, both of the developed solutions will be compared with other existing pupil detection algorithms. This comparison will allow to zoom out from the project view and compare its results to other algorithms in order to understand whether the developed models are good or not.

Below a chart extracted from the paper that introduces PuRe algorithm is attached. On it, four algorithms, Swirski, ExCuSe, ElSe and PuRe are compared. All of them have been developed using traditional algorithms based on image processing techniques.

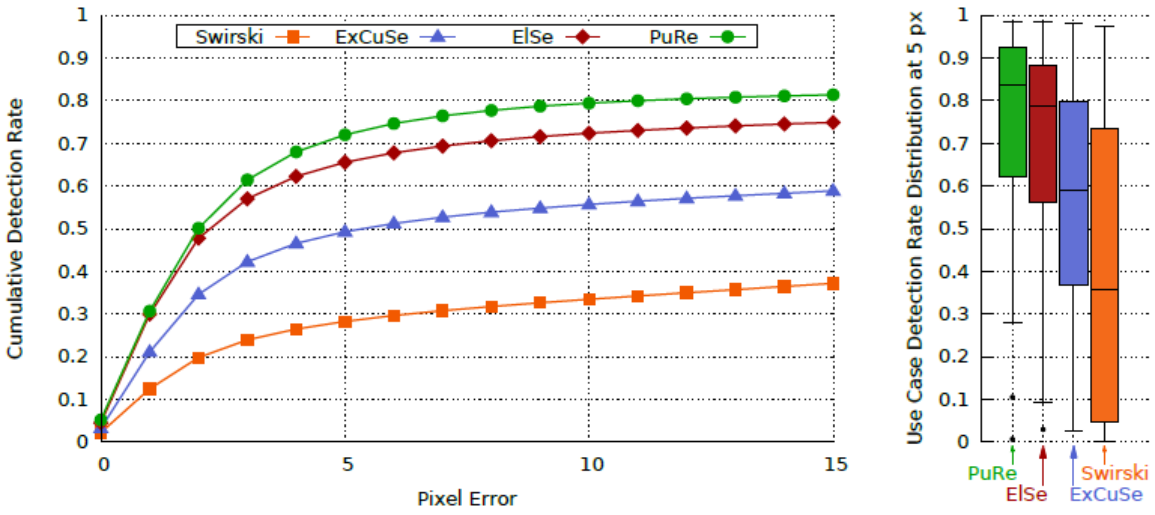



*Figure 63. Graphic showing the performance of Swirski, ExCuSe, ElSe and PuRe algorithms together with the distribution of the detection rate per use case considering a 5px threshold. (Source: Paper "PuRe: Robust pupil detection for real-time pervasive eye tracking")*

When comparing the detection rate curves in the previous graphic with the ones provided in the previous graphic showing the average detection rate for several thresholds, it can be seen that the solutions proposed in this project are not bad at all.

First of all, it can be observed how the Starburst based solution has not a good performance but it is still not the worst one. Additionally, when taking a look on the CNN based solution it happens to be the best one out of the six, by having a 10% more detection rate than PuRe. This is probably because the technology used, a convolutional neural network, is more powerful and appropriate for the purpose of detecting a pupil. Other solutions based on neural networks are also available and their average performance will be compared in the next page's table.

The following table compares four already existing pupil detection methods, PuRe, PuReSt, DeepEye and Eivazi's CNN approach, with the two developed in this project. The papers introducing them can be found in the bibliography at the end of this document. The comparison is performed for a 5px threshold and the detection rate is represented in percentage.

| Dataset | Detection Rate (%) considering 5px threshold | | | | | |
|---|---|---|---|---|---|---|
| | PuRe | PuReSt | DeepEye | Eivazi | Ours (CNN) | Ours (Starburst) |
| LPW | 75 | 82 | 50 | 84 | **78** | **40** |

*Table 4. Detection rate in percentage of different pupil detection solutions over the LPW dataset considering a 5px threshold. (Source: Paper "Improving Real-Time CNN-Based Pupil Detection Through Domain-Specific Data Augmentation")*

From the table can be seen that the developed solutions in this project do not have the best performance. PuReSt and Eivazi's CNN approach have a better detection rate and Starburst based method is the one with the worst performance what automatically discards it as a usable method as there are others already implemented with better results. Although the developed methods are not the best ones among the existing ones the developed CNN method is really close to the top ones which is clearly a triumph.

To sum up, after this comparison it is possible to affirm that the CNN approach has the best performance, much better than the Starburst one but a little less than other available pupil detection methods.

# CHAPTER 6.
# Analysis of the social and environmental impact

Centuries ago, Isaac Newton, the world-famous physician, stated in its third law each action produces a reaction. If this statement is generalized it can be applied to practically every aspect of life and this project is not an exception. The solutions proposed in this project, no matter how small are they or if they are used on a large scale or not, they end up producing an impact on the environment and society.

This chapter pretends to analyse and raise awareness about this impact as well as, despite of the difficulty, to quantify how big it is.

## 6.1. Environmental impact

When analysing the environmental impact of a software solution there are several aspects that have to be taken into account. In this section the main of them will be analysed and detailed.

### 6.1.1. Electrical consumption impact

The first one and perhaps the most important is the electricity consumption caused by the execution of the software. This point will be calculated in the following sections as it varies depending on which device is used to run the developed solutions, the Starburst and the CNN based one.

In order to calculate the spent energy per hour, the consumption of the laptop used for the implementation and testing of both solutions will be taken. To do so, the energy per hour is calculated taking into account that the battery has a capacity of 50Wh and lasts, approximately, 4 hours when executing on of both solutions.

The calculation of the energy consumed per hour is represented by the following formula.

$$E_{hour} = \frac{Battery\ capacity}{t} = \frac{50\ Wh}{4\ h} = 12{,}5\ W$$

The obtained results are not exact as the computer has a lot of internal processes not related with the pupil detection code. Nevertheless, due to the complexity of this calculation because of the big number of factors affecting it and the fact that this section does not seek to provide exact data but to raise awareness about the cost of executing an algorithm, the value of 12,5Wh is considered valid.

Then, if it is considered that the device that executes the pupil detection solution is running non-stop all year, the consumption would be:

$$E_{year} = E_{hour} * 24\ h * 365\ days = 109{,}5\ kWh$$

Which is a consumption to take into account as implies $CO_2$ emissions into the atmosphere in case the electricity is produced from non-renewable power plants based on fossil fuels. To calculate that, the following table that shows the cost in kg of $CO_2$ per kWh depending on the type of power plant used to produce it.

| ELECTICAL GENERATION TECHNOLOGY | EMISSION FACTOR AT THE POINT OF CONSUMPTION |
|---|---|
| Coal thermal power plant | 1,09 kg $CO_2$/kWh |
| Combined cycle power plant | 0,41 kg $CO_2$/kWh |
| Alternating internal combustion engine cogeneration plant | 0,45 kg $CO_2$/kWh |
| Gas turbine cogeneration plant | 0,42 kg $CO_2$/kWh |
| Steam turbine cogeneration plant | 0,48 kg $CO_2$/kWh |
| Combined cycle cogeneration plant | 0,42 kg $CO_2$/kWh |
| Urban solid waste incineration plant | 0,28 kg $CO_2$/kWh |
| Petroleum products plant | 0,80 kg $CO_2$/kWh |

*Table 5. Emission factors between kg of CO2 and kWh depending on the type of electricity generation technology which uses the central. (Source: Ministry of Industry, Energy and Tourism, Government of Spain)*

From the previous calculus and table, a new table has been generated using the following equation which shows the amount of $CO_2$ that would be emitted into the atmosphere if the execution of the software would be done using devices powered by energy produced in non-renewable power plants based on fossil fuels.

$$Emitted\ CO_2\ in\ kg = E_{year} \cdot Emission\ factor$$

| ELECTICAL GENERATION TECHNOLOGY | EMISSION FACTOR AT THE POINT OF CONSUMPTION |
|---|---|
| Coal thermal power plant | 119,36 kg $CO_2$ |
| Combined cycle power plant | 44,90 kg $CO_2$ |
| Alternating internal combustion engine cogeneration plant | 49,28 kg $CO_2$ |
| Gas turbine cogeneration plant | 45,99 kg $CO_2$ |
| Steam turbine cogeneration plant | 52,56 kg $CO_2$ |
| Combined cycle cogeneration plant | 45,99 kg $CO_2$ |
| Urban solid waste incineration plant | 30,66 kg $CO_2$ |
| Petroleum products plant | 87,60 kg $CO_2$ |

*Table 6. Amount of CO2 in kg emitted into the atmosphere if the device running the pupil detection solution is powered with electricity generated by non-renewable fossil fuel power plants.*

As can be seen in the previous table, an important amount of $CO_2$ can be emitted to the atmosphere if the electricity source is not consciously selected. For this reason, the importance of using green power sources can make the difference whether the use of this technology has a negative impact on the planet or not. Once more, as also will be reflected in the social impact section, how the technology is used is much more important than the technology itself when estimating its impact.

