Universidade do Minho
Escola de Engenharia

Diogo Filipe Rodrigues Ferreira

A system for drivers drowsiness detection

Outubro de 2022

Universidade do Minho
Escola de Engenharia

Diogo Filipe Rodrigues Ferreira

A system for drivers drowsiness detection

Dissertação de Mestrado
Mestrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor António Ribeiro

Outubro de 2022

Acknowledgment

This dissertation describes the performed work during 2021/2022. This is the final step of the Master's course, which encompasses all the academic experience acquired at this institution. It represents all the years of my academic education, filled with so many good memories. It is therefore important to thank all those who helped me to grow, not only in academic terms, but as a person, along this path.

I would like to thank my professor, Fernando Ribeiro, for the opportunity he gave me to develop this work in the Laboratory of Automation and Robotics. For his availability, for every feedback and lesson he gave.

I would also like to thank the Laboratory of Automation and Robotics for all the great environment, knowledge shared and incredible conditions to work.

A thanks to Paulo Garcia and Bruno Martins from Novatronica, for their availability, all the ideas shared and the material made available in order to carry out this work.

To all of my friends, from childhood and those I made in this academy and stayed close, for always being by my side in good and bad moments.

And last but not least, a huge thanks to my parents. It's because of them that I got to where I am, always pushed me to do my best and to reach my goals.

DECLARAÇÃO DE INTEGRIDADE

Resumo

Um sistema para deteção de sonolência nos condutores

A condução é uma atividade que requer concentração e um estado de atenção de alerta, pois em caso de distração ou sonolência pode levar a graves consequências. No entanto, o comportamento de conduzir num estado de fadiga, é subestimado por muitas pessoas. Além disso, para trabalhos como empresas transportadoras, os camionistas têm de levar a sua mercadoria ao destino o mais rápido possível e por vezes o descanso é evitado, levando a acidentes. Desta forma, torna-se pertinente o desenvolvimento de um sistema que consiga detetar sonolência nos condutores e que os alerte. Para a realização deste projeto foi então desenvolvido um sistema que deteta quando um condutor se encontra com sinais de sonolência, e para isso foram projetados dois modelos, utilizando visão por computador. O primeiro utiliza *object detection* para fazer um rastreio a regiões-chave como olhos, boca e cabeça. Para isto teve de ser construido um *dataset* e uma arquitetura de redes neuronais teve de ser escolhida, e o You Only Look Once (YOLO) foi a que se destacou e mais se adequou ao caso. O segundo modelo calcula os valores de *Percentage of Eyelid Closure (PERCLOS)* e *Frequency of Open Mouth (FOM)* de maneira a comparar com valores *threshold* para fazer um juízo final e dar o alerta. Com resultados bastante satisfatórios obtidos, testes foram efetuados numa Raspberry Pi, de forma a ver como o sistema construído se comportava num dispositivo com menos poder computacional.

Palavras-chave: Sonolência, visão por computador, *object detection*, YOLO, PERCLOS, FOM

Abstract

A system for drivers drowsiness detection

Driving is an activity that requires concentration and a state of alert attention, as in case of distraction or drowsiness it can lead to serious consequences. However, the behavior of driving in a fatigued state is underestimated by many people. Furthermore, for jobs such as transport companies, truck drivers have to deliver their goods to their destination as quickly as possible and sometimes rest is avoided, leading to accidents. Therefore, it becomes relevant to develop a system that can detect drowsiness in drivers and alert them. In order to carry out this project, a system was developed that detects when a driver has signs of drowsiness, and for that, two models were designed, using computer vision. The first uses object detection to track key regions like eyes, mouth and head. For this, a dataset had to be built and a neural network architecture had to be chosen, and YOLO was the one that stood out and best suited the case. The second model calculates the values of PERCLOS and FOM in order to compare with threshold values to make a final judgment and give the alert. With very satisfactory results obtained, tests were carried out on a Raspberry Pi, in order to see how the built system behaved on a device with less computing power.

Keywords: Drowsiness, computer vision, object detection, YOLO, PERCLOS, FOM

Contents

List of figures

List of tables

Glossary

CNN	Convolutional Neural Network.
ECG	Electrocardiography.
EEG	Electroencephalography.
Faster RCNN	Faster region-convolutional neural network.
FOM	Frequency of Open Mouth.
FPS	Frames per second.
HOG	Histogram of Oriented Gradient.
KCF	Kernelized Correlation Filters.
LBP	Local Binary Pattern.
PERCLOS	Percentage of Eyelid Closure.
PVT	Psychomotor Vigilance Task.
RCNN	Region-Convolutional Neural Network.
SIFT	Scale-Invariant Feature Transform.
SSD	single shot multibox detector.
SVM	Support Vector Machine.
YOLO	You Only Look Once.

1 Introduction

This chapter intends to present the theme of this project. Some of the causes and consequences of driving while drowsy are discussed, the objectives for carrying out this work and the structure of this dissertation are presented.

1.1 Background and Motivation

The set of factors that condition road driving and, consequently, road fatigue is one of the most common, due to the fact that fatigue reduces the ability to pay attention, which can lead to risky maneuvers and, in extremes, to the death of the driver and/or third parties. Many people are unaware of how dangerous the practice of driving under the influence of fatigue is. Others have this notion, but find it difficult to recognize the signs of tiredness in a timely manner.

Fatigue is caused by a number of factors such as long hours behind the wheel, activities performed before driving, sleep duration and quality, and the time of day when driving. This compromises cognitive and motor functions, which, when driving, leads to increased reaction times, decreased concentration on the task of driving, leads to poorer psychomotor coordination and less efficient information processing. This condition compromises the driver's ability to control their vehicle and increases the risk of being involved in an accident. Consequently, many drivers try to compensate for the influence of fatigue by increasing the demand on the task of driving (such as accelerating even more), which is a mistake (PRP, 2022).

A study carried out by the European Sleep Research Society aimed to estimate the prevalence, determining factors and consequences of falling asleep at the wheel. 12,434 surveys were conducted in 19 countries using an anonymous online survey, which collected demographic and sleep-related data, driving behavior, history of drowsy driving, and accidents. The average prevalence of falling asleep while driving in the last 2 years (on the date of publication of the article) was 17%. Among those who responded that they fell asleep, the mean prevalence of sleep-related accidents was 7.0% (13.2% involved hospital care and 3.6% caused death). Sleep quality, sleep the night before (42.5%) and poor sleep habits in general (34.1%) were the most frequent factors for falling asleep at the wheel. (Gonçalves et al., 2015).

Figure 1 (data taken from (Gonçalves et al., 2015)) shows the prevalence (%), by country, of drivers who reported a history of falling asleep at the wheel in the last two years (on the date of publication of the article). Note for Portugal, which stands out, but negatively, as one of the countries with the highest percentage of falling asleep while driving.

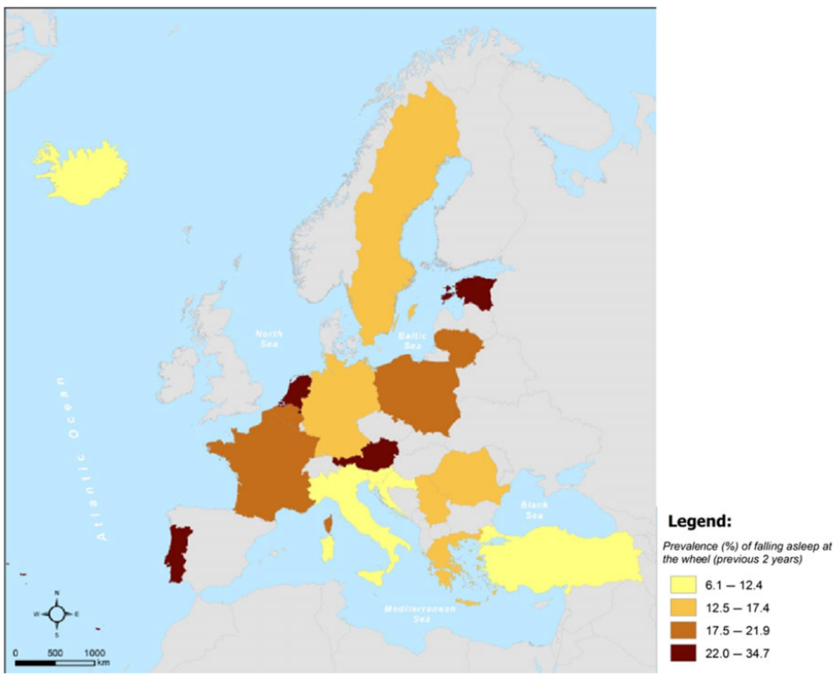


Figure 1: Prevalence (%) of drivers reporting a history of having fallen asleep at the wheel during the previous two years (of the published article) (Gonçalves et al., 2015)

Fatigue is also one of the causes of work accidents in the transportation sector. Truck drivers must have some characteristics like high alertness when driving, have vision in all directions and far ahead, have the ability to make quick and accurate decisions, have expertise and skills as a learning process and driving experience. However, this does not prevent drivers from having accidents when feeling tired after long hours of travel (Sulaeman & Fauzi, 2022).

Nowadays, there are safety systems which detect drowsiness, but these are not widespread and are uncommon among drivers because they are usually available in luxury vehicles. For example, Volvo developed the Driver Alert Control, which warns drivers suspected of drowsy driving by using a vehicle-mounted camera connected to its lane departure warning system. Another example is the Attention Assist System that has been developed and introduced by Mercedes-Benz that collects data drawn from a driver's driving patterns incessantly ascertains if the obtained information correlates with the steering movement and the driving circumstance at hand (Jabbar et al., 2018).

1.2 Goals

The main objective of this project is to develop a system capable of detecting signals of drowsiness on drivers. For that, the next steps had to be carried out:

- Study on machine learning - This project is all about computer vision and artificial intelligence. There are many ways to detect drowsiness, some of them use the color of the images, others use the image's features. For that, machine learning is more promising and can be way more accurate and in this order, it can become more safe for solving a problem like this. Because of this, a better understanding of machine learning approaches is required.

- Analyzes carried out by other authors - For a better understanding of how to accomplish the main goal of the project, it is important to see if there are works already carried out by other authors and if there are, to see and analyze how they made it.
- Dataset - It is needed a dataset for having a model trained, and in this way it will be created a dataset with images of people simulating signs of fatigue. It will be needed labeling all the images.
- Training the model - For this, it will be used a neural network and the dataset created. During this step, some arguments will be changed in order to reach the best results.
- Tests - Tests will be made for a later discussion and see if the results are adequate for solving the problem.
- Deployment on a board to see how it handles with the system designed.

1.3 Dissertation Structure

This dissertation is organized in the following order:

- Introduction - A brief introduction to the theme is made, and it shows some studies carried out about drowsiness and road accidents
- State of the art - In order the work to be carried out achieves good results, it is necessary to study the tools to be used, as well as the theory related to the topic. Thus, the research carried out will be presented in this chapter.
- Development - All work done after most of the majority of the research has been done it is shown in this chapter, step by step.
- Conclusion - A conclusion is made after all the work done since the beginning of this project.

2 State of the art

This chapter presents the state of the art. The research carried out was based on works by other authors related to the theme of this project, mainly in deep learning methods. A brief review of other methods and techniques related to computer vision to detect drowsiness, like eye-tracking, were also made.

For a better understanding of the work done by other authors, a brief explanation is given on concepts related to deep learning and YOLO, the framework used to train the object detection model. Aside from doing researches mainly about deep learning methods and others, a review of non-computer vision methods was also made.

2.1 Eye-Tracking

Human eye behavior also provides significant information about a driver's alertness, and therefore eye tracking is also one of the best techniques to monitor a driver's alertness.

Detecting fatigue and distraction in a driver by eye tracking is a complex task as it is a combination of computer vision expertise (at least one camera required), human factors, automotive systems, mechanics, optics, electronics and software. Also, ensuring that the system works in automotive conditions is not an easy challenge as there are certain barriers to overcome such as lighting, temperature, camera type, camera position and driver head position (Edenborough et al., n.d.).

Within this method, there are several ways to track the eyes and detect drowsiness, such as analyzing the driver's gaze direction, blink frequency and eyelid opening.

Blinking is a natural phenomenon, simply observable and easily accessible, which reflects the influence of central nervous system activation without voluntary manipulation and, therefore, is considered an adequate eye indicator for the detection of fatigue. The rapid closing and reopening of the eyelids occur spontaneously, that is, no external stimulus is required. The analysis of the phenomenon of eye blinking can provide essential information about the activation and fatigue processes of the central nervous system. Blink frequency and duration parameters are especially subject to characteristic changes with increasing drowsiness or fatigue. Figure 2 (Caffier, Erdmann, & Ullsperger, 2003) shows the waveform of the parameters of the closing duration, the period when the eyes are closed and the reopening.

The duration of the eye blink starts when the eye begins to close until the moment it returns to the open state. This parameter and the eye closing time are parameters that present different results for each alert state.

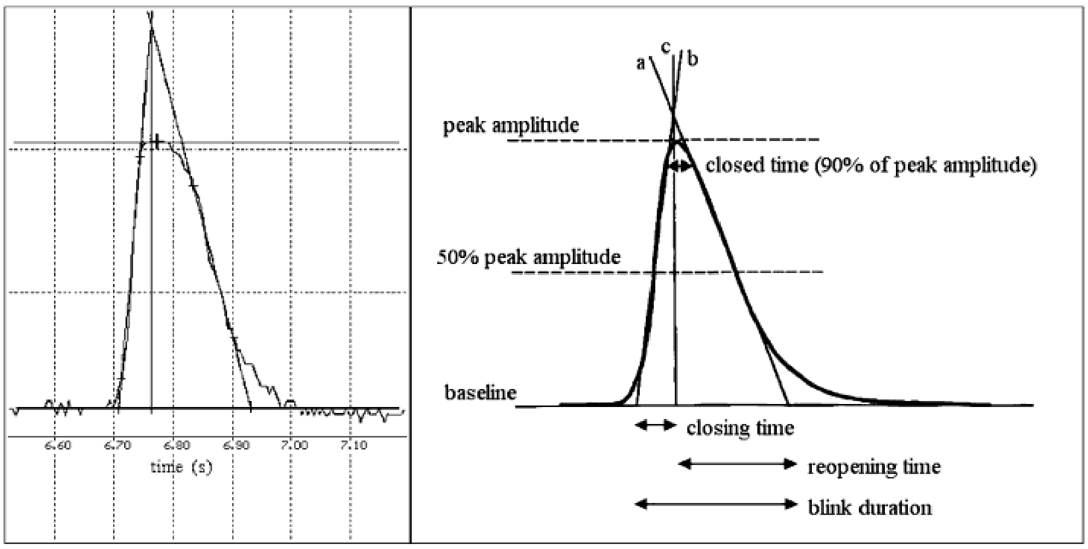


Figure 2: Example of a recorded blink signal (left) and schematic indication of parameters (right)(Caffier, Erdmann, Ullsperger, 2003)

2.1.1 PERCLOS

Of the several existing fatigue detection methods, PERCLOS is one of the most reliable. It is a non-invasive method and also consists of eye tracking. This describes the percentage of eyelid closure over the pupil over time and reflects slow eyelid closure rather than blinking. It was established in a 1994 driving simulator study as the proportion of time in a minute when the eyes are at least 80 percent closed (Dinges & Grace, 1998).