To finish with the electricity consumption impact section, emphasize once more the fact that all those calculations are approximated as there are many factors that can change the duration of a battery and are difficult to estimate such as the battery degradation along its life. It is considered not worth it to dedicate more effort to be more exact in the numbers than the necessary to raise awareness about the greenhouse emissions and that running a software implies also an impact to the environment.

### 6.1.2. Device manufacturing impact

The second aspect to take into account is the device manufacturing cost. In order to run the software, a device, maybe a laptop o maybe another type of device, is needed. As it happens with every manufacturing process it has an impact to the environment.

It is difficult to quantify which is this cost in a specific case, nevertheless there is information indicating that electronic devices and chips are one of the products with the highest manufacturing cost in terms of energy. This means that its ecological footprint, at the end of its life, presents a distribution where the cost of the manufacturing process for the planet outweighs the energy cost of its use. This means that replace them with a new one has a high impact that is difficult to compensate by the energy savings due to the efficiency of the new device.

That is why you have to be aware of this fact and realise that sometimes the problem is not in the energy consumption due to its use, as emphasized in the previous section, but the cost of manufacturing them. Assuring a proper recycling process at the end of its life can help in reducing its impact on the environment.

## 6.2. Social impact

From the social point of view an anti-spoofing mechanism for face recognition has probably few direct implications. It is not what the developed solution does but more what it is contributing to a more secure, robust and reliable face recognition system could contribute to extend the use of this technology among the society and the organizations.

A world where face recognition is used instead of the traditional ways of identification such as plastic cards or users and passwords would be a much more secure world where identity fraud is difficult to do. Also, the user avoids carrying cards or other types of recognition that can be lost or forgotten easily which is translated in a more comfortable way of identification as, if there is something that everyone always brings with himself is the face. Consequently, works focused in improving face recognition technology as this one contributes to a more comfortable and greener world, where use of paper and plastics is reduced and technology is employed as a way to enhance the wellness of the people.

Nevertheless, it is important to raise awareness about the need to use face recognition in an ethical way as, in the same way that happens with all the really powerful technologies, not doing so can lead to a deterioration of the freedoms and privacy of the people.

# Conclusions

Throughout the design and implementation process of the pupil detection solution, numerous challenges have arisen that have required various solutions, ending in two validated and operational solutions as a result of information research, modifications and improvements in order to overcome all obstacles. Both solutions provided present different detection rates and strengthens and, together, make up a picture of all the work done.

When the foundations of this project were laid, a series of limited and concise objectives were defined, such as developing a pupil detection algorithm that would be incorporated, in the near future, in an anti-spoofing mechanism. At the same time, this project has the aim of guaranteeing the proper functioning of the implemented solutions by testing them over a high-quality dataset. Now, looking back, it is possible to say that, with varying degrees of success and satisfaction, all the initial goals have been achieved.

Paying attention to the main and most important of the objectives set, it can be said that the design and implementation has been one of the most successfully covered points, as both solutions have proved to be able to detect the centre of a pupil in a picture. Moreover, the CNN solution has achieved really promising detection rates, nearly comparable to other already existing methods.

Analysing the first developed solution, the Starburst based one, its results have been not good enough regarding the detection rate. The results revealed by testing it over the LPW dataset have shown that, although its capacity to successfully detect pupils in some pictures, its average detection is not enough nowadays. The main reason for these results is that relies on a threshold that can be affected easily when the environment conditions change and, thus, it is difficult to parametrize the algorithm so it is able to generalize well. Despite of that, it must be said that this aspect is possibly the point at which work needs to be continued. The reasons exposed in this paragraph, together with the long time required to process a single picture, made the decision to implement a new method based on a different and more modern technology.

The second solution implemented and also the most successful one relies on a cutting-edge technology that revealed to be useful in this type of problems, the convolutional neural networks. Based on this technology the solution implemented presented a good performance during the tests with a detection rate and a processing time that validated it. A good detection rate implies a good accuracy which is the main objective when detecting pupils. These results enable to say that the main objective has been successfully achieved after making some modifications and facing several problems. It has to be said that the training process has not been easy and, due to the lack of time, many tests to improve the results could not be done. Improving the training with more and better labelled eye images or by tuning hyperparameters and a better debugging represent a field where much work could be done yet.

Regarding other points also mentioned in the objectives of the project, the present document aimed to display in detail all the process followed so the reader can use this project as a guide to implement his or her own solution for pupil detection or whatever is needed. This aim goes together with the objective of entering to the artificial intelligence and deep learning world which has been clearly been achieved with the amount of knowledge gathered.

As a last point to assess already, if one pays attention to the software developed, after the tests performed and confirming that the implemented part is able to do its work successfully, only one part remains missing, its assembly in the whole anti-spoofing detection mechanism. As stated in the beginning of this document, this work is part of a wider project that is still being built. Because of that, the implementation of this part into the whole mechanism will have to wait until everything is ready. This fact has left a small bittersweet sensation due to not being able to see the whole result.

Finally, to conclude with the conclusions and assessment of the work results, it is worthwhile to make a brief reflection of a more personal nature on what this project has provided. A very positive experience emerges from all the time spent on this work, both in terms of knowledge and staff, as I have acquired and improved knowledge in fields as diverse as computer science, image processing, deep learning, artificial intelligence or even in research and information management. Also make clear the desire to make this work a tool that helps and provides valuable information to its readers while leaving for the future the implementation of the improvements expressed throughout this text and that have been left out of the scope of the project in addition to others deemed appropriate.

# Bibliography

[1] Abbasi, M., Khosravi, M.R. ***"A Robust and Accurate Particle Filter-Based Pupil Detection Method for Big Datasets of Eye Video".*** J Grid Computing 18, 305–325 (2020). https://doi.org/10.1007/s10723-019-09502-1

[2] Chaudhary, A.K., Kothari, R., Acharya, M., Dangi, S., Nair, N., Bailey, R., Kanan, C., Diaz, G.J., & Pelz, J.B. (2019). ***"RITnet: Real-time Semantic Segmentation of the Eye for Gaze Tracking".*** 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 3698-3702.

[3] D. Wen, H. Han and A. K. Jain, **"Face Spoof Detection with Image Distortion Analysis"** in IEEE Transactions on Information Forensics and Security, vol. 10, no. 4, pp. 746-761, April 2015, https://doi.org/10.1109/TIFS.2015.2400395.

[4] Dongheng Li, D. Winfield and D. J. Parkhurst, **"Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches"** 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops, San Diego, CA, USA, 2005, pp. 79-79, https://doi.org/10.1109/CVPR.2005.531.

[5] Eivazi, Shahram & Santini, Thiago & Keshavarzi, Alireza & Kübler, Thomas & Mazzei, Andrea. (2019). "***Improving real-time CNN-based pupil detection through domain-specific data augmentation"***. 1-6. https://doi.org/10.1145/3314111.3319914.

[6] Fuhl W., Rosenstiel W., Kasneci E. (2019) ***"500,000 Images Closer to Eyelid and Pupil Segmentation".*** In: Vento M., Percannella G. (eds) Computer Analysis of Images and Patterns. CAIP 2019. Lecture Notes in Computer Science, vol 11678. Springer, Cham. https://doi.org/10.1007/978-3-030-29888-3_27

[7] Fuhl W., Kübler T., Sippel K., Rosenstiel W., Kasneci E. (2015) ***"ExCuSe: Robust Pupil Detection in Real-World Scenarios".*** In: Azzopardi G., Petkov N. (eds) Computer Analysis of Images and Patterns. CAIP 2015. Lecture Notes in Computer Science, vol 9256. Springer, Cham. https://doi.org/10.1007/978-3-319-23192-1_4

[8] Fuhl, Wolfgang & Santini, Thiago & Kübler, Thomas & Kasneci, Enkelejda. (2015). ***"ElSe: Ellipse Selection for Robust Pupil Detection in Real-World Environments".***

[9] Fuhl, W., Tonsen, M., Bulling, A. et al. ***"Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art. Machine Vision and Applications"*** 27, 1275–1288 (2016). https://doi.org/10.1007/s00138-016-0776-4

[10] Fuhl, W., Santini, T., Kasneci, G., & Kasneci, E. (2016). ***"PupilNet: Convolutional Neural Networks for Robust Pupil Detection".*** ArXiv, abs/1601.04902.