A study carried out by the FHWA (Federal Highway Administration) and NHTSA (National Highway Traffic Safety Administration) came to the conclusion that, among various fatigue detection methods, such as Electroencephalography (EEG), head position and blink frequency, PERCLOS was the one that presented the best results. This study consisted of monitoring the alertness of 14 adult men for a period of 42 hours, while working on a battery of computerized tests every 2 hours. The tests included a 20-minute Psychomotor Vigilance Task (PVT), which required each subject to maintain attention and respond to a light that appeared randomly on the computer screen by pressing a button. PVT performance lapses are the times when a subject did not respond to the task in a timely manner (i.e., less than 500 ms) (Dinges & Grace, 1998).

Six technologies were used and tested: two electroencephalographic algorithms (EEG), two blink detection monitors, which measure fatigue based on eye blink activity, a head position monitoring device, which detects fatigue based on head movement and PERCLOS. The latter had three fatigue metrics: P70, the proportion of time the eyes were closed by at least 70%, P80, the proportion of time the eyes were closed by at least 80%, and EYEMEAS (EM), the mean square percentage of the eyelid closure rating. In this way, the study design allowed estimating the degree of similarity, or coherence, between PVT performance failures and alertness. Table 1 (Dinges & Grace, 1998) shows the results obtained for each individual and through these data it can be concluded that of all the technologies tested, PERCLOS was the one that presented the greatest coherence.

Table 1 Correlation between the total number of visual performance failures in a 20-minute PVT test and the drowsiness detection results of each technology (Dinges Grace, 1998)

	PERCLOS			**EEG Algorithms**		**Head-position metrics**		**Eye-blink monitors**	
Subject	**P70**	**P80**	**EM**	**1**	**2**	**1**	**2**	**1**	**2**
1	0.89	0.92	0.89	*	*	0.83	0.82	0.10	*
2	0.85	0.83	0.84	*	0.40	*	*	0.71	0.20
3	0.95	0.97	0.95	*	*	0.91	0.85	0.90	*
4	0.84	0.83	0.83	*	*	-0.54	0.20	0.54	0.77
5	0.94	0.94	0.95	0.54	*	*	*	-0.10	*
6	0.95	0.96	0.94	0.57	0.95	*	*	0.39	0.54
7	0.92	0.92	0.92	0.36	0.31	*	*	0.54	0.32
8	0.55	0.67	0.70	0.66	0.84	*	*	0.50	0.85
9	0.78	0.77	0.71	*	*	0.23	0.13	0.67	*
10	0.95	0.97	0.95	*	*	0.87	0.65	-0.48	*
11	*	*	*	*	*	*	*	0.31	0.79
12	*	*	*	*	*	*	*	0.34	*
13	*	*	*	*	*	*	*	0.14	*
14	*	*	*	*	*	*	*	0.17	*

Detection methods can be divided into two categories, one based on features or colors such as skin color or facial geometry and another based on images. Detection methods based on facial features use some knowledge about human faces, such as face shape, location of eyes, nose, and mouth, to determine whether features seen in an image meet these criteria (Horng, Chen, Chang, & Fan, 2004). On the other hand, with image-based methods, facial pattern detection depends on samples, statistical analysis results and machine learning (Chehrehgosha & Emadi, 2016).

2.2 Color and features based detection method

Tracking the face through color has certain advantages in that color processing is much faster than processing other face features, and under certain lighting conditions color is invariant in orientation. However, it also has its downsides. The color representation of a face is influenced by many factors such as ambient light, object movement, camera characteristics and human skin colors vary from person to person (Yang & Waibel, 2002).

2.2.1 An approach using intensity changes on the face

In (Devi & Bajaj, 2008) the authors followed a color-based approach to face detection where a camera is used to acquire a video file and converted it to frames. In this work, the first step starts with face detection algorithm and is performed in three steps. The first step is to classify each pixel in the given image as a skin pixel or a non-skin pixel. The second step is to identify different skin regions in the skin-detected image by using connectivity analysis. The last step is to decide whether each of the skin regions identified is a face or not. Figure 3 (Devi & Bajaj, 2008) shows the results of the first step, the face detection.

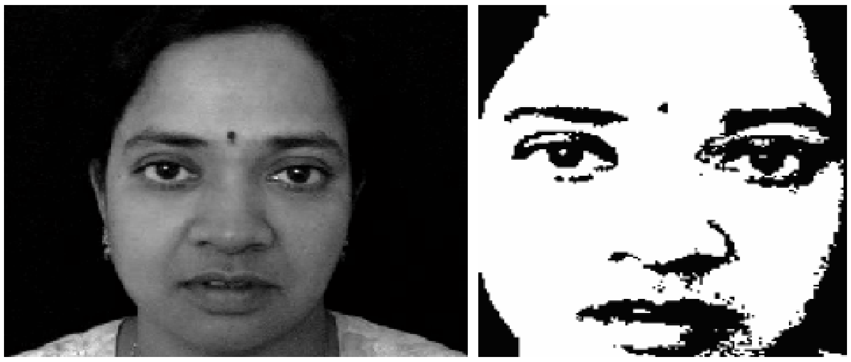


Figure 3: Face detection using a color-based method (Devi Bajaj, 2008)

The color based method is then used to detect the face in the image and after this step the lower part of the face is removed as the eyes are on the upper part of the face and thus the working area is reduced. After these steps, they made the eye detection algorithm where it was used an approach of finding the intensity changes on the face. The center of the face is used as a reference to compare the left and right eyes and is found using the boundaries of the face. The horizontal averages (average intensity value for each x-coordinate) of the face area are calculated, from the top of the face to the bottom, and this is done using the grayscale image. These values are obtained for each eye separately. Like figure 4 (Devi & Bajaj, 2008) shows, the given plot of this calculation shows the intensity variations and two significant intensity changes can be observed, where the first is the eyebrow and the second is the upper edge of the eye. The distance between the two changes in intensity determines the state of the eyes (open or closed). When the eyes are closed, the distance between the x – coordinates of the intensity changes is larger compared to when the eyes are open (Devi & Bajaj, 2008).

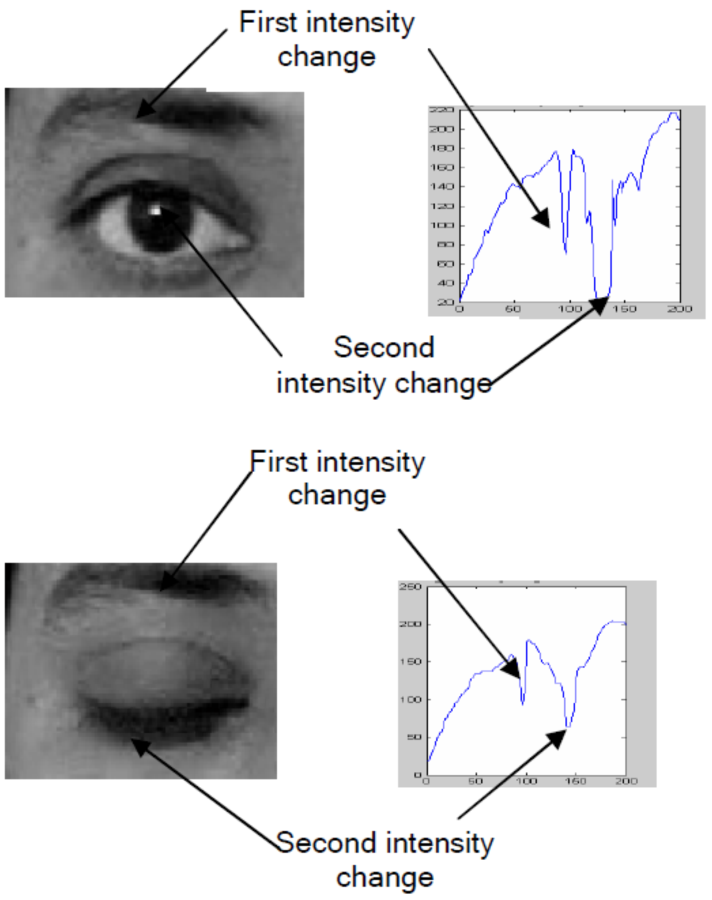


Figure 4: Average intensity variation on the face when eyes are open and close, where the x-axis represents the x-coordinates and the y-axis represents the average intensity (Devi Bajaj, 2008)

For fatigue detection, the eye regions are observed and if the eyes are closed for 5 frames in a row, the system concludes that the driver is in a short micro-sleep state and emits a (Devi & Bajaj, 2008) warning signal.

2.2.2 Viola and Jones method for object detection

In (Viola & Jones, 2004) the authors developed a face detection system that uses a set of features that, in a way, is based on the basic functions of Haars. It has three contributions: the first is the integral image, which allows the features used by their detector to be computed very quickly. The second is a simple and efficient classifier which is built using the AdaBoost learning algorithm to select a small number of critical visual features from a very large set of potential features. The third contribution is a method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions. Furthermore, this system has the advantage of running much faster than a pixel-based system.

So for the first contribution to this framework, the authors use a set of features which are reminiscent of Haar

Basis functions, more precisely, three kinds of features. The value of a feature of two rectangles is the difference between the sum of pixels within two rectangular regions. The regions have the same size, shape, and are horizontally or vertically adjacent. A three-rectangle feature calculates the sum inside two outer rectangles, subtracted from the sum on a central rectangle, and a four-rectangle feature calculates the difference between diagonal pairs of rectangles, as shown in figure 5, which presents examples of characteristics that are used in the identification of objects. The sum of the pixels that are inside the white rectangles is subtracted by the sum of the pixels of the gray rectangles. In (A) and (B) are the characteristics of two rectangles, in (C) the characteristics of three rectangles and in (D) the characteristics of four rectangles (Viola & Jones, 2004).

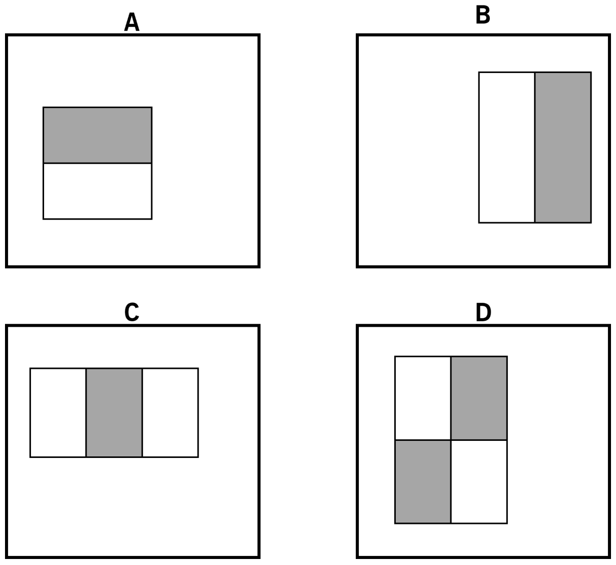


Figure 5: Examples of features that are used in object identification (Viola Jones, 2004)

The integral image located at x, y contains the sum of the pixels above and to the left of x, y (Viola & Jones, 2004):

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{2.1}$$

where *ii(x, y)* is the integral image and *i(x, y)* is the original image. Using the following equations:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{2.2}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{2.3}$$

the integral image can be computed at once over the original image (where s(x, y) is the cumulative sum of the rows, s(x,-1) = 0 e ii(-1,y) = 0).

A variant of AdaBoost is then used to select the best features and speed up the image classifier's performance. Adaboost is a machine learning algorithm that creates a stronger ranking algorithm using the output of weighted ranking algorithms known as weak learners.

A weak learner is defined in the sense that it can only be better than random, i.e., it is closer to the correct orientation. Strong learners refer to a set of classification algorithms that are highly correlated with correct orientation. In this way, poor learners can change to minimize errors in ungraded subjects. Thus, many of the images that do not belong to human faces will be eliminated and more calculations will be done on the frames most likely to be on the face. Unlike other methods, Adaboost chooses features that provide the most predictability (Chehrehgosha & Emadi, 2016).

This learning algorithm was used to select the single rectangle feature that best separates positive and negative examples. For each feature, the algorithm determines the optimal threshold classification function, so that the minimum number of examples is classified incorrectly (Viola & Jones, 2004).

The final contribution to this framework is a cascade of classifiers which achieves increases detection performance while radically reducing computation time. Stages in the cascade are constructed by training classifiers using AdaBoost (Viola & Jones, 2004).

Figure 6, recreated from (Viola & Jones, 2004), shows the face detection task. The initial rectangular features selected by AdaBoost are meaningful and easily interpreted. On the top row are the features, and on the bottom row the features are superimposed on a typical training face. The first feature selected seems to focus on the property that the eye region is many times darker than the nose and cheek region. This feature is relatively large compared to the detection subwindow and should be somewhat insensitive to face size and location. The second characteristic selected depends on the property that the eyes are darker than the bridge of the nose. The blur on the image was made on purpose.

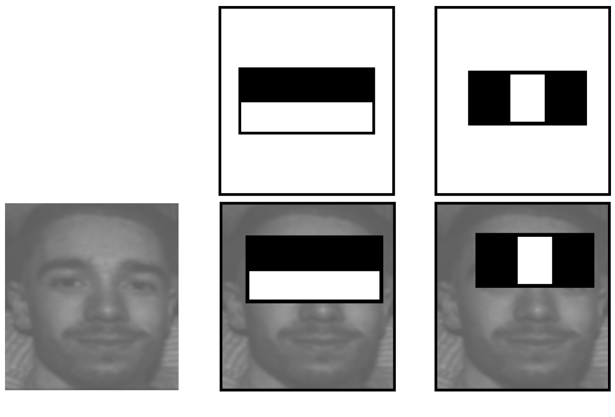


Figure 6: Face detection using a set of features

2.2.3 Drowsiness detection system using the Viola and Jones method

With techniques like PERCLOS, Local Binary Pattern (LBP), and Viola and Jones method, it is possible to create a system for detecting and monitoring the alertness of the driver. LBP stands for local binary pattern and is used in image retrieval, remote assessment, medical image analysis, facial image analysis, motion analysis and environment modeling. For each pixel in the image, the binary code establishing a threshold of the pixel value relative to the center pixel value is determined and a histogram is created to collect the various binary patterns (Chehrehgosha & Emadi, 2016). Figure 7 (of Electrical, Section, de Control Automático, of Electrical, & Engineers, n.d.) shows how it is calculated the binary relationship between each pixel of the image and its neighboring points in gray scale.

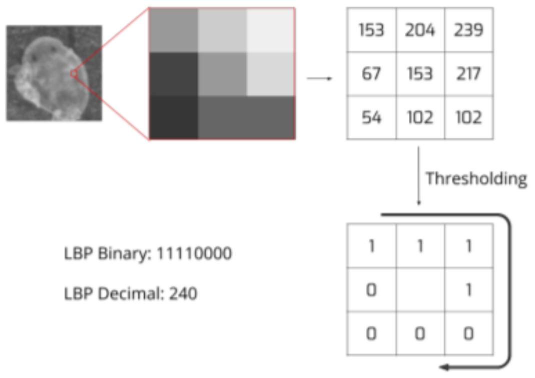


Figure 7: A better understanding of LBP

In (Manu, 2017), to perform the drowsiness detection task, the author used the Viola and Jones method for face detection as a first step to perform eye tracking and yawn detection in the following phases.