[11] Fuhl, W., Santini, T., Kasneci, G., Rosenstiel, W., & Kasneci, E. (2017). ***"PupilNet v2.0: Convolutional Neural Networks for CPU based real time Robust Pupil Detection".*** ArXiv, abs/1711.00112.

[12] Garbin, S.J., Shen, Y., Schuetz, I., Cavin, R., Hughes, G., & Talathi, S.S. (2019). ***"OpenEDS: Open Eye Dataset".*** ArXiv, abs/1905.03702.

[13] George, A., & Routray, A. (2018). ***"ESCaF: Pupil Centre Localization Algorithm with Candidate Filtering".*** ArXiv, abs/1807.10520.

[14] George, Anjith & Routray, Aurobinda. (2016). ***"A Fast and Accurate Algorithm for Eye Localization for Gaze Tracking in Low Resolution Images".*** IET Computer Vision. https://doi.org/10.10.1049/iet-cvi.2015.0316.

[15] Martinikorena, Ion & Cabeza, Rafael & Villanueva, Arantxa & Urtasun, Iñaki & Larumbe-Bergera, Andoni. (2018). ***"Fast and robust ellipse detection algorithm for head-mounted eye tracking systems. Machine Vision and Applications"***. https://doi.org/10.1007/s00138-018-0940-0.

[16] Park, S., Zhang, X., Bulling, A., & Hilliges, O. (2018). ***"Learning to find eye region landmarks for remote gaze estimation in unconstrained settings".*** Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications.

[17] Santini, T., Fuhl, W., & Kasneci, E. (2018). ***"PuRe: Robust pupil detection for real-time pervasive eye tracking".*** ArXiv, abs/1712.08900.

[18] Santini, T., Fuhl, W., & Kasneci, E. (2018). ***"PuReST: robust pupil tracking for real-time pervasive eye tracking".*** Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications.

[19] Szegedy, Christian & Ioffe, Sergey & Vanhoucke, Vincent & Alemi, Alexander. (2016). ***"Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning".*** AAAI Conference on Artificial Intelligence.

[20] Swirski, L., Bulling, A., & Dodgson, N.A. (2012). ***"Robust real-time pupil tracking in highly off-axis images".*** ETRA '12.

[21] W. J. Ryan, D. L. Woodard, A. T. Duchowski and S. T. Birchfield, **"Adapting Starburst for Elliptical Iris Segmentation"** 2008 IEEE Second International Conference on Biometrics: Theory, Applications and Systems, Arlington, VA, 2008, pp. 1-7, https://doi.org/10.1109/BTAS.2008.4699340.

[22] Y. Luo, X. Qin, N. Tang and G. Li, ***"DeepEye: Towards Automatic Data Visualization"*** 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, 2018, pp. 101-112, https://doi.org/10.1109/ICDE.2018.00019.

[23] Yiu, Y., Aboulatta, M., Raiser, T., Ophey, L., & Ahmadi, S. (2019). ***"DeepVOG: Open-source Pupil Segmentation and Gaze Estimation in Neuroscience using Deep Learning".*** Journal of neuroscience methods, 324, 108307.

[24] Y. Atoum, Y. Liu, A. Jourabloo and X. Liu, **"Face anti-spoofing using patch and depth-based CNNs"** 2017 IEEE International Joint Conference on Biometrics (IJCB), Denver, CO, 2017, pp. 319-328, https://doi.org/10.1109/BTAS.2017.8272713.

[25] Zoph, Barret & Vasudevan, Vijay & Shlens, Jonathon & Le, Quoc. (2018). ***"Learning Transferable Architectures for Scalable Image Recognition"***. 8697-8710. https://doi.org/10.1109/CVPR.2018.00907.

# Webography

[1] Browns, T. (n.d.). **Robust video-based eye tracking using recursive estimation of pupil characteristics**. GitHub. https://github.com/tbrouns/eyestalker

[2] Center for Biometrics and Security Research. (n.d.). **CBSR - Databases**. http://www.cbsr.ia.ac.cn/IrisDatabase.htm

[3] Chauhan, S. (2020, January 12). **Understanding mean Average Precision for Object Detection (with Python Code)**. Medium. https://medium.com/analytics-vidhya/map-mean-average-precision-for-object-detection-with-simple-python-demonstration-dcc7b3850a07

[4] Dalmaijer, E. (n.d.). **PyGaze - the open-source toolbox for eye tracking**. GitHub. https://github.com/esdalmaijer/PyGaze

[5] Dutta, A. (n.d.). **Pupil Detection**. GitHub. https://github.com/arnavdutta/Pupil-Detection

[6] Flores, T. (2019, April 4). **Median Filtering with Python and OpenCV** - Tony Flores. Medium. https://medium.com/@florestony5454/median-filtering-with-python-and-opencv-2bce390be0d1

[7] Fuhl, W. (n.d.). **Algorithms And Data Public**. Algorithms And Data Public. https://atreus.informatik.uni-tuebingen.de/seafile/d/8e2ab8c3fdd444e1a135/

[8] Garon, M. (n.d.). **Detection mAP (Mean Average Precision)**. GitHub. https://github.com/MathGaron/mean_average_precision

[9] Hui, J. (2020, February 6). **mAP (mean Average Precision) for Object Detection** - Jonathan Hui. Medium. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[10] Karim, R. (2020, June 26). **Illustrated: 10 CNN Architectures** - Towards Data Science. Medium. https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

[11] Long, A. (2019, September 16). **Understanding Data Science Classification Metrics in Scikit-Learn in Python**. Medium. https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019

[12] **Max-Planck-Institut für Informatik: Labelled Pupils in the Wild (LPW)**. (n.d.). Max-Planck-Institut Für Informatik. https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/labelled-pupils-in-the-wild-lpw/

[13] **Measuring Object Detection models - mAP - What is Mean Average Precision?** (2018, January 27). Tarang Shah - Blog. https://tarangshah.com/blog/2018-01-27/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-models/

[14] Open Datasets. (n.d.). **Pathmind**. https://pathmind.com/wiki/open-datasets?fbclid=IwAR0emLbn6L9rKrTedPb8Je3al5fx0PywHSjf3rdndZcGFyy6RyJ5eUv_bUE

[15] OpenVINO Toolkit. (n.d.). **Open Model Zoo repository**. GitHub. https://github.com/openvinotoolkit/open_model_zoo

[16] Parkhurst, D. (n.d.). **Starburst Algorithm**. GitHub. https://github.com/thirtysixthspan/Starburst

[17] Real Python. (2020, August 7). **Primer on Python Decorators**. Primer on Python Decorators. https://realpython.com/primer-on-python-decorators/

[18] Rosebrock, A. (2020, April 18). **Training a custom dlib shape predictor**. PyImageSearch. https://www.pyimagesearch.com/2019/12/16/training-a-custom-dlib-shape-predictor/

[19] Ruizendaal, R. (2018, October 21). **Deep Learning #3: More on CNNs & Handling Overfitting**. Medium. https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d

[20] Tan, R. J. (2020, July 6). **Breaking Down Mean Average Precision (mAP)** - Towards Data Science. Medium. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

[21] Tsang, S. (2019, August 1). **Review: NASNet — Neural Architecture Search Network (Image Classification)**. Medium. https://medium.com/@sh.tsang/review-nasnet-neural-architecture-search-network-image-classification-23139ea0425d

[22] Tsang, S. (2020, April 24). **Review: Inception-v4 — Evolved From GoogLeNet, Merged with ResNet Idea (Image Classification)**. Medium. https://towardsdatascience.com/review-inception-v4-evolved-from-googlenet-merged-with-resnet-idea-image-classification-5e8c339d18bc

[23] Tyszka, M. (n.d.). **MRGAZE Gaze Tracking Package**. GitHub. https://github.com/jmtyszka/mrgaze

# Annexes

In this section a collection of data is provided to help understanding the chapters of the master thesis. In order to facilitate the reading the information has been structured in the following sections.

- A1 Installation instructions and requirements to execute the software
- A2 Starburst based solution results
- A3 CNN based solution results

# A1. Installation instructions and requirements to execute the software

This section is aimed to explain the steps and software required to run the code in a device right from the beginning. The requirements stated here are the ones that have been satisfactorily tested and work without any problem. However, this does not exclude the possibility of using the code with other configurations since the code is in Python language and, theoretically, can be executed on any device that has the interpreter and the necessary libraries installed correctly.

## A1.1. Software required

The used software and its versions are the following:

- Ubuntu 2018.04
- PyCharm Community Edition 2020.1
- Anaconda3 2019.10

## A1.2. Steps to install and run the code

1. Install the software specified in the previous section in the Ubuntu machine.
2. Create the Anaconda virtual environment by restoring it from the environment file called *environment.yml.* To do so the following command can be used.

   *conda env create -f environment.yml*

3. Download the code from GIT or a ZIP file and place it in the desired location.
4. Open PyCharm IDE and open the project from the folder where it has been stored.
5. Download the LPW dataset from the following URL.

   *https://www.kaggle.com/andresrios/lpw-labeled-pupils-in-the-wild*

6. Decompress the downloaded dataset and place it inside the folder *data/lpw/.*
7. Run one of the following files.

   - **test_starburst.py:** By running this file a pupil can be detected on a video using the Starburst algorithm.
   - **test_cnn.py:** By running this file a pupil can be detected on a video using the Google's Inception-V4 CNN.

The table below shows all the libraries installed in the Anaconda virtual environment. The interpreter used was Python 3.6.

| LIBRARY | VERSION |
|---|---|
| _libgcc_mutex | 0.1 |
| _tflow_select | 2.3.0 |
| absl-py | 0.9.0 |
| astor | 0.8.0 |
| blas | 1 |
| blinker | 1.4 |
| brotlipy | 0.7.0 |
| bzip2 | 1.0.8 |
| c-ares | 1.15.0 |
| ca-certificates | 2020.6.24 |
| cachetools | 4.1.0 |
| cairo | 1.16.0 |
| certifi | 2020.6.20 |
| cffi | 1.14.0 |
| chardet | 3.0.4 |
| click | 7.1.2 |
| cloudpickle | 1.5.0 |
| cryptography | 2.9.2 |
| cycler | 0.10.0 |
| cytoolz | 0.10.1 |
| dask-core | 2.20.0 |
| dbus | 1.13.6 |
| decorator | 4.4.2 |
| dlib | 19.2 |
| expat | 2.2.6 |
| ffmpeg | 4.1.3 |
| fontconfig | 2.13.1 |
| freetype | 2.10.2 |
| gast | 0.2.2 |
| giflib | 5.2.1 |
| git | 2.23.0 |

| LIBRARY | VERSION |
|---|---|
| glib | 2.63.1 |
| gmp | 6.2.0 |
| gnutls | 3.6.13 |
| google-auth | 1.17.2 |
| google-auth-oauthlib | 0.4.1 |
| google-pasta | 0.2.0 |
| graphite2 | 1.3.13 |
| grpcio | 1.27.2 |
| gst-plugins-base | 1.14.0 |
| gstreamer | 1.14.0 |
| h5py | 2.10.0 |
| harfbuzz | 2.4.0 |
| hdf5 | 1.10.5 |
| icu | 58.2 |
| idna | 2.1 |
| imageio | 2.9.0 |
| jasper | 1.900.1 |
| jpeg | 9d |
| keras-applications | 1.0.8 |
| keras-preprocessing | 1.1.0 |
| kiwisolver | 1.2.0 |
| krb5 | 1.17.1 |
| lame | 3.1 |
| ld_impl_linux-64 | 2.33.1 |
| libblas | 3.8.0 |
| libcblas | 3.8.0 |
| libcurl | 7.69.1 |
| libedit | 3.1.20181209 |
| libffi | 3.3 |
| libgcc-ng | 9.1.0 |
| libgfortran-ng | 7.3.0 |
| libiconv | 1.15 |
| liblapack | 3.8.0 |

| LIBRARY | VERSION |
|---|---|
| liblapacke | 3.8.0 |
| libopenblas | 0.3.6 |
| libpng | 1.6.37 |
| libprotobuf | 3.12.3 |
| libssh2 | 1.9.0 |
| libstdcxx-ng | 9.1.0 |
| libtiff | 4.1.0 |
| libuuid | 2.32.1 |
| libwebp | 1.0.2 |
| libxcb | 1.13 |
| libxml2 | 2.9.9 |
| lz4-c | 1.9.2 |
| markdown | 3.1.1 |
| matplotlib | 3.1.3 |
| matplotlib-base | 3.1.3 |
| ncurses | 6.2 |
| nettle | 3.4.1 |
| networkx | 2.4 |
| numpy | 1.17.0 |
| numpy-base | 1.17.0 |
| oauthlib | 3.1.0 |
| olefile | 0.46 |
| opencv | 4.1.0 |
| openh264 | 1.8.0 |
| openssl | 1.1.1g |
| opt_einsum | 3.1.0 |
| pandas | 1.0.5 |
| patsy | 0.5.1 |
| pcre | 8.43 |
| perl | 5.26.2 |
| pillow | 7.0.0 |
| pixman | 0.38.0 |
| protobuf | 3.12.3 |

| LIBRARY | VERSION |
|---|---|
| pthread-stubs | 0.4 |
| pyasn1 | 0.4.8 |
| pyasn1-modules | 0.2.7 |
| pycparser | 2.2 |
| pyjwt | 1.7.1 |
| pyopenssl | 19.1.0 |
| pyparsing | 2.4.7 |
| pyqt | 5.9.2 |
| pysocks | 1.7.1 |
| python | 3.6.10 |
| python-dateutil | 2.8.1 |
| python_abi | 3.6 |
| pytz | 2020.1 |
| pywavelets | 1.1.1 |
| pyyaml | 5.3.1 |
| qt | 5.9.7 |
| readline | 8 |
| requests | 2.24.0 |
| requests-oauthlib | 1.3.0 |
| rsa | 4 |
| scikit-image | 0.16.2 |
| scipy | 1.5.0 |
| seaborn | 0.9.0 |
| setuptools | 46.4.0 |
| sip | 4.19.8 |
| six | 1.15.0 |
| sqlite | 3.31.1 |
| statsmodels | 0.11.1 |
| tensorboard | 2.2.1 |
| tensorboard-plugin-wit | 1.6.0 |
| tensorflow | 2.0.0 |
| tensorflow-base | 2.0.0 |
| tensorflow-estimator | 2.0.0 |

| LIBRARY | VERSION |
|---|---|
| termcolor | 1.1.0 |
| tk | 8.6.8 |
| toolz | 0.10.0 |
| tornado | 6.0.4 |
| tqdm | 4.46.0 |
| urllib3 | 1.25.9 |
| werkzeug | 0.16.1 |
| wheel | 0.34.2 |
| wrapt | 1.12.1 |
| x264 | 152.20180806 |
| xorg-kbproto | 1.0.7 |
| xorg-libice | 1.0.10 |
| xorg-libsm | 1.2.3 |
| xorg-libx11 | 1.6.9 |
| xorg-libxau | 1.0.9 |
| xorg-libxdmcp | 1.1.3 |
| xorg-libxext | 1.3.4 |
| xorg-libxrender | 0.9.10 |
| xorg-renderproto | 0.11.1 |
| xorg-xextproto | 7.3.0 |
| xorg-xproto | 7.0.31 |
| xz | 5.2.5 |
| yaml | 0.2.5 |
| zlib | 1.2.11 |
| zstd | 1.4.4 |

# A2. Starburst based solution results

In this section the tables containing the results of the execution of the Starburst based solution for pupil detection over each one of the 66 videos of the LPW dataset will be attached. Further detailed results for each one of the videos can be found inside the ZIP file.