Once the face is detected, the image is converted to YCbCr domain, in order to perform skin segmentation. Converting the image to the YCbCr domain has the advantage that, the influence of luminosity can be eliminated by considering only the chromatic components. In the RGB domain, each component of the image i.e., red, green and blue has a different brightness. But, in the YCbCr domain all the brightness information is given by the Y-component, since the Cb (blue) and Cr (red) components are completely independent of the luminosity. It was found that the color remains distributed over a very tiny region in the chrominance plane, although skin colors change from person to person, and race to race. This method detects skin regions over the entire face image and rejects most of the non face image. As it can be seen in figure 8 (Manu, 2017), this method detects skin regions across the entire face image and rejects most of the non-facial image.

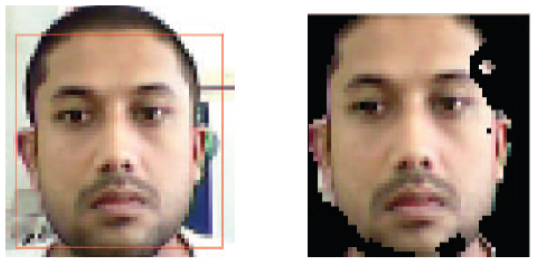


Figure 8: Face detection and corresponding skin regions (Manu, 2017)

After the driver's face is detected, comes the task of eye-tracking. The position of the driver's eyes are determined by using Viola Jones method as well. Then the two eyes are separated using edge detection and the center of the eye is determined in accordance with the symmetrical properties of the eye. Finally, the pupil is identified. If eyes are open, then it is concluded that the driver is in a normal state during which the alarm is not set off. If eyes are closed then it is concluded that the driver is in a fatigue state during which the alarm is set on (Manu, 2017).

The same approach is used to detect mouth area and once the regions of it are found, the mouth region alone is segmented by K means clustering and tracked using correlation coefficient template matching. K means partitions the objects into K number of mutually exclusive clusters, so that objects in each cluster are closest to each other, and farthest from objects in other clusters. Each of the K clusters is characterized by its center point. The function K-means performs K-Means clustering, by using an iterative algorithm that assigns objects to each clusters so that the sum of distances from each object to its corresponding cluster center point, over all K clusters, is a minimum. The objective function aims to obtain the minimum distance between the classes, or basically between the image pixels.

In a final step, a binary SVM classifier with a linear kernel is used for classification. This proposed system achieves an overall accuracy of 94.58% in four test cases.

2.2.4 Drowsiness detection based on PERCLOS and grayscale image

In (Yan, Kuo, Lin, & Liao, 2016) the authors designed a drowsiness detection system based on PERCLOS and grayscale image, and consists in three main parts. First, the system collects the necessary data to locate the driver's face region and eyes. Second, the system analyzes the captured data and collects new data from the images to set the fatigue model. Once the model is built, the system then starts to detect the driver's physical state, and if the driver shows signs of fatigue, an alarm sounds to warn the driver.

In the image preprocessing, the authors used the Sobel Operator for edge detection, since it can locate approximate edges in the shortest period of time. The edge detection is to find edge pixels (positions where the gray level suddenly changes) in a gray-scale image. The face region can be located through the histogram of edge pixels, because both sides of the face have the maximum amount of edge pixels (Yan et al., 2016).

Because for a human face, the edge pixels are concentrated in hair, eyes, nose, and mouth, they used those characteristics to establish a unique and distinct model from the other portions. Then it is applied a median filter and after these steps, eyes can be located (Yan et al., 2016).

After the eyes being located, it is applied the binarization in the regions of concern and in a final step comes PERCLOS. Once the amount of black pixels in an image is less than threshold (C + (O – C)*0.2, where C stands for close state), which means that in this image the eyes are under the P80 situation, it is defined as closed, otherwise, it means the eyes are open more than twenty per cent, and therefore defined as open. But every person has a unique frequency and speed when blinking, so it is computed the personal information of blinking to determine when the time proportion of eyelids are closed more than 80%, with the following expression (Yan et al., 2016):

$$TP = \frac{NC}{NC + NO} \tag{2.4}$$

where TP is the personal standard time proportion of P80, while NC is the number of images in the closed state, and NO is the number of images in the open state (Yang & Waibel, 2002).

2.3 Object detection and YOLO

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results (Works, 2022). The main purpose is to identify and locate one or more effective targets from still image or video data. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning (Cao, Chen, & Gao, 2020).

The object detection process is traditionally established by manually extracting feature models, where the common features are represented by Histogram of Oriented Gradient (HOG), Scale-Invariant Feature Transform (SIFT), Haar (Haar-like features) and other classic algorithms based on grayscale. Following feature extraction, the Support Vector Machine (SVM) or Adaboost algorithms are used for classification in order to obtain target information. These traditional extracting feature models have limitations in detecting multiple targets under complex scenes due to their poor generalization performance. They are only able to determine low-level feature information, such as contour information and texture information (Cao et al., 2020). Different from the traditional feature extraction methods, deep convolutional neural networks can achieve a high level of accuracy by extracting the features using multilayer convolution operations. In addition, they are robust in terms of geometric transformation, deformation, and illumination and can overcome the difficulties caused by environmental changes.

The deep learning methods can construct the feature description adaptively using the training data, and they are highly flexible and have high generalization ability. Among the deep learning methods, Region-Convolutional Neural Network (RCNN), Faster region-convolutional neural network (Faster RCNN), YOLO and single shot multibox detector (SSD) are the most widely used methods in object detection (Lu et al., 2019).

There are two approaches to do the task of object detection using deep learning:

- Create and train a custom object detector. To train a custom object detector from scratch, a design of the network architecture to learn the features of the objects of interest it is needed. Also, it must compile a very large set of labeled data to train the Convolutional Neural Network (CNN). The results of a custom object detector can be remarkable. That said, the layers and weights in the CNN are manually set up, which requires a lot of time and training data (Works, 2022).

- Use a pretrained object detector. Many object detection workflows using deep learning leverage transfer learning, an approach that enables to start with a pretrained network and then fine-tune it for custom application. This method can provide faster results because the object detectors have already been trained on thousands, or even millions of images (Works, 2022).

In addition to these approaches, it is necessary to choose the type of object detection network, a two-stage network or a single-stage network.

The initial stage of two-stage networks, such as RCNN and its variants, identifies region proposals, or subsets of the image that might contain an object. The second stage classifies the objects within the region proposals. Two-stage networks can achieve very accurate object detection results; however, they are typically slower than single-stage networks (Works, 2022).

In single-stage networks, such as YOLO, the CNN produces network predictions for regions across the entire image using anchor boxes, and the predictions are decoded to generate the final bounding boxes for the objects. Single-stage networks can be much faster than two-stage networks, but they may not reach the same level of accuracy, especially for scenes containing small objects (Works, 2022).

Table 2, data taken from (Cui et al., 2021), shows a brief comparison of different networks and their results, where it is possible to see the different values of precision, FLOPs and FPS for each network. From this, it is concluded that YOLOv4 is the most accurate network.

Table 2 Object detection model experiment comparison results (Cui et al., 2021)

Method	**Net**	**Precision**	**FLOPs**	**FPS**
Faster R-CNN	VGG16	96.61	121.32	-
SSD	VGG16	93.73	68.2	-
Slim-YOLOv3	SlimNet	90.93	31.60	9
Mini-YOLOv3	Mini-DarkNet	97.51	13.72	15
YOLOv3-tiny	TinyNet	88.51	5.21	21
YOLO-NANO	NanoNet	91.50	5.10	24
YOLO-LITE	LiteNet	89.67	4.92	26
YOLOv3	DarkNet-53	98.77	59.22	1
YOLOv4	CSPDarkNet53	99.57	98.22	1

Current detection systems, to detect an object, they take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image (Felzenszwalb, Girshick, McAllester, & Ramanan, 2010). Other approaches, like RCNN, use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene (Redmon, Divvala, Girshick, & Farhadi, 2015).

On the other hand, YOLO has a different approach for object detection. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. As it can be seen in figure 9 (Redmon et al., 2015), the system runs a single convolutional network simultaneously that predicts multiple bounding boxes and class probabilities for those boxes. It trains on full images and directly optimizes detection performance.

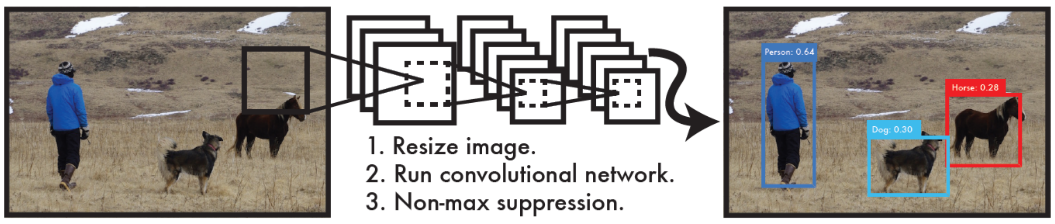


Figure 9: YOLO detection system

Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time, so it implicitly encodes contextual information about classes as well as their appearance (Redmon et al., 2015).

At this moment, YOLO has five versions, despite the last one being a little controversial. The main improvement measures of YOLO network are (Jiang, Ergu, Liu, Cai, & Ma, 2021):

- YOLO: The grid division is responsible for detection, confidence loss.
- YOLO V2: Anchor with K-means added, two-stage training, full convolutional network.
- YOLO V3: Multi-scale detection by using FPN.
- YOLO V4: SPP, MISH activation function, data enhancement Mosaic/Mixup, GIOU (Generalized Intersection over Union) loss function.
- YOLO V5: Flexible control of model size, application of Hardswish activation function, and data enhancement.

2.4 Deep learning for drowsiness detection

While color-based or feature-based methods are somewhat easy to implement, and a method like Viola and Jones is quite robust, on a more complex problem like detecting drowsiness in a driver they have several drawbacks, in addition to being methods of the past. Brightness is the biggest obstacle, as in color-based methods, the values of a color (RGB) can change according to lighting conditions and that can lead to false alarms. To solve these obstacles, methods based on deep learning can give better results.

2.4.1 Drowsiness detection using facial features

In (Deng & Wu, 2019) the authors proposed a system that was built using a commercial camera automobile device, a cloud server that processes video data, and a commercial cellphone that stores the result. So with this system, the automobile's camera captures the driver's portrait and uploads the video stream to the cloud server in real-time. Then, the cloud server analyzes the video and detects the driver's degree of drowsiness.

To monitor and warn the driver in real-time, first, it is made a face-tracking by using an algorithm called Multiple Convolutional Neural Networks Kernelized Correlation Filters (MC-KCF). This optimizes Kernelized Correlation Filters (KCF) algorithm and the combination of the CNN with it improves the performance of the latter in a complex environment, such as low light. The key regions of the driver's face (eyes and mouth) are recognized by using CNN. The system measures the angle of an opening eye to determine if the eye is closed and to detect yawning it assesses the duration of the mouth opening. Also, it measures the eye blinking frequency. In overall, this is all analyzed by a cloud server and in this stage, three main parts are analyzed: the driver's face tracking, facial key-region recognition, and driver's fatigue state. To finish, the cloud server transmits the result to the driver's cellphone and other apps, through which a warning tone is transmitted if the driver is observed to be drowsy (Deng & Wu, 2019).

2.4.2 Drowsiness detection using deep learning techniques

In (Jabbar et al., 2018) the authors developed a drowsiness detection system that can be implemented on Android applications with high accuracy. The first step was to extract videos from NTHU (National Tsing Hua University) Database. Then, extracted images from video frames. The third step consisted in extracting landmark coordination from images and Dlib library was used and it serves for estimating the location of 68 (x, y)-coordinates to map the facial structures of the face. Dlib is an open source SDK developed using C++ language to provide a Machine Learning algorithm used in many applications and in server domains such as robotics, cloud solutions, Internet of things and embedded systems. The following step was to train the algorithm, and the third step served as input to it. This was based on Multilayer Perceptron Classifier which is is a non-complex network of neurals which consists of intertwined nodes (neurons) that map out the output from the input class. The artificial neuron accepts one or more resembling dendrites (inputs), adds the accepted input based on connection weights, and thereafter produces an output class. An example of the architecture of a Multilayer Perceptron can be seen in figure 10.

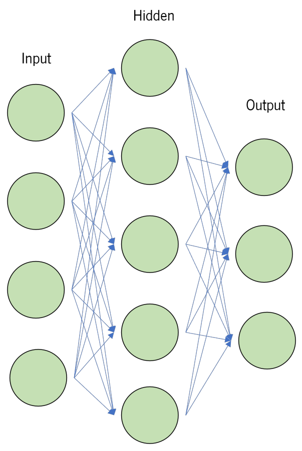


Figure 10: Example of a Multilayer Perceptron architecture

2.5 Other methods to detect drowsiness

Drowsiness can be detected by many ways, where some can be quite reliable and others less so. They can be intrusive and non-intrusive and until this point, the research carried out was all about non-intrusive, more precisely, about computer vision. But the intrusive ones can be reliable too. For example, the EEG (Electroencephalogram) is a diagnostic test that records the electrical activity of the brain. With the EEG it is possible to identify neurological changes, such as altered consciousness (fainting), epilepsy or detection of brain inflammation, and can be performed with the person awake or asleep, which is a non-invasive test without contraindications.

Electrodes are used, placed on the scalp, which are connected to a computer that records the brain's electrical activity. Traditional electrodes are Silver (Ag) or Silver Chloride (AgCl) with an electrolyte gel that is used between the electrode and the patient's scalp to increase signal impedance and improve accuracy. Electrodes must always be cleaned after use, which is time-consuming and inconvenient (Zhang & Eskandarian, 2021). However, there are capacitive electrodes that do not need direct contact with the skin, being non-intrusive and capable of measuring the biopotential (generated by the bioelectrical activity of organs such as the heart and brain, and conducted by neurons) outside the hair and thus avoiding the skin irritation. A method using capacitive electrodes is advantageous for long-term monitoring purposes as it places less mental or physical burden on (Sun & Yu, 2014) conductors.

For a driver alert state monitoring system, using the EEG method, the most convenient type of electrode to be used would be capacitive, as they do not use electrolytic gel and do not need to be cleaned each time after use, although Ag/AgCl electrodes have less noise and RMS (Root Mean Square) results for detecting brain activity are lower.

Due to the fact that the human brain works differently when he is awake, focused or sleepy, it is possible to use the EEG method to monitor the driver's alertness in order to analyze the different records of the brain's electrical activity for each individual state, and thus being able to reach a conclusion of the driver's alert state (whether sleepy or not).

Another way that can detect fatigue on drivers is the driver behavior. Driving behavior measures can be steering wheel motion, vehicle state information, road departure detection, and other accessible sensors, like the throttle pedal. Despite these techniques not being intrusive, they are subjected to several limitations as the vehicle type, driver experience, geometric characteristics and state of the road (Wang, Yang, Ren, & Zheng, 2006).

To summarize, driver's fatigue can be detected by many ways and the choice of these techniques can depend on the money to invest in the product, the reliability, and the intrusiveness and figure 11 shows, in a summarized way, the techniques that are possible to do it.