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1/1.avi** | 0.33 | 0.56 | 0.655 | 0.705 | 0.735 | 0.765 | 0.78 | 0.7925 | 0.81 | 0.83 | 0.845 | 0.8575 | 0.8725 | 0.88 | 0.8825 |
| **1/4.avi** | 0.147869674 | 0.403508772 | 0.543859649 | 0.573934837 | 0.601503759 | 0.619047619 | 0.639097744 | 0.646616541 | 0.656641604 | 0.669172932 | 0.676691729 | 0.689223058 | 0.706766917 | 0.714285714 | 0.724310777 |
| **1/9.avi** | 0.1325 | 0.3725 | 0.455 | 0.5075 | 0.5575 | 0.6 | 0.6325 | 0.655 | 0.685 | 0.7 | 0.71 | 0.7375 | 0.7675 | 0.785 | 0.795 |
| **10/11.avi** | 0.04 | 0.0925 | 0.115 | 0.13 | 0.135 | 0.1375 | 0.14 | 0.14 | 0.1425 | 0.145 | 0.1475 | 0.1525 | 0.1525 | 0.1525 | 0.1525 |
| **10/1.avi** | 0.3225 | 0.6925 | 0.8125 | 0.845 | 0.8825 | 0.905 | 0.91 | 0.9125 | 0.9125 | 0.9125 | 0.9125 | 0.915 | 0.9175 | 0.9175 | 0.9175 |
| **10/8.avi** | 0.0975 | 0.2925 | 0.4325 | 0.5025 | 0.5625 | 0.63 | 0.67 | 0.705 | 0.7275 | 0.745 | 0.755 | 0.7625 | 0.765 | 0.765 | 0.7675 |
| **11/13.avi** | 0.0025 | 0.0225 | 0.06 | 0.0925 | 0.12 | 0.125 | 0.13 | 0.13 | 0.135 | 0.135 | 0.1375 | 0.14 | 0.1425 | 0.1425 | 0.1525 |
| **11/2.avi** | 0.0875 | 0.2725 | 0.3975 | 0.465 | 0.5075 | 0.535 | 0.555 | 0.5625 | 0.575 | 0.5875 | 0.5975 | 0.6025 | 0.605 | 0.61 | 0.6175 |
| **11/7.avi** | 0.085 | 0.3275 | 0.385 | 0.4075 | 0.42 | 0.43 | 0.44 | 0.445 | 0.4475 | 0.4525 | 0.455 | 0.4575 | 0.4575 | 0.46 | 0.465 |
| **12/1.avi** | 0.2825 | 0.4325 | 0.455 | 0.455 | 0.46 | 0.465 | 0.47 | 0.475 | 0.4775 | 0.48 | 0.48 | 0.4825 | 0.4825 | 0.4825 | 0.4825 |
| **12/2.avi** | 0.14 | 0.4275 | 0.53 | 0.5875 | 0.6225 | 0.65 | 0.675 | 0.68 | 0.705 | 0.725 | 0.755 | 0.7775 | 0.8025 | 0.835 | 0.8525 |
| **12/9.avi** | 0.0625 | 0.1925 | 0.2575 | 0.37 | 0.445 | 0.5025 | 0.5375 | 0.5775 | 0.5975 | 0.6175 | 0.6375 | 0.6725 | 0.6875 | 0.7025 | 0.725 |
| **13/1.avi** | 0.02 | 0.0525 | 0.09 | 0.12 | 0.14 | 0.155 | 0.19 | 0.2 | 0.215 | 0.2175 | 0.2275 | 0.2475 | 0.265 | 0.2875 | 0.3075 |
| **13/2.avi** | 0.0125 | 0.1125 | 0.1775 | 0.19 | 0.21 | 0.2125 | 0.2325 | 0.2375 | 0.2425 | 0.2525 | 0.2525 | 0.2525 | 0.2625 | 0.2625 | 0.2675 |
| **13/9.avi** | 0.004444444 | 0.026666667 | 0.048888889 | 0.075555556 | 0.088888889 | 0.124444444 | 0.155555556 | 0.2 | 0.248888889 | 0.275555556 | 0.302222222 | 0.337777778 | 0.377777778 | 0.413333333 | 0.44 |
| **14/10.avi** | 0.15 | 0.585 | 0.735 | 0.8 | 0.8525 | 0.87 | 0.89 | 0.9 | 0.91 | 0.92 | 0.92 | 0.9225 | 0.9225 | 0.925 | 0.9275 |
| **14/17.avi** | 0.02 | 0.0875 | 0.105 | 0.1175 | 0.1275 | 0.1375 | 0.16 | 0.1825 | 0.2 | 0.2175 | 0.225 | 0.245 | 0.265 | 0.28 | 0.3 |
| **14/22.avi** | 0.2375 | 0.59 | 0.7325 | 0.81 | 0.8425 | 0.87 | 0.885 | 0.895 | 0.9075 | 0.9225 | 0.9275 | 0.9275 | 0.93 | 0.935 | 0.9375 |
| **15/1.avi** | 0.01 | 0.0325 | 0.05 | 0.075 | 0.1025 | 0.1175 | 0.1475 | 0.1675 | 0.185 | 0.2175 | 0.2275 | 0.24 | 0.2525 | 0.2725 | 0.295 |
| **15/2.avi** | 0.11 | 0.21 | 0.235 | 0.265 | 0.285 | 0.295 | 0.3075 | 0.3375 | 0.365 | 0.375 | 0.385 | 0.4025 | 0.42 | 0.44 | 0.455 |
| **15/7.avi** | 0.03 | 0.1075 | 0.19 | 0.245 | 0.31 | 0.345 | 0.3675 | 0.39 | 0.405 | 0.425 | 0.4525 | 0.465 | 0.4825 | 0.5025 | 0.5125 |

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **16/13.avi** | 0.023121387 | 0.147398844 | 0.210982659 | 0.289017341 | 0.341040462 | 0.401734104 | 0.433526012 | 0.450867052 | 0.48265896 | 0.505780347 | 0.514450867 | 0.523121387 | 0.531791908 | 0.537572254 | 0.537572254 |
| **16/1.avi** | 0.2775 | 0.6125 | 0.6675 | 0.685 | 0.695 | 0.705 | 0.705 | 0.7075 | 0.7125 | 0.715 | 0.72 | 0.7225 | 0.7225 | 0.73 | 0.74 |
| **16/2.avi** | 0.175 | 0.385 | 0.465 | 0.5075 | 0.54 | 0.575 | 0.5925 | 0.605 | 0.6425 | 0.655 | 0.665 | 0.685 | 0.69 | 0.7175 | 0.7325 |
| **17/12.avi** | 0 | 0 | 0 | 0 | 0.0025 | 0.0025 | 0.005 | 0.005 | 0.0075 | 0.0075 | 0.0075 | 0.0075 | 0.0075 | 0.01 | 0.01 |
| **17/3.avi** | 0.1425 | 0.495 | 0.57 | 0.6125 | 0.645 | 0.67 | 0.69 | 0.695 | 0.705 | 0.7075 | 0.7075 | 0.71 | 0.7125 | 0.7125 | 0.7125 |
| **17/5.avi** | 0.155 | 0.4975 | 0.6625 | 0.775 | 0.8325 | 0.8925 | 0.935 | 0.945 | 0.965 | 0.9725 | 0.9775 | 0.98 | 0.9825 | 0.985 | 0.985 |
| **18/11.avi** | 0.005 | 0.0275 | 0.045 | 0.0575 | 0.0775 | 0.1 | 0.105 | 0.14 | 0.1875 | 0.2275 | 0.28 | 0.335 | 0.375 | 0.42 | 0.4525 |
| **18/2.avi** | 0.2275 | 0.4975 | 0.555 | 0.61 | 0.635 | 0.6675 | 0.69 | 0.73 | 0.76 | 0.8 | 0.84 | 0.885 | 0.925 | 0.9375 | 0.96 |
| **18/7.avi** | 0.0825 | 0.555 | 0.6775 | 0.72 | 0.745 | 0.755 | 0.7675 | 0.7775 | 0.7775 | 0.795 | 0.7975 | 0.7975 | 0.7975 | 0.7975 | 0.8 |
| **19/2.avi** | 0.0525 | 0.1325 | 0.1975 | 0.245 | 0.2725 | 0.2825 | 0.305 | 0.325 | 0.35 | 0.36 | 0.3675 | 0.3725 | 0.375 | 0.38 | 0.3875 |
| **19/3.avi** | 0.0675 | 0.2375 | 0.3475 | 0.425 | 0.4675 | 0.4825 | 0.5075 | 0.525 | 0.5425 | 0.56 | 0.5725 | 0.5725 | 0.5825 | 0.595 | 0.6025 |
| **19/6.avi** | 0.0225 | 0.0825 | 0.1225 | 0.155 | 0.185 | 0.2125 | 0.2375 | 0.2575 | 0.2675 | 0.28 | 0.285 | 0.29 | 0.3075 | 0.3175 | 0.325 |
| **2/10.avi** | 0 | 0.0025 | 0.0025 | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 | 0.0075 | 0.0125 | 0.0175 | 0.02 | 0.025 | 0.0375 | 0.04 |
| **2/13.avi** | 0.0375 | 0.105 | 0.1725 | 0.2025 | 0.22 | 0.2375 | 0.2575 | 0.275 | 0.295 | 0.3225 | 0.3375 | 0.355 | 0.3575 | 0.3725 | 0.3825 |
| **2/4.avi** | 0.095 | 0.245 | 0.3375 | 0.39 | 0.415 | 0.4325 | 0.445 | 0.4575 | 0.465 | 0.475 | 0.48 | 0.495 | 0.5075 | 0.5125 | 0.53 |
| **20/3.avi** | 0.0625 | 0.19 | 0.2925 | 0.345 | 0.3825 | 0.3975 | 0.405 | 0.43 | 0.445 | 0.455 | 0.4725 | 0.48 | 0.495 | 0.4975 | 0.5 |
| **20/4.avi** | 0.025 | 0.0625 | 0.1075 | 0.125 | 0.145 | 0.17 | 0.1875 | 0.1925 | 0.195 | 0.2025 | 0.21 | 0.22 | 0.225 | 0.23 | 0.24 |
| **20/7.avi** | 0.1275 | 0.3925 | 0.5525 | 0.635 | 0.68 | 0.7025 | 0.7175 | 0.7175 | 0.73 | 0.7325 | 0.735 | 0.745 | 0.7525 | 0.76 | 0.7625 |
| **21/11.avi** | 0.1075 | 0.3375 | 0.5475 | 0.6275 | 0.68 | 0.715 | 0.735 | 0.75 | 0.76 | 0.7775 | 0.79 | 0.795 | 0.81 | 0.8175 | 0.82 |
| **21/12.avi** | 0.08 | 0.3875 | 0.6975 | 0.8275 | 0.8975 | 0.9225 | 0.9375 | 0.9425 | 0.95 | 0.9525 | 0.955 | 0.9575 | 0.96 | 0.965 | 0.9675 |
| **21/4.avi** | 0.0475 | 0.2525 | 0.35 | 0.4425 | 0.5275 | 0.5725 | 0.6 | 0.615 | 0.62 | 0.6225 | 0.625 | 0.6325 | 0.64 | 0.645 | 0.645 |
| **22/17.avi** | 0.0025 | 0.05 | 0.0625 | 0.065 | 0.065 | 0.07 | 0.07 | 0.07 | 0.07 | 0.07 | 0.075 | 0.075 | 0.0775 | 0.0775 | 0.0775 |
| **22/1.avi** | 0.0275 | 0.07 | 0.0875 | 0.1075 | 0.125 | 0.135 | 0.155 | 0.16 | 0.1675 | 0.18 | 0.1825 | 0.19 | 0.1925 | 0.2075 | 0.215 |
| **22/2.avi** | 0.12 | 0.31 | 0.3575 | 0.38 | 0.3975 | 0.4025 | 0.4125 | 0.4125 | 0.4175 | 0.4225 | 0.4225 | 0.4225 | 0.4225 | 0.425 | 0.425 |
| **3/16.avi** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0025 | 0.005 | 0.005 | 0.005 | 0.005 | 0.0075 | 0.0075 | 0.0075 |
| **3/19.avi** | 0.015 | 0.0575 | 0.09 | 0.145 | 0.175 | 0.1925 | 0.2175 | 0.2425 | 0.26 | 0.2825 | 0.2975 | 0.31 | 0.3175 | 0.345 | 0.355 |
| **3/21.avi** | 0.225 | 0.4525 | 0.5525 | 0.5975 | 0.62 | 0.63 | 0.6475 | 0.665 | 0.675 | 0.6825 | 0.69 | 0.7 | 0.7125 | 0.7375 | 0.76 |

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4/12.avi** | 0.0175 | 0.0825 | 0.1325 | 0.17 | 0.21 | 0.2475 | 0.29 | 0.3175 | 0.3625 | 0.3975 | 0.415 | 0.4375 | 0.45 | 0.47 | 0.4925 |
| **4/1.avi** | 0 | 0 | 0.0075 | 0.0175 | 0.0275 | 0.0275 | 0.0325 | 0.045 | 0.0475 | 0.05 | 0.0525 | 0.0625 | 0.0825 | 0.09 | 0.1 |
| **4/2.avi** | 0.06 | 0.1575 | 0.2025 | 0.235 | 0.2725 | 0.3075 | 0.3375 | 0.37 | 0.3875 | 0.4075 | 0.4325 | 0.4625 | 0.48 | 0.495 | 0.5075 |
| **5/10.avi** | 0.005 | 0.02 | 0.025 | 0.03 | 0.0425 | 0.0575 | 0.06 | 0.0725 | 0.075 | 0.09 | 0.0925 | 0.1075 | 0.115 | 0.1225 | 0.1275 |
| **5/11.avi** | 0 | 0 | 0 | 0.0025 | 0.0025 | 0.0025 | 0.005 | 0.0075 | 0.0125 | 0.015 | 0.0175 | 0.0225 | 0.0225 | 0.0375 | 0.0375 |
| **5/6.avi** | 0 | 0 | 0 | 0.0025 | 0.0025 | 0.005 | 0.005 | 0.0075 | 0.01 | 0.0125 | 0.015 | 0.015 | 0.02 | 0.0225 | 0.03 |
| **6/13.avi** | 0.0525 | 0.2275 | 0.4225 | 0.515 | 0.5775 | 0.6 | 0.62 | 0.6425 | 0.665 | 0.675 | 0.6825 | 0.685 | 0.6925 | 0.7025 | 0.71 |
| **6/2.avi** | 0.112781955 | 0.293233083 | 0.398496241 | 0.468671679 | 0.516290727 | 0.538847118 | 0.553884712 | 0.558897243 | 0.568922306 | 0.576441103 | 0.578947368 | 0.586466165 | 0.593984962 | 0.601503759 | 0.611528822 |
| **6/5.avi** | 0.015 | 0.2675 | 0.6 | 0.7275 | 0.8325 | 0.86 | 0.885 | 0.9025 | 0.9125 | 0.92 | 0.935 | 0.9375 | 0.9425 | 0.9425 | 0.945 |
| **7/15.avi** | 0.16 | 0.295 | 0.34 | 0.37 | 0.3975 | 0.4175 | 0.42 | 0.4425 | 0.45 | 0.4625 | 0.475 | 0.4875 | 0.5 | 0.5075 | 0.5275 |
| **7/18.avi** | 0.2525 | 0.655 | 0.7275 | 0.7775 | 0.8025 | 0.825 | 0.8475 | 0.8575 | 0.875 | 0.89 | 0.9 | 0.9025 | 0.9125 | 0.9225 | 0.93 |
| **7/21.avi** | 0.0575 | 0.1975 | 0.245 | 0.275 | 0.2875 | 0.3 | 0.3025 | 0.3075 | 0.315 | 0.3175 | 0.32 | 0.325 | 0.33 | 0.335 | 0.3375 |
| **8/2.avi** | 0.3575 | 0.48 | 0.5225 | 0.5675 | 0.64 | 0.68 | 0.7075 | 0.73 | 0.745 | 0.7625 | 0.78 | 0.81 | 0.825 | 0.8525 | 0.8625 |
| **8/7.avi** | 0.205 | 0.4775 | 0.5025 | 0.54 | 0.5525 | 0.565 | 0.5775 | 0.58 | 0.5875 | 0.6025 | 0.61 | 0.61 | 0.615 | 0.6175 | 0.62 |
| **8/9.avi** | 0.0075 | 0.025 | 0.045 | 0.0575 | 0.0725 | 0.08 | 0.0925 | 0.11 | 0.1275 | 0.1575 | 0.1825 | 0.22 | 0.2475 | 0.275 | 0.3 |
| **9/16.avi** | 0.0775 | 0.155 | 0.19 | 0.205 | 0.2175 | 0.245 | 0.255 | 0.265 | 0.275 | 0.2875 | 0.3025 | 0.31 | 0.325 | 0.33 | 0.3425 |
| **9/17.avi** | 0.0325 | 0.085 | 0.12 | 0.1625 | 0.2225 | 0.2925 | 0.33 | 0.38 | 0.425 | 0.465 | 0.51 | 0.5325 | 0.565 | 0.59 | 0.615 |
| **9/18.avi** | 0.39 | 0.795 | 0.85 | 0.885 | 0.905 | 0.9275 | 0.935 | 0.9375 | 0.9425 | 0.9475 | 0.9525 | 0.9575 | 0.9575 | 0.9575 | 0.96 |
| **TOTAL:** | **0.095919962** | **0.252891021** | **0.326132234** | **0.368555749** | **0.39943521** | **0.421235959** | **0.438440364** | **0.452520922** | **0.466622905** | **0.479385605** | **0.489580488** | **0.500630127** | **0.510989721** | **0.521427198** | **0.530392604** |

*Table 7. Table containing the summary of detection rates obtained after running the CNN based solution over all the videos of the LPW dataset.*

# A3. CNN based solution results

In this section the tables containing the results of the execution of the CNN based solution for pupil detection over each one of the 66 videos of the LPW dataset will be attached. Further detailed results for each one of the videos can be found inside the ZIP file.