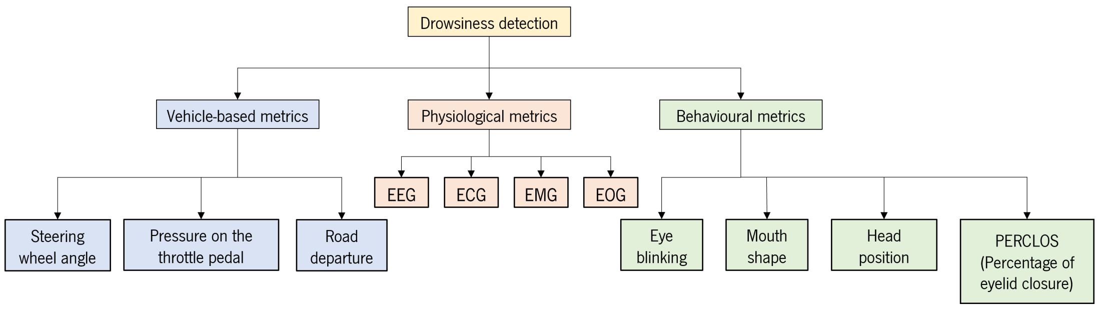


Figure 11: Techniques that can detect drowsiness

3 Development

This chapter focus on presenting the development of the project, in which it shows all the work done after the research carried out for the state of the art. Each step applied is explained in detail, in which the respective results are presented and a discussion about them is made.

3.1 Proposed solution

The task of monitoring the driver and detecting if he is in a state of fatigue or not can be done in many ways. It can be done by capturing real-time images using computer vision, by using physiological metrics like EEG or Electrocardiography (ECG), by using vehicle-based metrics like the steering wheel angle, position angle on the roadway or pressure on the throttle pedal and each way has its own advantages and disadvantages.

After conducting the research, it was concluded that an approach using computer vision to detect drowsiness can be more reliable and, moreover, is non-intrusive. However, inside this approach, there are many methods that can be used to reach the main goal. Drowsiness can be detected by the blinking frequency, PERCLOS, head position or FOM and this can be reached by using artificial intelligence or color based methods.

Deep learning methods can be more reliable, as color-based methods can lead to more false positives. Because of that, an approach of using computer vision with deep learning was chosen to solve the problem in question. But, to arrive at a proposal for a concrete solution, this project went through several phases. In an initial phase, after the research carried out for the state of the art, it dedicated itself to a greater knowledge about the tools to be used. At the beginning, drowsiness would be detected only by eye tracking and the calculation of PERCLOS. Then, mouth tracking and FOM calculation were added. However, drowsiness can be detected by using other metrics as well, and so, adding the head position tracking to the other two metrics can be more advantageous.

In this way, the final model consists of two parts. Like it is represented on the schematic of the proposed solution (fig. 12), the first part consists of an object detection model and the second part consists of a drowsiness detection model. The object detection model aims to track the eyes, mouth and head of the driver, which are captured by a camera in real-time. So each frame of the video is processed to see if the eyes and mouth are open or closed, and if the head is up or down. After this, the information about the eyes, mouth and head are processed by the drowsiness detection model. The PERCLOS and FOM values are calculated in a time window while the position of the head is verified in a certain amount of time. These values are then compared with a threshold defined in order to give the output, which is sober or drowsy.

The following sections explain in more detail about these two models.

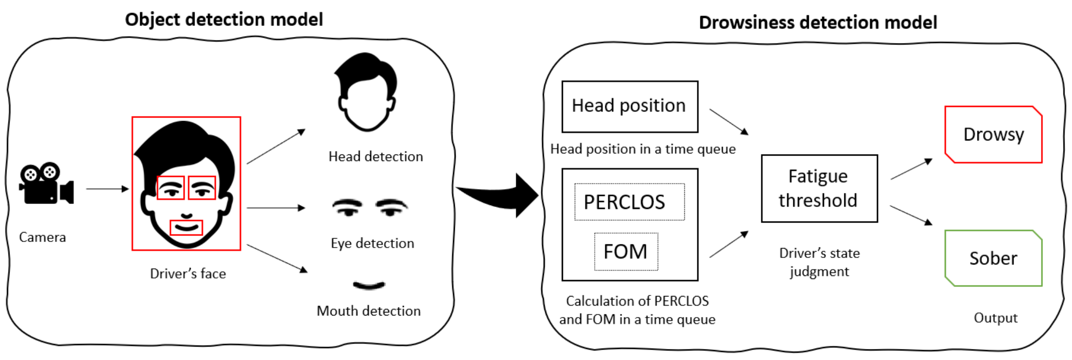


Figure 12: Schematic of the proposed solution

3.2 Object detection model

Computer vision is the approach chosen for the development of the system capable of detecting fatigue, however, as already mentioned, this area encompasses several methods to achieve this objective. So for the eye, mouth and head tracking it was opted using the object detection technique.

3.2.1 Choice of architecture to training the object detection model

Object detection is a computer vision technique that is widely used in tasks like face detection, vehicle counting, tracking objects and so on. As seen in the state of the art, object detection can be done by non-neural network approaches like the Viola and Jones method, SIFT or HOG and by neural network approaches like RCNN, Faster RCNN or YOLO.

One of the first steps of the development of this project was to choose the approach to do the object detection model. So to apply this technique, it was selected a neural network based approach, because it can give better results and non-neural approaches can be more time-consuming and are older methods. Then, as there are several working procedures within this approach to do the object detection task, one was chosen, after verifying the advantages and disadvantages of some of the best known. The points to be taken into account were:

- RCNN: Can achieve good results, however it has slow rate of training and high prediction time.
- Faster RCNN: Huge improvement of RCNN model. Performance speed is significantly higher. But this method has also some limitations, like the amount of time delay in the proposition of different objects. The speed can depend on the type of system being used.
- YOLO: This architecture currently has five major versions, and in general, the computation and processing speed of YOLO is high, in particular, in real-time compared to most of the training methods and object

detection algorithms. Also, YOLO can achieve high accuracy. The architecture of YOLO allows the model to learn and develop an understanding of numerous objects more efficiently. However, there are some failures to detect smaller objects in an image or video because of the lower recall rate.

- SSD: Is one of the fastest ways to achieve the real-time computation of object detection tasks, but can decrease the resolution of the images to a lower quality and typically perform worse than the Faster R-CNN for small-scale objects.

Making an overview of the points to consider of these main deep learning architectures for computer vision tasks, it is concluded that YOLO is the one that stands out the best. Detecting drowsiness is a real-time problem that requires, in addition to good accuracy, high speed detection, and that was the main reason for choosing YOLO, due to its high processing speed. YOLO has certain limitations regarding the detection of small objects, however it is not a problem for this case due to the position of the camera that will always be used in front of and close to the driver. The reason the camera is used in this way is that in the future a device will be built around this system, which will be placed in vehicles, for example, above the steering shaft. For a better understanding, an example of how the camera will be placed is show on figure 13.

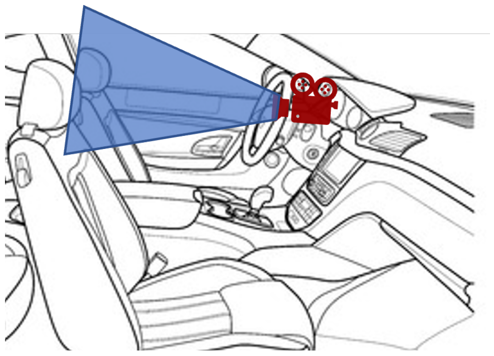


Figure 13: Placement of the camera in the vehicle in order to detect the key regions of the face's driver

As already mentioned, YOLO currently has five major versions (YOLOv1, YOLOv2, YOLOv3, YOLOv4 and YOLOv5), with the last two having the best results in terms of accuracy and detection speed. YOLOv4 is written in C and is based on the darknet neural network. However, YOLOv5 is written in python and uses the PyTorch framework. Usually, python is better for computer vision projects because it is a well-supported language in the area of machine learning, so this was the main reason to choose YOLOv5. Also, with this framework, it is easy to convert a trained model to other formats like TensorFlow Lite, ONNX and so on.

In a final step, because YOLOv5 contains 5 different models (see table 3, data taken from the official Ultralytics organization repository on GitHub (Ultralytics, 2021)), one had to be chosen. These models go from the smallest, capable of giving real-time Frames per second (FPS) on edge devices, to very large and accurate models meant for cloud GPU deployments. The smaller the model, the higher the detection speed is, but lower accuracy, and vice-versa, the larger the model, the lower the detection speed is, but higher the accuracy. Despite this, the largest and the smallest models do not differ in layers, but in the scaling multipliers of the width and depth of the network. Because one of the main requirements in the problem in question is detection speed, the two smaller models were used for various trainings and the results were compared. The performance of these trainings and the respective results will be presented in the following sections.

Table 3 Pretrained checkpoints of YOLOv5

Model	Size (pixels)	mAPval (0.5:0.95)	mAPval (0.5)	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.5	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

3.2.2 Dataset

Having a neural network architecture that presents good accuracy and speed results is not enough to achieve the intended results. It is necessary to have a good dataset to take full advantage of the neural network architecture. In this way, a dataset was built in order to train the object detection model.

Before a dataset was built, the MRL (Media Research Lab) Eye Dataset was used for knowledge and testing of the tools that were used. The MRL Eye Dataset is a large-scale dataset of human eye images. It contains infrared images in low and high resolution, all captured in various lightning conditions and by different devices. Data from 37 different people (33 men and 4 women) were collected, and has a total of 84,898 images. This dataset contains the images captured by three different sensors (Intel RealSense RS 300 sensor with 640 × 480 resolution, IDS Imaging sensor with 1280 × 1024 resolution, and Aptina sensor with 752 × 480 resolution). For testing, not all images were used, only 10,000. Figure 14 shows what some of the images that make up this dataset looks like (MRL, 2017).

(a) Closed eye (b) Open eye

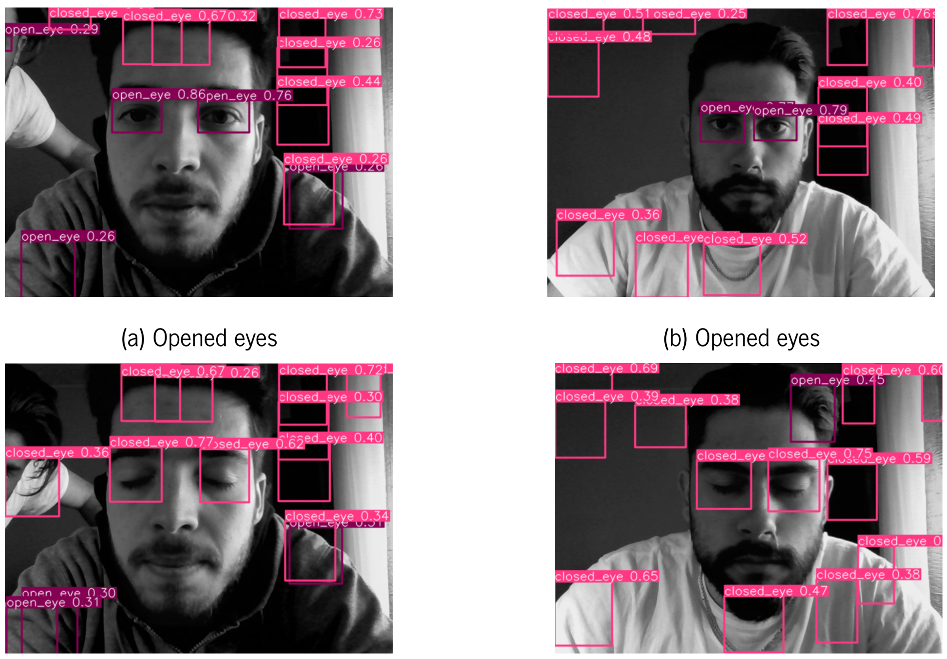
Figure 14: Images of a closed eye and an open eye of the MRL Eye Dataset

So after labeling the images, the model was trained, and the results can be seen in figure 15. It can be observed that the open and closed eyes are well detected, however there are a lot of false positives. One of the reasons of this happening it is because the dataset only contains images of eyes and the YOLO algorithm works with full images. It divides the image into n grids, each having an equal dimensional region of m × m and each of these n grids is responsible for the detection and localization of the object it contains, unlike other methods like sliding window. What this means is that if the dataset had images collected from people's faces, under different conditions, as the system will monitor in real time, and not just the eyes, the results would be better. Therefore, this dataset is not very suitable for this architecture. Other reason it is because the model was trained with a low value for the image size. This will be explained in the following subsection.

Like it was mentioned, the development went through some phases. The first phase was to get to know how this architecture works using the MRL Eye Dataset, only doing an eye tracking. So after seeing how YOLO works, it was time to create the dataset to train the object detection model. Some images were used from a dataset that served as training for a model for detecting the use of cellphones in drivers.

In the second stage, the dataset was created in order to do an eye and mouth tracking. Eight volunteers recorded videos of themselves inside a car and simulated signs of drowsiness. The videos were recorded in daytime and in nighttime. Also, four of these volunteers recorded videos in the same way, but in front of a white wall. The reason of doing this, it's because of not having a background with objects and in this order the number of false positives can be reduced, thus getting better results. The direction of the face was taken into account, thus having images with the face facing right, left and forward. Some volunteers recorded themselves using eyeglasses.

All videos were then split into frames giving a total of 6000 images. Then, 68% of the total of the images were converted to grayscale. This conversion is due to the fact of training the model, to be implemented in the future in a device that will have an IR camera incorporated and that will be built around this system, as already mentioned.


(a) Opened eyes (b) Opened eyes

(c) Closed eyes (d) Closed eyes

Figure 15: First results of the object detection model using the MRL Eye Dataset

IR cameras are more suitable to solve the fatigue detection problem as they can capture images at night or in low light environments.

For the third phase, the head position tracking was added and in this order some images had to be added to the dataset. But as the results were not good when detecting the head down, it was necessary to add even more images so that the model could detect the head position well. The final dataset turned out to have 7610 images.

So basically, throughout the development of the project, the dataset had to be restructured a few times. Figure 16 shows the general process that led to achieve the best results.

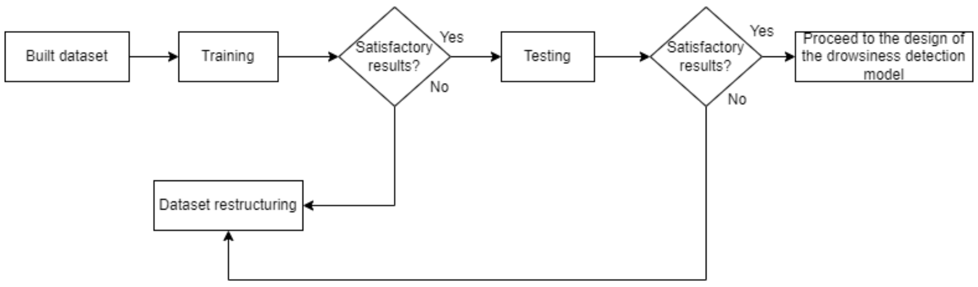


Figure 16: Flowchart of the optimization process that led to achieve the best results

Because the model needs some validation data to determine how good the inferences are during and after training, the set of images was split into train and validation datasets, with a split ratio of 75% for training and

25% for validation. All images were labeled manually using the tool labelImg, like as was done with the MRL Eye dataset. LabelImg is a free, open source tool for graphically labeling images. It's written in Python and uses QT for its graphical interface (see figure 17).

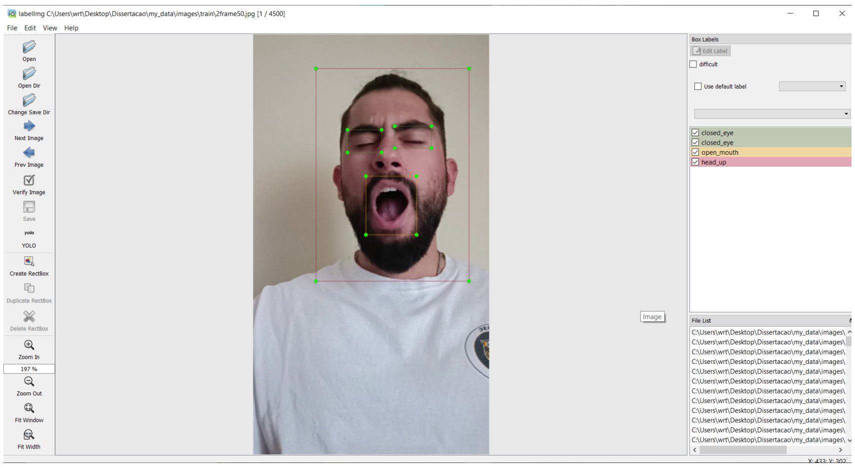
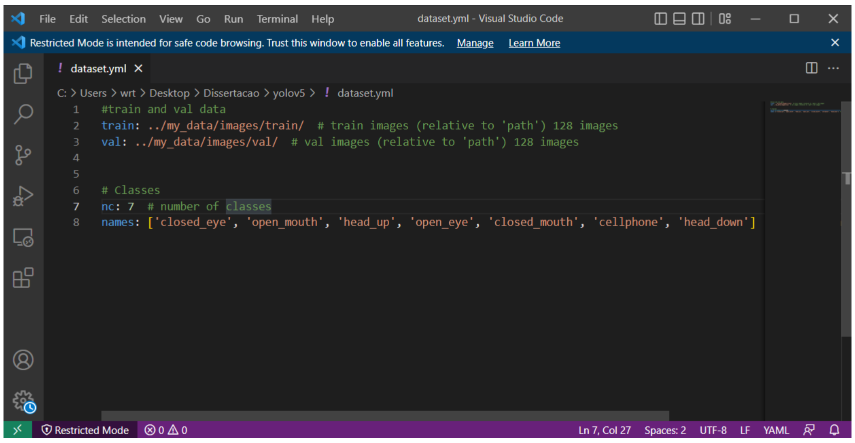


Figure 17: LabelImg tool

The final step to complete the dataset was to check if the file dataset.yml was correct. The dataset.yml file is part of the Yolov5 framework and tells the model how the dataset is distributed, how many classes there are, and what their names are. Like it is represented in Figure 18 the second line indicates the relative path for the train images set, and the third line indicates the relative path for the validation set. Line 7 states how many classes the dataset contains, and line 8 defines the names for each class.


File
Edit
Selection
View
Go
Run
Terminal
Help
dataset.yml - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features.
Manage
Learn More
dataset.yml
C: > Users > wrt > Desktop > Dissertacao > yolov5 > dataset.yml
Restricted Mode
Ln 7, Col 27
Spaces: 2
UTF-8
LF
YAML

```
1   #train and val data
2   train: ../my_data/images/train/  # train images (relative to 'path') 128 images
3   val: ../my_data/images/val/  # val images (relative to 'path') 128 images
4
5
6   # Classes
7   nc: 7  # number of classes
8   names: ['closed_eye', 'open_mouth', 'head_up', 'open_eye', 'closed_mouth', 'cellphone', 'head_down']
```

Figure 18: Dataset.yml file

3.2.3 Trainings and results

The next step after the dataset is built is to use YOLOv5 to train the object detection model. However, in this project, the dataset that was built underwent minor changes like it was mentioned. Trainings were carried out along these changes in order to obtain the best results, and they were made on a computer based on the Ubuntu 19.04 system, Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 16GB RAM. The GPU was a NVIDIA GeForce GTX 1650. Before presenting the results of the trainings performed, some concepts and parameters related to the framework that was used need to be explained.

Hyperparameters in machine learning control various aspects of training and the whole learning process of a deep learning model. YOLOv5 has about 30 hyperparameters used for various training settings. To do all the trainings, the default values of the hyperparameters were used, which are optimized for YOLOv5 COCO training from scratch. Some of the most important are:

- Learning-rate start (lr0): learning rate starts to determine step size at each iteration.
- Learning-rate end (lr1): used to stop or continue training by comparing it with the current learning rate. If the current learning-rate is higher or equal to the learning-rate end, the training stops, otherwise it continues.
- Momentum: is the tuning parameter for the gradient descent algorithm, its work is to replace the gradient with an aggregate of gradient.
- Mosaic: is used to increase model accuracy by creating a new image from a combination of multiple images, and then using newly created images for training. It is well known for data augmentation and finding optimal feature techniques.
- Degree: is used to increasing model accuracy by randomly rotating images up to 360 degrees in whole training data. This parameter helps most when you need to detect objects in multiple positions/angles.
- Scaling: is used to resize an image, either to match with grid size or for optimization of results.
- flipud: is used to flip images up and down randomly from the complete dataset for achieving better results. This can be considered in the data augmentation technique.
- fliplr: is used to flip images left and right randomly from the complete dataset for achieving better results. This can be considered in the data augmentation technique.
- weight-decay: a type of regularization technique, that works by adding a penalty term to the cost function of a network, which has the effect of shrinking/compressing the weights during the backpropagation process.

So after everything is configured correctly, the training is started, running a command from a terminal and using the following arguments that specify the training:

- img or imgsz (image size) in pixels (default — 640) – one value can be used (instead of 640×480, for example), as this defines the longest side.
- batch – the batch size defines the number of samples that will be propagated through the network (number of training examples utilized in one iteration). The higher the batch size, the more memory space it will be needed.
- epochs – defines the number of epochs. One epoch is when an entire dataset is passed forward and backward through the neural network only once. Corresponds to one cycle through the full training dataset
- data – path to the data-configurations file.
- Weights – path to initial weights. In this case, it was used yolov5s.
- cache - cache images for faster training.
- cfg - path to the model-configurations file.
- workers - denotes the number of processes that generate batches in parallel.

Finally, it remains to know which metrics are used to observe and draw conclusions about the trainings performed. The metrics used in YOLOv5 are as follows:

- Precision, also called positive predictive value, measures how much of the bounding box predictions are correct (how good it finds all the positives). The following equation shows how precision is calculated, where TP is True Positives (predicted positive / actual positive), TN is True Negatives (predicted negative/ actual negative), FP is False Positives (predicted positive/ actual negative), and FN is False Negatives (predicted negative/ actual positive):

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

- Recall, also called as sensitivity, measures how much of the true bounding box were correctly predicted. Figure 19 shows an illustration of sensitivity VS specificity for a better understanding of these two metrics (precision and recall). The following equation shows how recall is calculated:

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

- F1 score, measures the balance between precision and recall. To calculate it, the following equation is used:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.7}$$

- Average precision (AP) is a way to summarize the precision-recall curve into a single value representing the average of all precisions. represents the area under the curve (AUC) Precision Recall Curve. The higher the curve is in the upper right corner, the larger the area, so the higher the AP, and the better the model. Can be calculated by using the following expression:

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k) \tag{3.8}$$

- IoU (Intersection over Union) measures the overlap between two boundaries. It is used to measure how much the predicted boundary overlaps with the ground truth:

$$IoU = \frac{A \cap B}{A \cup B} \tag{3.9}$$

- mAP_0.5 is the mean Average Precision (mAP) at IoU (Intersection over Union) threshold of 0.5. The mAP is the mean of the APs for all classes and it is calculated by using the following expression (AP_k is the AP for the class k and n is the number of classes) :

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{3.10}$$

- mAP_0.5:0.95 is the average mAP over different IoU thresholds, ranging from 0.5 to 0.95.

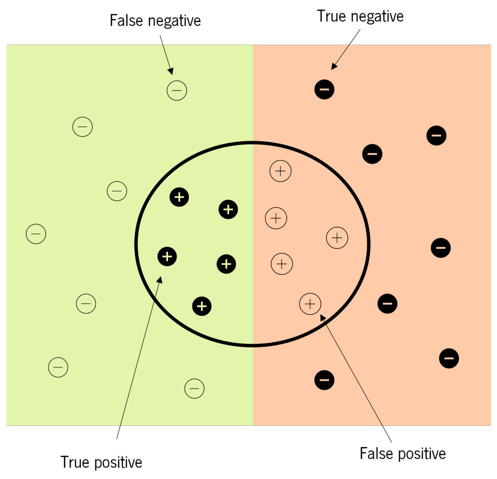


Figure 19: Illustration of sensitivity vs specificity

After explaining some of the concepts related to this framework, it remains to present the results obtained in each training performed. All the results that will be presented were obtained using the dataset that was built. To observe these results, TensorBoard was used in Google Colab.

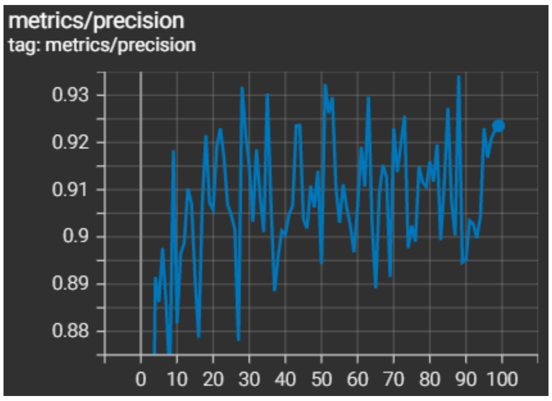
a) First training

The first training was done in order the model could track only the eyes and mouth, so at this time the dataset used had only labeled images for eyes and mouth (and cellphone). To start the training, the following command was used:

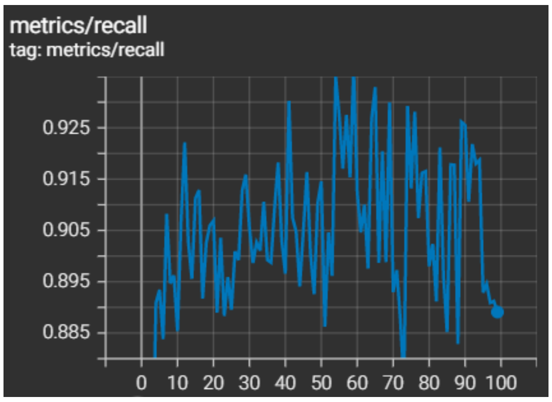
```
python train.py --img 640 --weight yolov5s.pt --batch 8 --epochs 100
--workers 2 --data dataset.yml
```

The reason it was used a batch of size 8 it is because a larger batch size would lead to poor generalization. Smaller batch sizes allow the model to start learning before having to see all the data. As for the size of the image, it was used the maximum that the computer that was used to train the model allowed. Usually, larger image sizes lead to better results, however take longer to process. Finally, it was established a number of 100 epochs to train the model.

For an object detection model, the higher the precision, the more confident the model is when it classifies a sample as positive and, the higher the recall, the more positive samples the model correctly classified as positive. As it can be seen in figure 21, precision reached values of 93% and 92% for the best value (at 88th epoch) and last value (at 100th epoch), respectively, and recall reached values of 94% (at 59th epoch) and 88% (at 100th epoch) for best value and last value.



(a) Precision (in the y-axis and the epochs in the x-axis)

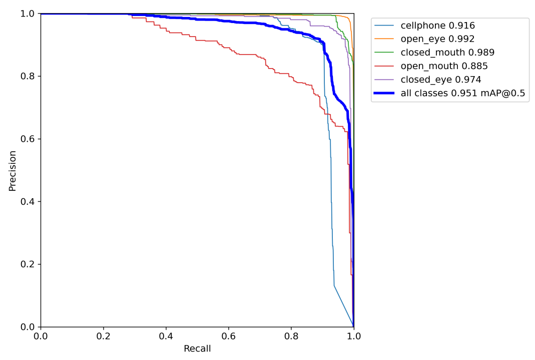


(b) Recall (in the y-axis and the epochs in the x-axis)

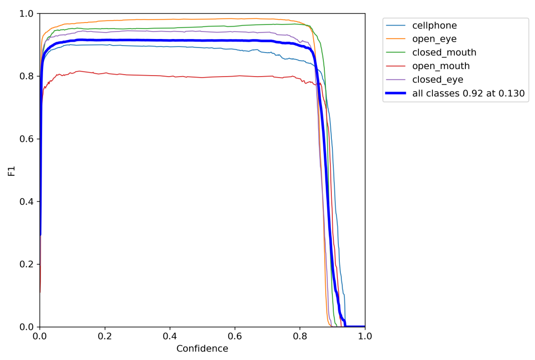
Figure 20: Results of precision and recall of the first training throughout each epoch

So both precision and recall reached high values, however, this information alone is not enough to draw conclusions from the model. A model that produces no false positives has a precision of 100%, but the model can have such precision even if there are undetected or not detected bounding boxes that should be detected. Because these two metrics are important, a precision-recall curve shows the tradeoff between the precision and recall values for different thresholds and its main objective is to select the best threshold to maximize both metrics. Figure 21a shows the precision-recall curve of this first trained model, and as it can be seen, as recall increases, precision decreases. This is due to the fact that as the number of positive samples increases (i.e., when recall is high), the accuracy of detecting each sample correctly decreases. This is expected, because when there are many samples, the model is more likely to fail.

Another way to select a threshold in order to maximize both metrics (precision and recall) is to use the F1 score. The higher its value, the better the model is. In figure 21b it can be seen that f1 reached a high value. Also, the confidence interval at which the F1 value remains high is large. In this way, it is possible to verify that this is a good model.



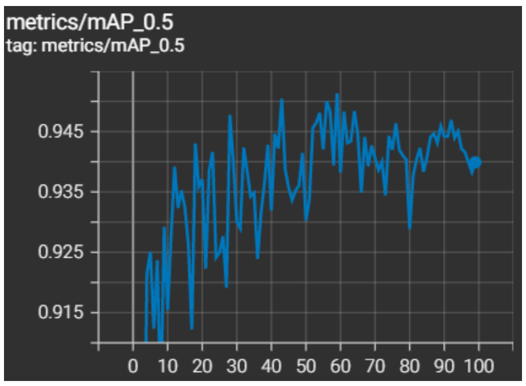
(a) Precision-Recall curve



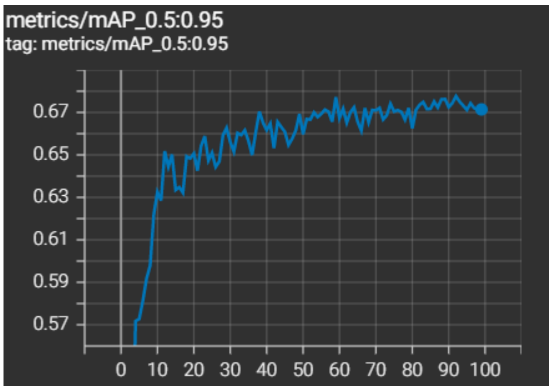
(b) F1 curve

Figure 21: Results of precision-recall curve and the F1 score of the first training

To confirm the success of this trained model, mAP is verified. So, in figure 22, the evolution of mAP_0.5 and mAP_0.5:0.95 over each epoch is shown. In summary, for these two metrics, the higher the score, the more accurate the model is in its detections. And as it can be seen, the best value achieved for mAP_0.5 was 0.95 (at 59th epoch) and the last value was 0.94 (at 100th epoch). As for mAP_0.5:0.95 the best one was 0.68 (at 92nd epoch) and the last one was 0.67. Thus, it is concluded that the model was successfully trained.