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1/1.avi** | 0.155 | 0.365 | 0.59 | 0.69 | 0.7425 | 0.78 | 0.825 | 0.8425 | 0.8575 | 0.885 | 0.8975 | 0.9075 | 0.9325 | 0.95 | 0.96 |
| **1/4.avi** | 0.25 | 0.71 | 0.9 | 0.9575 | 0.9725 | 0.985 | 0.99 | 0.995 | 0.995 | 0.995 | 0.995 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **1/9.avi** | 0.1125 | 0.2775 | 0.4525 | 0.5725 | 0.64 | 0.705 | 0.775 | 0.825 | 0.8725 | 0.89 | 0.9325 | 0.9375 | 0.9375 | 0.9375 | 0.94 |
| **10/11.avi** | 0.0975 | 0.18 | 0.3025 | 0.3975 | 0.4475 | 0.485 | 0.525 | 0.5425 | 0.5625 | 0.575 | 0.59 | 0.5925 | 0.5975 | 0.5975 | 0.6075 |
| **10/1.avi** | 0.0925 | 0.3125 | 0.49 | 0.6475 | 0.735 | 0.7775 | 0.815 | 0.85 | 0.875 | 0.895 | 0.9175 | 0.925 | 0.9325 | 0.9375 | 0.945 |
| **10/8.avi** | 0.1125 | 0.3525 | 0.6275 | 0.8225 | 0.9325 | 0.945 | 0.9675 | 0.9775 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.985 |
| **11/13.avi** | 0.0225 | 0.1275 | 0.2975 | 0.4925 | 0.58 | 0.6825 | 0.725 | 0.7725 | 0.8 | 0.8225 | 0.8425 | 0.8575 | 0.8725 | 0.88 | 0.885 |
| **11/2.avi** | 0.125 | 0.2975 | 0.505 | 0.62 | 0.72 | 0.785 | 0.8275 | 0.8525 | 0.87 | 0.895 | 0.9075 | 0.915 | 0.925 | 0.9325 | 0.9425 |
| **11/7.avi** | 0.0775 | 0.3175 | 0.56 | 0.685 | 0.7625 | 0.8 | 0.8425 | 0.8775 | 0.9025 | 0.915 | 0.9325 | 0.94 | 0.95 | 0.9525 | 0.955 |
| **12/1.avi** | 0.14 | 0.4325 | 0.5725 | 0.6925 | 0.7225 | 0.765 | 0.795 | 0.825 | 0.8525 | 0.875 | 0.8875 | 0.91 | 0.935 | 0.97 | 0.9725 |
| **12/2.avi** | 0.1825 | 0.4375 | 0.6375 | 0.7825 | 0.855 | 0.875 | 0.9 | 0.9275 | 0.9425 | 0.9625 | 0.9775 | 0.9825 | 0.985 | 0.99 | 0.99 |
| **12/9.avi** | 0.0725 | 0.325 | 0.51 | 0.71 | 0.81 | 0.8775 | 0.9225 | 0.95 | 0.97 | 0.9775 | 0.9925 | 0.995 | 0.995 | 0.995 | 0.995 |
| **13/1.avi** | 0.025 | 0.11 | 0.175 | 0.285 | 0.345 | 0.42 | 0.495 | 0.605 | 0.6625 | 0.695 | 0.7375 | 0.785 | 0.8125 | 0.8475 | 0.8625 |
| **13/2.avi** | 0.0375 | 0.205 | 0.3525 | 0.4875 | 0.5675 | 0.64 | 0.7075 | 0.7475 | 0.7625 | 0.7925 | 0.825 | 0.855 | 0.885 | 0.895 | 0.915 |
| **13/9.avi** | 0.066371681 | 0.225663717 | 0.438053097 | 0.610619469 | 0.774336283 | 0.89380531 | 0.924778761 | 0.955752212 | 0.969026549 | 0.969026549 | 0.982300885 | 0.986725664 | 0.986725664 | 0.986725664 | 0.986725664 |
| **14/10.avi** | 0.1 | 0.37 | 0.61 | 0.7425 | 0.8375 | 0.885 | 0.9175 | 0.9325 | 0.9475 | 0.955 | 0.965 | 0.9875 | 0.995 | 0.9975 | 0.9975 |
| **14/17.avi** | 0.1025 | 0.2675 | 0.4225 | 0.62 | 0.7325 | 0.8175 | 0.885 | 0.9125 | 0.9375 | 0.9525 | 0.9575 | 0.9625 | 0.965 | 0.965 | 0.97 |
| **14/22.avi** | 0.07 | 0.3675 | 0.57 | 0.725 | 0.805 | 0.8725 | 0.9075 | 0.9375 | 0.9525 | 0.965 | 0.9725 | 0.9775 | 0.985 | 0.99 | 0.99 |
| **15/1.avi** | 0.055 | 0.2075 | 0.3825 | 0.49 | 0.5875 | 0.6625 | 0.72 | 0.7725 | 0.8325 | 0.875 | 0.8975 | 0.9275 | 0.9375 | 0.9475 | 0.9625 |
| **15/2.avi** | 0.1025 | 0.305 | 0.495 | 0.6225 | 0.715 | 0.8175 | 0.875 | 0.9175 | 0.94 | 0.9575 | 0.965 | 0.975 | 0.9825 | 0.9875 | 0.995 |
| **15/7.avi** | 0.04 | 0.155 | 0.335 | 0.51 | 0.65 | 0.75 | 0.7825 | 0.84 | 0.8725 | 0.9025 | 0.92 | 0.9275 | 0.94 | 0.9425 | 0.945 |

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **16/13.avi** | 0.080691643 | 0.270893372 | 0.484149856 | 0.659942363 | 0.780979827 | 0.855907781 | 0.919308357 | 0.956772334 | 0.976945245 | 0.988472622 | 0.994236311 | 0.994236311 | 0.994236311 | 0.997118156 | 0.997118156 |
| **16/1.avi** | 0.1925 | 0.635 | 0.885 | 0.9725 | 0.985 | 0.995 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **16/2.avi** | 0.21 | 0.59 | 0.825 | 0.9275 | 0.9525 | 0.97 | 0.9725 | 0.9775 | 0.9775 | 0.9825 | 0.99 | 0.99 | 0.99 | 0.99 | 0.9925 |
| **17/12.avi** | 0.0725 | 0.2375 | 0.435 | 0.645 | 0.8075 | 0.8975 | 0.9525 | 0.975 | 0.985 | 0.9875 | 0.9875 | 0.9875 | 0.9875 | 0.9875 | 0.9875 |
| **17/3.avi** | 0.1825 | 0.5075 | 0.6525 | 0.755 | 0.8375 | 0.9175 | 0.945 | 0.97 | 0.98 | 0.99 | 0.9925 | 0.9925 | 0.9925 | 0.9925 | 0.9925 |
| **17/5.avi** | 0.2525 | 0.5775 | 0.7075 | 0.785 | 0.86 | 0.94 | 0.9775 | 0.985 | 0.9875 | 0.9875 | 0.9875 | 0.99 | 0.99 | 0.99 | 0.99 |
| **18/11.avi** | 0.11 | 0.315 | 0.59 | 0.77 | 0.855 | 0.9025 | 0.9475 | 0.9775 | 0.9925 | 0.995 | 0.995 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **18/2.avi** | 0.085 | 0.3075 | 0.55 | 0.685 | 0.7975 | 0.85 | 0.9075 | 0.9325 | 0.9625 | 0.975 | 0.985 | 0.9925 | 0.995 | 0.9975 | 0.9975 |
| **18/7.avi** | 0.0275 | 0.24 | 0.6075 | 0.8625 | 0.9425 | 0.96 | 0.9725 | 0.975 | 0.975 | 0.985 | 0.9925 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **19/2.avi** | 0.0475 | 0.245 | 0.4825 | 0.6425 | 0.765 | 0.8575 | 0.915 | 0.9425 | 0.965 | 0.97 | 0.9725 | 0.9775 | 0.9825 | 0.9825 | 0.9825 |
| **19/3.avi** | 0.06 | 0.225 | 0.46 | 0.635 | 0.765 | 0.8475 | 0.905 | 0.92 | 0.9425 | 0.97 | 0.9775 | 0.9775 | 0.9775 | 0.9775 | 0.98 |
| **19/6.avi** | 0.05 | 0.185 | 0.275 | 0.335 | 0.405 | 0.4675 | 0.4975 | 0.5275 | 0.5625 | 0.5725 | 0.575 | 0.5775 | 0.5875 | 0.59 | 0.595 |
| **2/10.avi** | 0.13 | 0.37 | 0.5575 | 0.7 | 0.7775 | 0.86 | 0.9175 | 0.95 | 0.965 | 0.9725 | 0.975 | 0.9825 | 0.9825 | 0.9825 | 0.985 |
| **2/13.avi** | 0.17 | 0.475 | 0.67 | 0.8475 | 0.9225 | 0.98 | 0.9925 | 0.995 | 0.995 | 0.995 | 0.995 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **2/4.avi** | 0.1625 | 0.55 | 0.8375 | 0.955 | 0.985 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **20/3.avi** | 0.1325 | 0.3825 | 0.6025 | 0.7325 | 0.8225 | 0.89 | 0.9375 | 0.9675 | 0.975 | 0.985 | 0.9925 | 0.9925 | 0.9925 | 0.9925 | 0.9925 |
| **20/4.avi** | 0.26 | 0.6425 | 0.8125 | 0.92 | 0.96 | 0.975 | 0.9775 | 0.99 | 0.9925 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 | 0.9975 |
| **20/7.avi** | 0.2675 | 0.6525 | 0.8625 | 0.9375 | 0.97 | 0.9775 | 0.985 | 0.9875 | 0.9925 | 0.9925 | 0.9925 | 0.9925 | 0.9925 | 0.9925 | 0.9925 |
| **21/11.avi** | 0.0775 | 0.345 | 0.7225 | 0.9075 | 0.9275 | 0.935 | 0.9425 | 0.9425 | 0.9475 | 0.9475 | 0.955 | 0.955 | 0.96 | 0.96 | 0.9625 |
| **21/12.avi** | 0.1625 | 0.655 | 0.89 | 0.9525 | 0.9625 | 0.97 | 0.975 | 0.985 | 0.985 | 0.985 | 0.985 | 0.985 | 0.99 | 0.99 | 0.99 |
| **21/4.avi** | 0.0825 | 0.335 | 0.5825 | 0.755 | 0.825 | 0.8425 | 0.8625 | 0.885 | 0.9025 | 0.91 | 0.9175 | 0.9225 | 0.9275 | 0.9275 | 0.93 |
| **22/17.avi** | 0.1225 | 0.305 | 0.4225 | 0.5175 | 0.5825 | 0.6425 | 0.6875 | 0.7075 | 0.745 | 0.7675 | 0.7875 | 0.8075 | 0.825 | 0.8375 | 0.85 |
| **22/1.avi** | 0.135 | 0.3325 | 0.4825 | 0.56 | 0.635 | 0.725 | 0.8025 | 0.8625 | 0.8975 | 0.9275 | 0.955 | 0.9625 | 0.98 | 0.9825 | 0.985 |
| **22/2.avi** | 0.0925 | 0.32 | 0.59 | 0.78 | 0.9 | 0.9475 | 0.9625 | 0.975 | 0.9825 | 0.985 | 0.9875 | 0.9875 | 0.9875 | 0.9875 | 0.9875 |
| **3/16.avi** | 0.025 | 0.12 | 0.265 | 0.375 | 0.4625 | 0.5175 | 0.55 | 0.58 | 0.5975 | 0.6175 | 0.64 | 0.6575 | 0.67 | 0.6825 | 0.6975 |
| **3/19.avi** | 0.0325 | 0.1575 | 0.2975 | 0.4425 | 0.56 | 0.655 | 0.7125 | 0.7725 | 0.83 | 0.87 | 0.9 | 0.925 | 0.9425 | 0.9525 | 0.9575 |
| **3/21.avi** | 0.2525 | 0.6075 | 0.8225 | 0.9025 | 0.95 | 0.96 | 0.965 | 0.97 | 0.97 | 0.97 | 0.9725 | 0.9725 | 0.9725 | 0.9725 | 0.9725 |