(a) mAP_0.5 (in the y-axis, epochs in the x-axis)



(b) mAP_0.5:0.95 (in the y-axis, epochs in the x-axis)

Figure 22: Mean Average precision (mAP) at IoU threshold of 0.5 and over different IoU thresholds, ranging from 0.5 to 0.95

Finally, to verify that the model corresponds to expectations, it was tested in real time for two cases, day and night. As it was tested using the computer camera, during the tests carried out at night it was necessary to have some light to capture the image with visible conditions. The results of the tests of the object detection model at daytime and nighttime can be seen in figure 23 and figure 24. The figures show the detection of open/closed eyes and open/closed mouth and as can be seen, the model is able to track the eyes and mouth as intended and with high confidence.

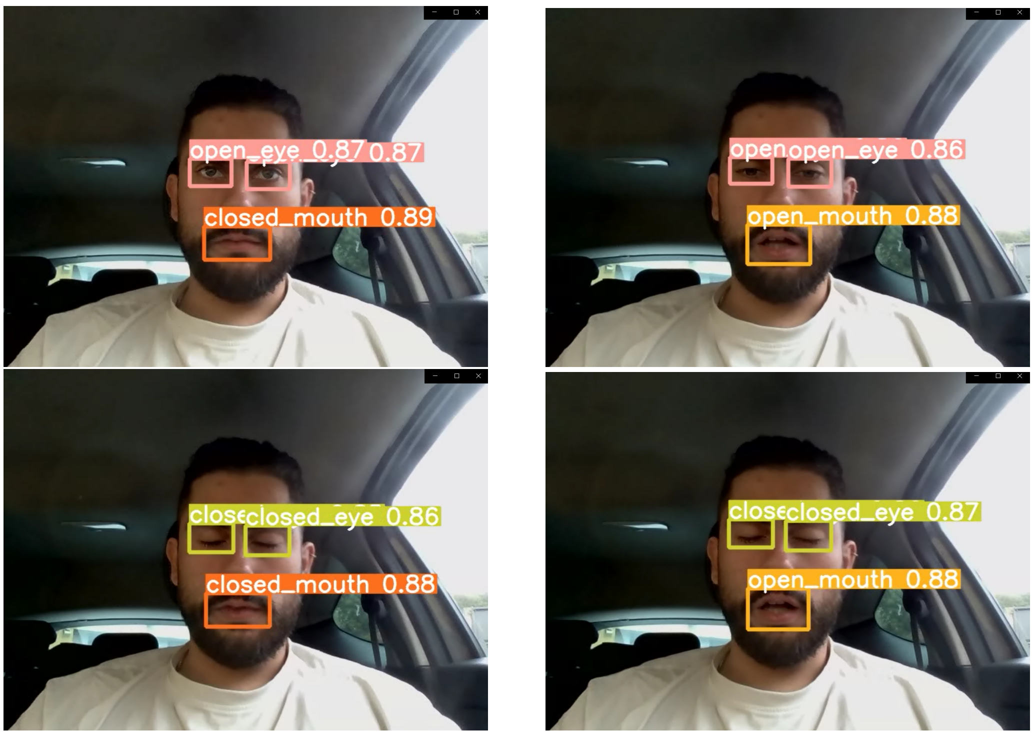


Figure 23: Test of the first trained model at day and nighttime

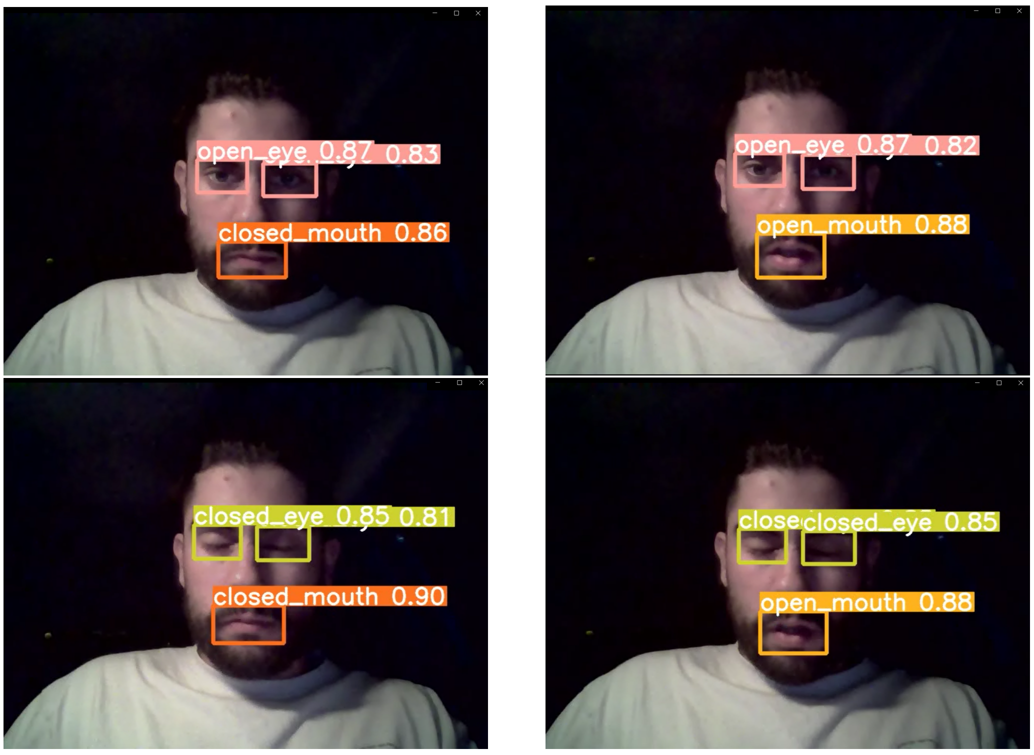


Figure 24: Test of the first trained model at day and nighttime

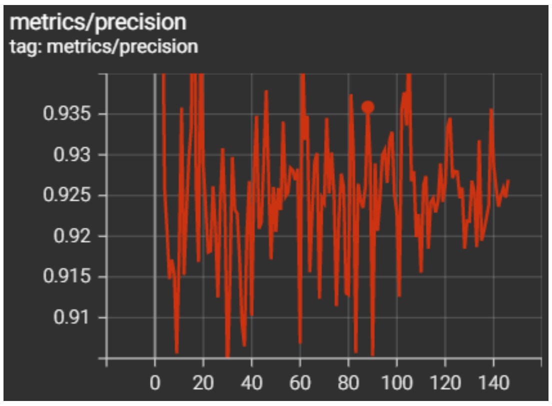
b) Second Training

The first model was successfully trained, however it was used a large image size and when it is being used to make detections in real time, the same image size should be used on which it was trained. Due to this fact, more power is required, which is not intended, because in the end a hardware will be built for this purpose and the less power the more profitable. Also, the model only detected eyes and mouth and adding the head position tracking is more complete for drowsiness detection. So for this second training, the dataset had to be restructured, and the resolution of the images was reduced to 75%. The following command was used:

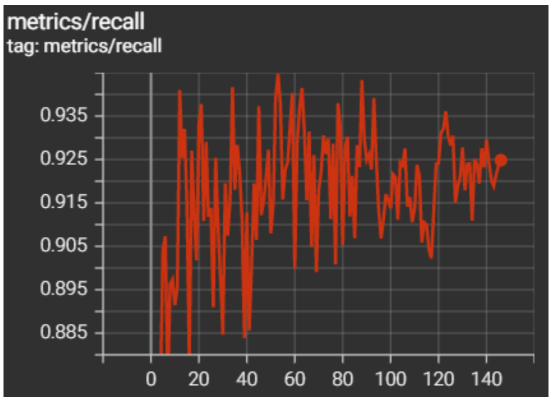
```
python train.py --img 320 --weight yolov5s.pt --batch 16 --epochs 200
--workers 2 --data dataset.yml
```

In this training, the argument of image size was reduced to 320 and the batch size was increased. Reducing the image size and increasing the batch size can lead to a faster training. Despite the number of epochs being greater, and set to 200 epochs, the training stopped at the 146th epoch because no improvement was observed from the 100th epoch.

Figure 25 shows the results of precision and recall throughout each epoch. As it can be seen, precision reached 96% for the best value (at 16th epoch) and 93% for the last value (at 146th epoch). As for recall, it reached 94% (at 53rd epoch) for the best value and 92% (at 146th epoch) for the last value. Comparing these values with those obtained in the first training, it appears that the precision increased and that the last recall value reached a higher value.



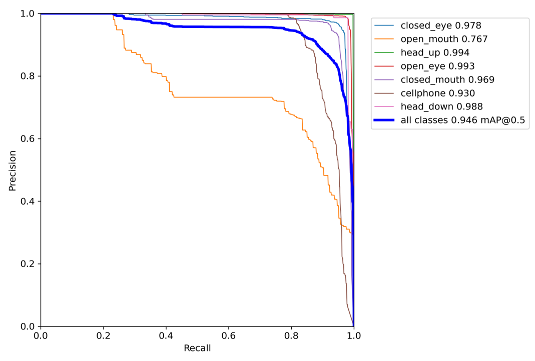
(a) Precision (in the y-axis and the epochs in the x-axis)



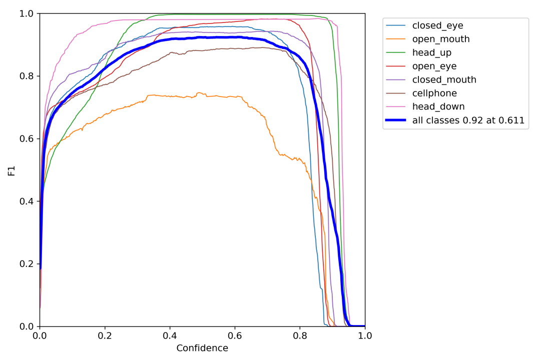
(b) Recall (in the y-axis and the epochs in the x-axis)

Figure 25: Results of precision and recall of the second training throughout each epoch

But like it was mentioned before, these two metrics are not enough to judge if the model will perform well or not, despite the results for both precision and recall being good. As it can be seen, the precision-recall curve, in figure 30, shows better results for open and closed eyes classes, compared to those obtained in the first workout, but for the open and closed mouth classes, the results are still better on the first training. As for the F1 curve, this time it shows a smaller confidence interval. This means that more true positives (TP) can be falsely classified as negatives (FN).



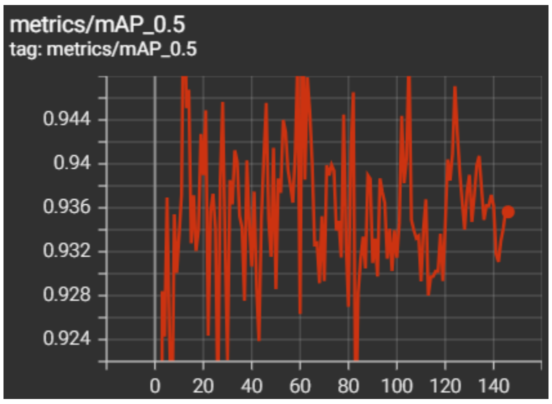
(a) Precision-Recall curve



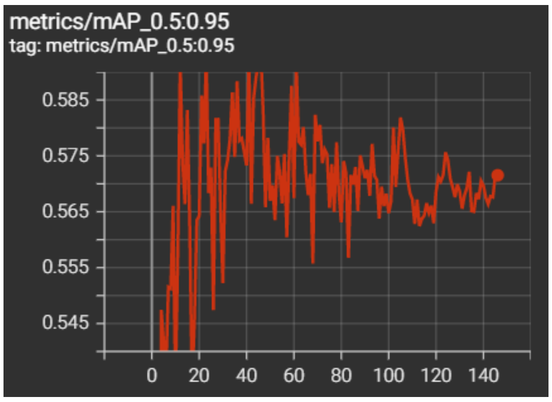
(b) F1 curve

Figure 26: Results of precision-recall curve and the F1 curve of the second training

So despite having worst results than the previous training, the mAP values obtained at IoU threshold of 0.5 in this second training were similar. Figure 27 shows the mAP_0.5 and mAP_0.5:0.95 values throughout each epoch. Both of these metrics show a greater oscillation, but the first shows that the best and last values were similar to those obtained in the first training. However, mAP_0.5:0.95 reached smaller values.



(a) mAP_0.5 (in the y-axis, epochs in the x-axis)



(b) mAP_0.5:0.95 (in the y-axis, epochs in the x-axis)

Figure 27: Mean Average precision (mAP) at IoU threshold of 0.5 and over different IoU thresholds, ranging from 0.5 to 0.95, of the second training

For a final step, to check if this model performs well or not, it was tested in the same conditions as the previous one was tested. And as expected, when the results were observed, the model falsely classified some true positives (TP) as false negatives (FN). In this order, another training had to be done. Figure 28 shows an example of that. The position of the head is down, and the model should detect that, as the mouth shape, however it marks as false negatives (FP).

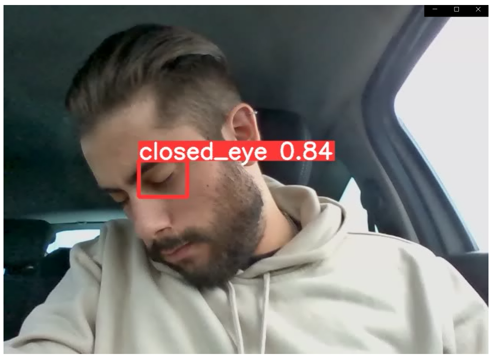


Figure 28: Example of how this model can fail at the task of detecting the key regions

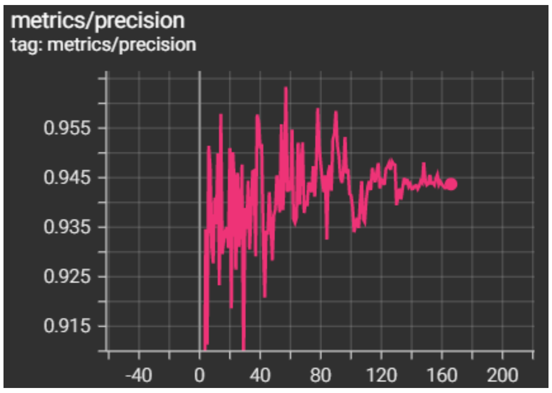
c) Third Training

For a problem like detecting drowsiness, the eyes, mouth and head tracking has to be done through an object detection model that not only has high detection speed but also high accuracy. Because the previous training was not successful, another one had to be done. So the dataset has been restructured once more, where more images of head down and open mouth were added. The following command was then used:

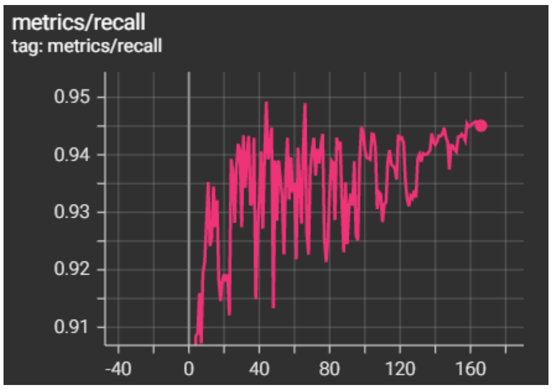
```
python train.py --img 320 --weight yolov5s.pt --batch 8 --epochs 200
--workers 2 --data dataset.yml
```

In this training, the batch size was reduced, but the other arguments were the same. The training stopped at the 166th epoch because no improvement was observed in the last 100 epochs.

The results of precision were very similar to the second training, however, the recall values in this training were better. The results can be seen in figure 29, and it shows that the best value for recall was achieved at the 44th epoch, reaching a value of 95% and the last values was 94%.



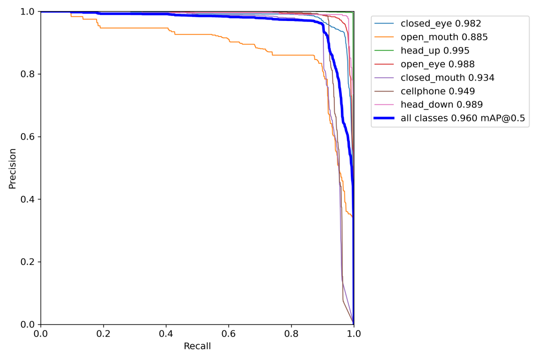
(a) Precision (in the y-axis and the epochs in the x-axis)



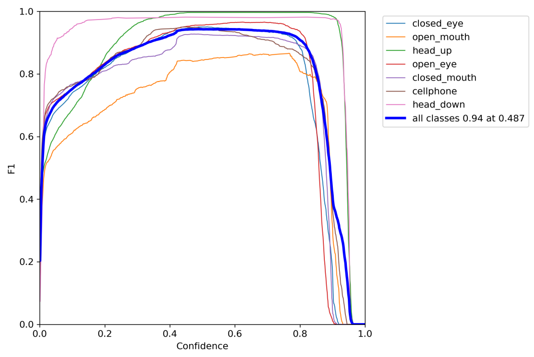
(b) Recall (in the y-axis and the epochs in the x-axis)

Figure 29: Results of precision and recall of the third training throughout each epoch

Figure 30a shows the precision-recall curve of the third training model. It can be seen that the majority of the classes achieved better results, where the open mouth class was the more noticeable. As for the F1 curve, figure 30b shows its results. The interval of confidence is not larger than the first trained model, but it is larger than the second. Also, it reached a higher F1 score. This means that in this model, the number of times a true positive (TP) is falsely recognized as a false negative (FN) has been reduced.



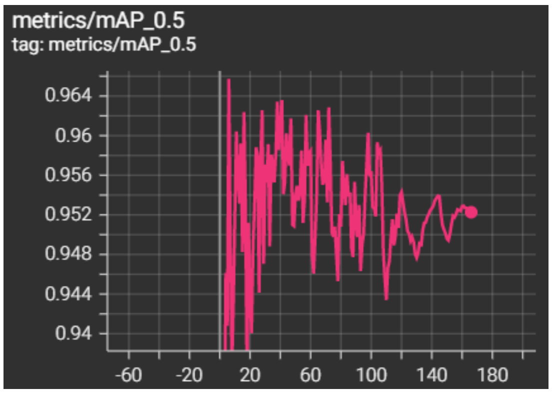
(a) Precision-Recall curve



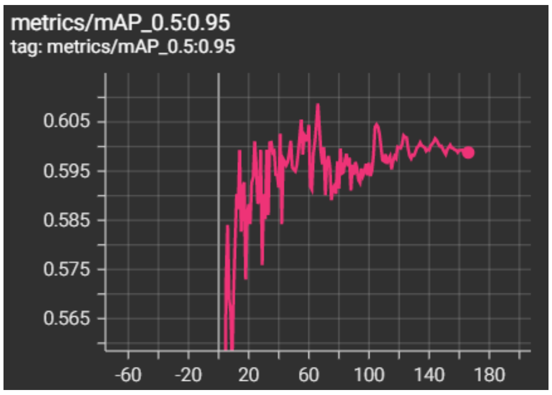
(b) F1 curve

Figure 30: Results of precision-recall curve and the F1 curve of the third training

After having analyzed the PR curve and F1 curve, expectations for this model to perform well were good. The results obtained for the map were then analyzed. The results can be seen in figure 31 and the best value achieved for mAP_0.5 was 96% (at 6th epoch) and the last value was 95% (at 166th epoch). As for mAP_0.5:0.95 the best one was 61% (at 66th epoch) and the last one was 60%. Comparing the values of these two metrics, to those obtained in the first and second training, in this third the results were better.



(a) mAP_0.5 (in the y-axis, epochs in the x-axis)



(b) mAP_0.5:0.95 (in the y-axis, epochs in the x-axis)

Figure 31: Mean Average precision (mAP) at IoU threshold of 0.5 and over different IoU thresholds, ranging from 0.5 to 0.95, of the third training

In overall, the results were good and better than the last training, that was used the same image size as argument. So the object detection model was putted into test, with the same conditions as the previous two were. Figures 32 and 33 shows the eye, mouth and head tracking at daytime and nighttime, respectively. As it can be seen, when it detects the head down, it does not detect the eyes and this is due to the fact that when a person looks down, the eyes are open, but seen from the front they appear to be closed, and in this way they are not detected, so there are no errors in the PERCLOS calculation.

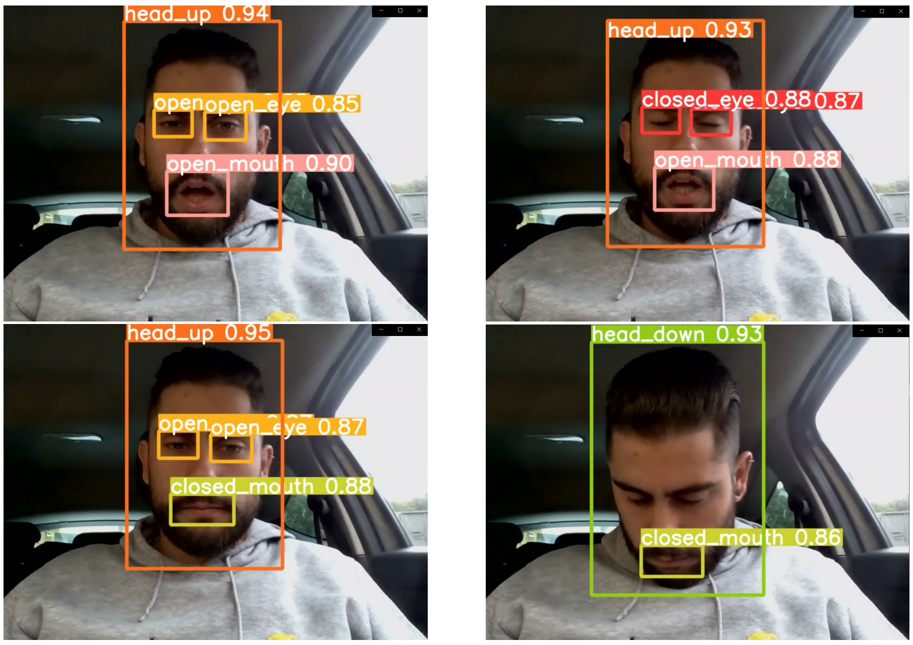


Figure 32: Test of the third trained model at daytime

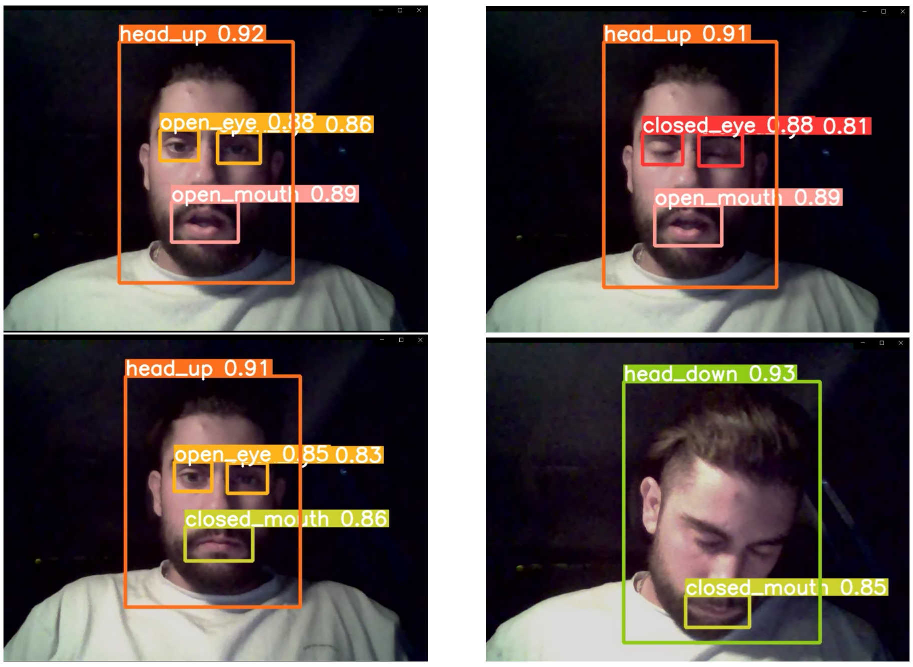


Figure 33: Test of the third trained model at nighttime

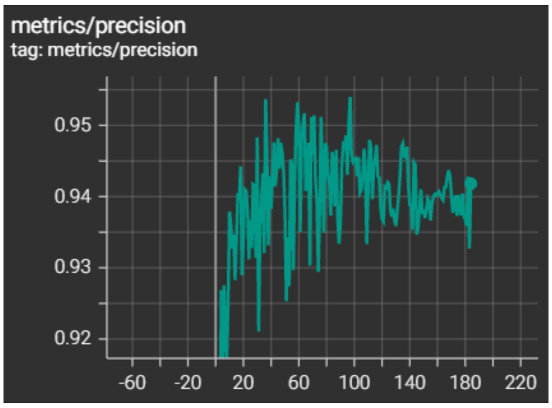
Concluding the analysis of this model, it can be said that the model has a satisfactory performance and as expected.

d) Fourth Training

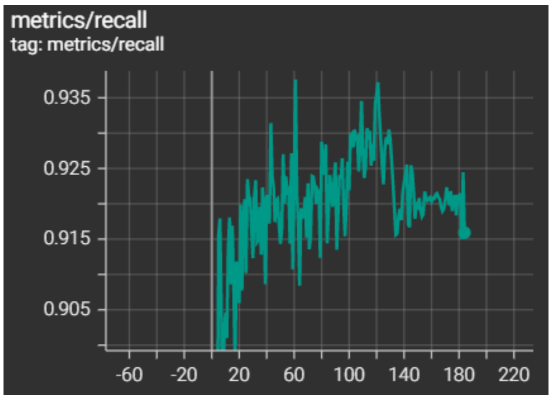
The third training model, is the chosen one to proceed to the development of the drowsiness detection model. However, a fourth training was done, but this time with the nano weight (yolov5n.pt). The reason of this it is that on a final step of the development of this project, a deployment on a Raspberry Pi is made in order to see how a device with low computational power behaves, and this weight is smaller than the other that was used in the last three trainings (yolov5s.pt). With this weight, the accuracy of the model decreases but the detection speed increases. The arguments were the same that the last training, except for the weight. The following command was used:

```
python train.py --img 320 --weight yolov5n.pt --batch 8 --epochs 200
--workers 2 --data dataset.yml
```

As it happened with the previous two trainings, this stopped training earlier too. In this model, precision and recall reached smaller values than the previous training. However, they are still good and better than the second training. As it shows in figure 34 precision reached 95% for the best value (at the 97th epoch) and 94% for the last one. Recall reached 94% for the best value (at the 61st epoch) and 91% for the last one.



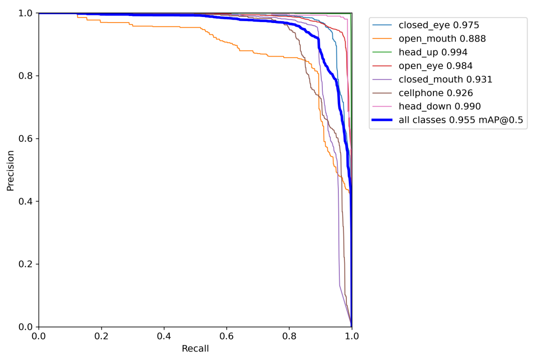
(a) Precision (in the y-axis and the epochs in the x-axis)



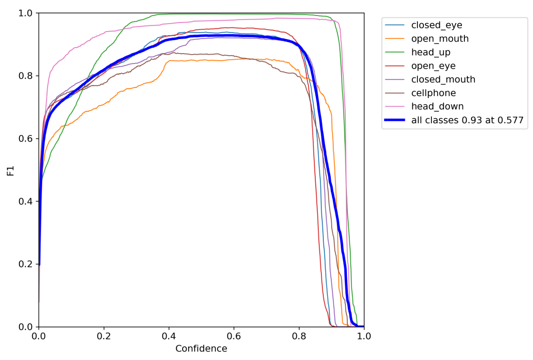
(b) Recall (in the y-axis and the epochs in the x-axis)

Figure 34: Results of precision and recall of the third training throughout each epoch

As it can be seen in figure 35a the precision-recall curve is very similar to the third model had. As for the F1 curve (figure 35b) the confidence interval is smaller but not so much like of the second training model.



(a) Precision-Recall curve

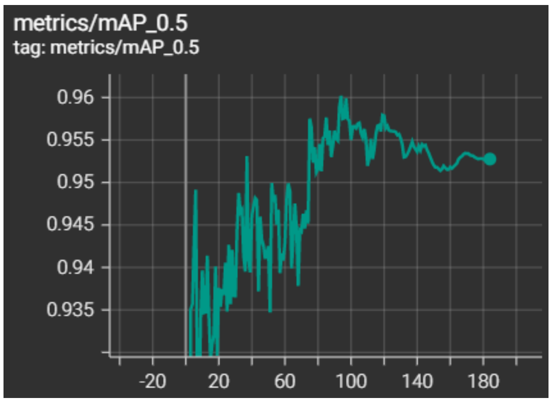


(b) F1 curve

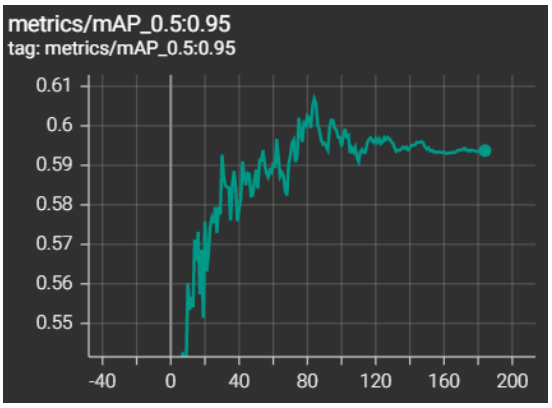
Figure 35: Results of precision-recall curve and the F1 curve of the third training

So, the expectations for this model were good, it only remained to analyze the mAP results, before testing it. As figure 36 shows, mAP_0.5 achieved the best value at the 94th epoch, being 96% and the last value being 95%. As for mAP_0.5:0.95 reached the best value at the 84th epoch, which was 60% and the last value was 59%. Comparing these values with those obtained previously, it appears that these are less than 1% than the others, which is not a huge difference, and still they are good results.

In a final step, the model was tested and it was concluded that the performance of this fourth trained model is as good as the third one.