| VIDEO | THOLD. 1 | THOLD. 2 | THOLD. 3 | THOLD. 4 | THOLD. 5 | THOLD. 6 | THOLD. 7 | THOLD. 8 | THOLD. 9 | THOLD. 10 | THOLD. 11 | THOLD. 12 | THOLD. 13 | THOLD. 14 | THOLD. 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4/12.avi** | 0.015 | 0.065 | 0.1975 | 0.3225 | 0.4325 | 0.55 | 0.605 | 0.645 | 0.69 | 0.725 | 0.7475 | 0.7775 | 0.81 | 0.83 | 0.85 |
| **4/1.avi** | 0.01 | 0.0625 | 0.125 | 0.21 | 0.2775 | 0.37 | 0.435 | 0.49 | 0.525 | 0.5725 | 0.6425 | 0.6675 | 0.695 | 0.72 | 0.7475 |
| **4/2.avi** | 0.0975 | 0.3625 | 0.6 | 0.7825 | 0.8625 | 0.9075 | 0.93 | 0.94 | 0.945 | 0.955 | 0.9625 | 0.9675 | 0.97 | 0.9725 | 0.9725 |
| **5/10.avi** | 0.055 | 0.135 | 0.2125 | 0.295 | 0.365 | 0.4425 | 0.505 | 0.5425 | 0.5775 | 0.62 | 0.6525 | 0.675 | 0.7175 | 0.74 | 0.755 |
| **5/11.avi** | 0.055 | 0.2375 | 0.3925 | 0.5175 | 0.57 | 0.605 | 0.635 | 0.6575 | 0.68 | 0.7025 | 0.715 | 0.73 | 0.7425 | 0.76 | 0.77 |
| **5/6.avi** | 0 | 0.0075 | 0.01 | 0.0275 | 0.0425 | 0.0675 | 0.0975 | 0.1375 | 0.1625 | 0.23 | 0.275 | 0.32 | 0.3675 | 0.385 | 0.415 |
| **6/13.avi** | 0.025 | 0.135 | 0.3375 | 0.4975 | 0.595 | 0.6825 | 0.725 | 0.7625 | 0.7875 | 0.8125 | 0.8225 | 0.83 | 0.84 | 0.8475 | 0.8575 |
| **6/2.avi** | 0.2825 | 0.535 | 0.6725 | 0.755 | 0.795 | 0.845 | 0.865 | 0.8925 | 0.9125 | 0.935 | 0.9425 | 0.9475 | 0.955 | 0.9675 | 0.97 |
| **6/5.avi** | 0.0275 | 0.1 | 0.345 | 0.575 | 0.7275 | 0.825 | 0.875 | 0.8925 | 0.91 | 0.9175 | 0.93 | 0.9425 | 0.955 | 0.9625 | 0.9675 |
| **7/15.avi** | 0.1725 | 0.4225 | 0.625 | 0.78 | 0.8275 | 0.855 | 0.8825 | 0.8975 | 0.905 | 0.925 | 0.9425 | 0.955 | 0.96 | 0.97 | 0.98 |
| **7/18.avi** | 0.2975 | 0.6125 | 0.78 | 0.85 | 0.8825 | 0.93 | 0.96 | 0.9625 | 0.9725 | 0.9775 | 0.98 | 0.985 | 0.9875 | 0.9925 | 0.9925 |
| **7/21.avi** | 0.17 | 0.3875 | 0.56 | 0.685 | 0.74 | 0.7625 | 0.8 | 0.8225 | 0.83 | 0.84 | 0.85 | 0.865 | 0.87 | 0.8775 | 0.885 |
| **8/2.avi** | 0.185 | 0.44 | 0.635 | 0.7425 | 0.7975 | 0.835 | 0.8575 | 0.8875 | 0.9125 | 0.95 | 0.965 | 0.97 | 0.985 | 0.985 | 0.985 |
| **8/7.avi** | 0.1775 | 0.585 | 0.82 | 0.865 | 0.8875 | 0.905 | 0.9175 | 0.925 | 0.9325 | 0.935 | 0.95 | 0.9575 | 0.9625 | 0.97 | 0.97 |
| **8/9.avi** | 0.145 | 0.39 | 0.62 | 0.7575 | 0.83 | 0.8725 | 0.9 | 0.92 | 0.9325 | 0.94 | 0.9475 | 0.9525 | 0.955 | 0.96 | 0.965 |
| **9/16.avi** | 0.06 | 0.2325 | 0.345 | 0.4375 | 0.4875 | 0.555 | 0.645 | 0.7075 | 0.745 | 0.79 | 0.82 | 0.86 | 0.895 | 0.92 | 0.935 |
| **9/17.avi** | 0.03 | 0.145 | 0.255 | 0.405 | 0.5575 | 0.6825 | 0.7725 | 0.845 | 0.885 | 0.9275 | 0.95 | 0.9675 | 0.9725 | 0.9825 | 0.985 |
| **9/18.avi** | 0.325 | 0.71 | 0.84 | 0.8875 | 0.915 | 0.945 | 0.965 | 0.97 | 0.98 | 0.985 | 0.985 | 0.985 | 0.985 | 0.9875 | 0.9875 |
| **TOTAL:** | **0.115826717** | **0.340894804** | **0.529843984** | **0.659326694** | **0.735194183** | **0.790828986** | **0.828395259** | **0.855454917** | **0.874560179** | **0.890795442** | **0.90381117** | **0.913423666** | **0.922097909** | **0.9281643** | **0.933202179** |

*Table 8. Table containing the summary of detection rates obtained after running the CNN based solution over all the videos of the LPW dataset.*