(a) mAP_0.5 (in the y-axis, epochs in the x-axis)



(b) mAP_0.5:0.95 (in the y-axis, epochs in the x-axis)

Figure 36: Mean Average precision (mAP) at IoU threshold of 0.5 and over different IoU thresholds, ranging from 0.5 to 0.95, of the fourth training

For a final comparison, of each trained model and their results, table 4 shows the best and last values reached for each metric analyzed. For a more fair comparison, the first model results are not in the table, because this was trained only to detect only eyes and mouth, and with a larger image size.

Table 4 Object detection model experiment comparison results

	Best values					**Last values**				
	Precision	Recall	mAP_0.5	mAP_0.5:0.95	F1	Precision	Recall	mAP_0.5	mAP_0.5:0.95	F1
2nd	0.966	0.945	0.960	0.594	0.92	0.927	0.925	0.935	0.571	-
3rd	0.963	0.949	0.965	0.609	0.94	0.944	0.945	0.952	0.598	-
4rd	0.954	0.937	0.960	0.606	0.93	0.942	0.916	0.953	0.594	-

3.3 Drowsiness detection model

After the object detection model was successfully trained, with satisfactory results obtained, came the next step, the design of the drowsiness detection model. This, starts with the definition of a timing queue for the eyes and mouth. So when the object detection model detects the information about the eyes and mouth, it is stored the number of times (frames) that the eyes/mouth are open/closed inside this timing queue. This is defined to two minutes by converting the number of frames to time. Depending on the system it is running, the number of frames can be different. Because in this case, it runs with 30 FPS, the number of frames to define the two minutes is 3600. So when it reaches this value, all information is reset, and starts again. While counting the number of frames that the eyes/mouth are open/closed, PERCLOS and FOM are being calculated in real-time. Then, the values of PERCLOS and FOM are submitted to a threshold judgment. In the first 5 seconds of each window, the system does not trigger the alarm even though the PERCLOS and FOM values are being calculated. This is to prevent false alarms at the beginning of each window, because if the driver is not drowsy, but blinks in the first seconds, the PERCLOS value is going to be high. For a better understanding, figure 37 shows the algorithm made for calculating PERCLOS and FOM.

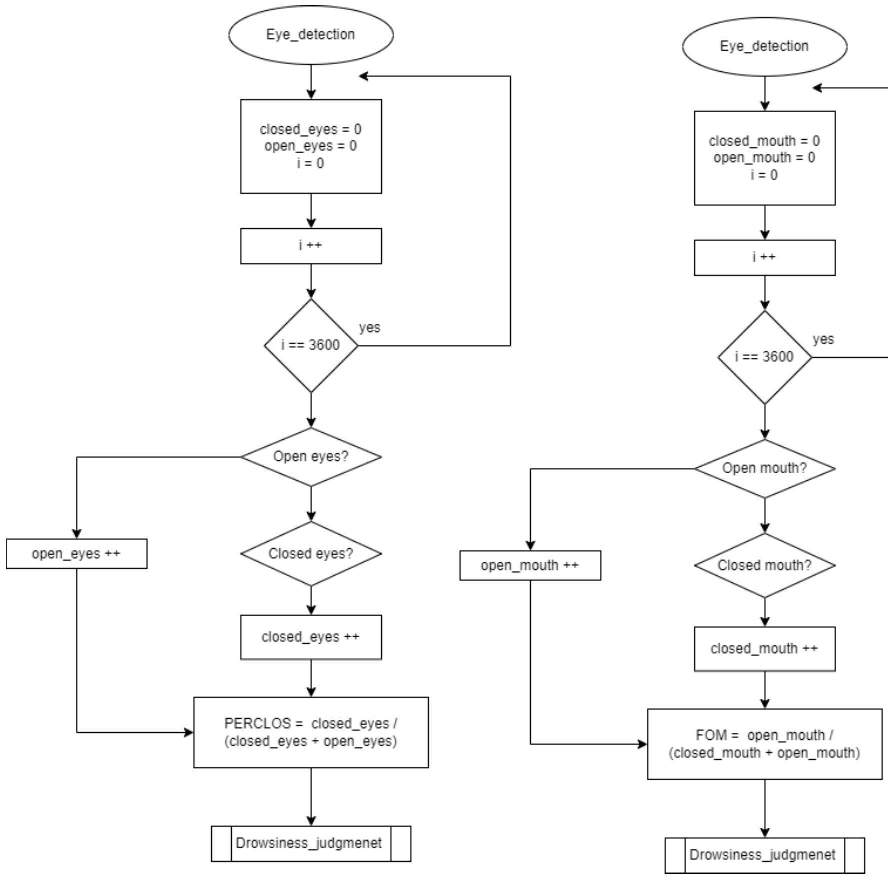
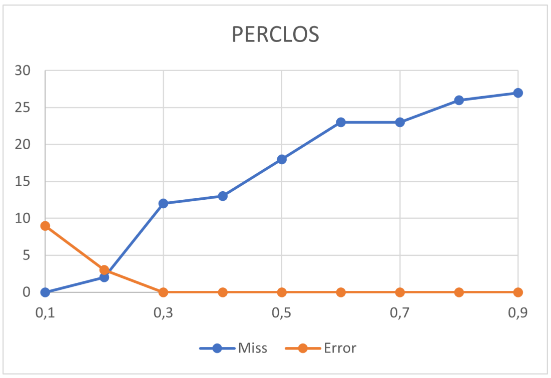


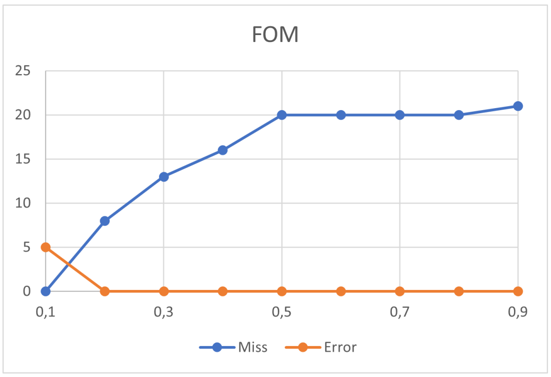
Figure 37: Flowchart of the algorithm made for calculating PERCLOS and FOM in a timing queue

PERCLOS and FOM values are submitted to a function that judges if the driver is drowsy or not. For that, a threshold for these two values had to be defined. In this order, five videos of five different persons simulating signals of drowsiness were recorded. These signals are different from person to person, so the goal with these recorded videos was to do a study made about the information acquired in order to establish the threshold for PERCLOS and FOM.

Like the authors in (Cui et al., 2021), two parameters were set to optimizing the thresholds, miss and error. Miss indicates that the driver is in a fatigue state, but the system is not successfully detected, and error indicates that the driver is not in a fatigue state, but the system is misjudged as a fatigue state. Figure 38 shows the relationship between these two parameters and the two metrics in question, in a line graph. Despite the authors in (Cui et al., 2021) having defined the thresholds for PERCLOS and FOM of 0.4 or 0.5 and 0.6 or 0.4, respectively, and like so many authors, in this case these values had to be smaller, because the goal is to give a simple warning when there is a minimum distraction or sign of fatigue. So, as it can be seen, in 38 when PERCLOS is 0.15 the miss and error parameters reach the lowest values. As for the FOM metric, when it is near 0.15, miss and error reach the lowest values as well.



(a) Relationship between PERCLOS, miss and error



(b) Relationship between FOM, miss and error

Figure 38: Experimental results diagram

After an analysis of the experimental results obtained, threshold values were defined for the drowsiness judgment. Figure 39 shows the algorithm of the function that determinate if the driver is drowsy or not. As it can be seen, the threshold for PERCLOS was defined as 0.15 or greater and FOM was defined as 0.2 or greater and the reason the threshold for FOM was defined as this value is that because drivers may be talking while driving, and with a smaller value of the threshold could lead to some false alarms. Also, the alarm can be activated just by PERCLOS. Drowsiness signals depends from person to person, and people can be seen in a fatigue state only by the closure of eyes. So for that, the threshold for PERCLOS was decided to be 0.2 or greater. That is due to the fact that because only eye tracking is considerate, eye blinking (a normal behavior) has to be in account as well, and the system with a lower value defined for PERCLOS's threshold, could falsely alarm people with a high rate of blinking (see figure 2).

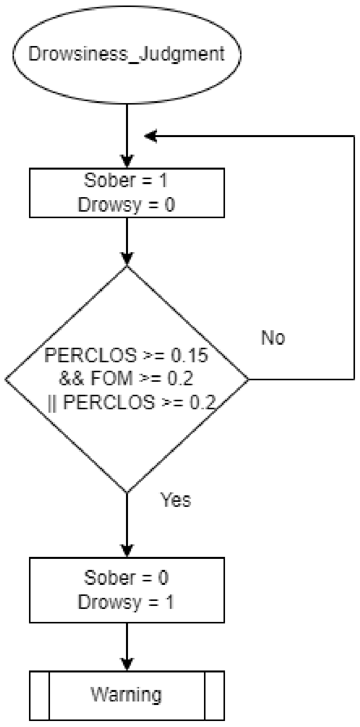
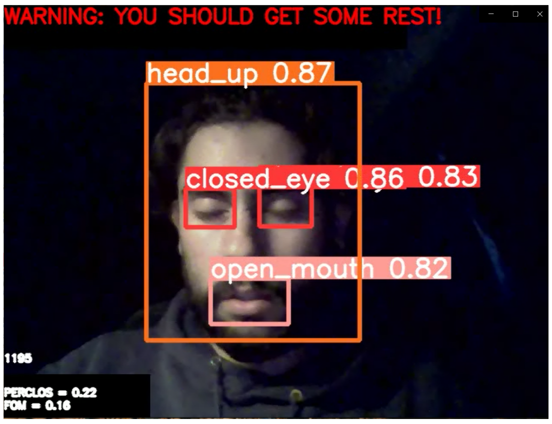


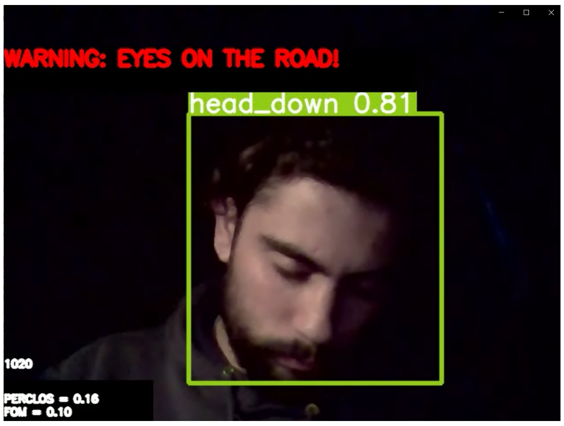
Figure 39: Flowchart of the function that verify the values of PERCLOS and FOM for the drowsiness judgment

The eye and mouth tracking combine well with each other, for the PERCLOS and FOM calculation and for the drowsiness judgment. The head position tracking could be added to these two in the same way, however, to judge drowsiness, this was decided to put it aside, and the reason is because the head position not only can demonstrate that the driver could be in a fatigue state, but also, can be a signal of distraction and not drowsiness. So for that, it was more simple to do than the PERCLOS and FOM way. The system gives a warning, if detects the head down in just three seconds in a row.

To test the designed system in real-time, the number of frames and the PERCLOS and FOM values were placed in the interface for better analysis and observation. Figure 40 shows signals of drowsiness being detected, and as it can be seen, in the bottom left corner, it shows the PERCLOS and FOM value in real-time, as well as the number of frames.



(a) Drowsiness detected by the values of PERCLOS and FOM



(b) Drowsiness detected by head position

Figure 40: Final test of the designed system

3.4 Deployment on a Raspberry Pi

Raspberry Pi is a tiny, cost-effective computer that comes in many shapes and sizes and facilitates a whole host of applications. Because it does not have the same computational power as a real computer, the challenge was to see how the designed system behaved on a weaker hardware. It was used the Raspberry Pi 4 model B.

To deploy the designed system on the Raspberry Pi, some essential libraries had to be installed and at the end it was putted into test. First, it was used the third trained model, that was heavier and the conclusions were the following:

- If the image size argument to detect is larger than the one used to train the model, it will have low recall, the board will consume too much power and eventually ends up crashing and turns off.
- If the image size argument to detect is equal than the one used to train the model, the system will detect the key regions well as expected, however it runs very slow and still it ends up crashing and turning off
- If the image size argument to detect is smaller than the one used to train the model, the system will run faster and not crashes, however the smaller the image size more information is lost, so it has to be used a middle term value.

Then it was tested with the fourth trained model, with the nano weights. The following conclusions were drawn:

- As same as it happens with the previous test, if the image size argument to detect is larger than the one used to train the model, it will have low recall, the board will consume too much power and eventually ends up crashing and turns off.

- If the image size argument to detect is equal than the one used to train the model, it runs faster than the previous model. However, the board still consumes some power and it ends up by turning it off.

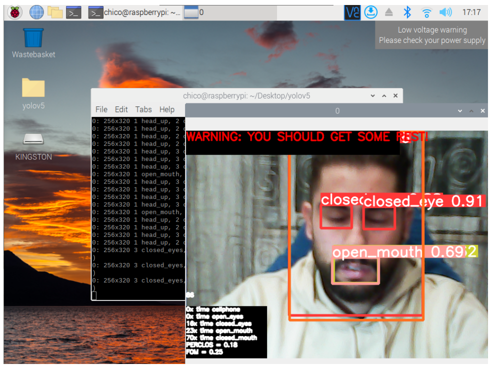
- If the image size argument to detect is smaller than the one used to train the model, it has the same behavior as it did with the previous model when using a smaller image size. But in conclusion, with this model, the system can run with a higher number of FPS.

Table 5 shows in a summarized way the analysis of the obtained results.

Table 5 Conclusion of results

A >B	A = B	A <B
Low recall and the system crashes	Key regions well detected, but runs very slow	System runs faster, but the smaller the image size (A) is, the more information is lost

Figure 41 shows one of the tests, that was used the third trained model, and with an image size of detection equal to the one used to train the model. As it can be seen, the system detects well all the key regions, as well that the driver is in a fatigue state, however, the Raspberry Pi warns about low voltage, due to the fact of the value of image size being greater, but also because it was powered by the computer during the tests.



```
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 3
0: 256x320 1 open_mouth,
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 3
0: 256x320 1 open_mouth,
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 2
0: 256x320 1 head_up, 3
0: 256x320 1 head_up, 2
0: 256x320 3 closed_eyes,
)
0: 256x320 3 closed_eyes,
)
0: 256x320 3 closed_eyes,
)
```

```
0x time cellphone
0x time open_eyes
16x time closed_eyes
23x time open_mouth
70x time closed_mouth
PERCLOS = 0.18
FOM = 0.25
```

Figure 41: Testing the third trained model with a image size of 320

To conclude, it was possible to see the designed system running successfully on a Raspberry Pi, however, using a camera with a lower resolution for input, and retrain a model with the same resolution as the camera could be a way to improve number of FPS and the power consumption.

4 Conclusion

Drowsiness on drivers is a serious and relevant problem that is the cause of many driving accidents, however, in a way tend to be underestimated because driving while tired is not punishable by law. Drinking coffee or energy drinks are not reliable solutions, and it is always best to stop the vehicle to rest. However, changing people's need to drive drowsy is not possible, and therefore the most reliable solution is the implementation of a fatigue detection system that alerts the driver when he is in this state. This results in a strong motivation for developing a countermeasure for this problem.

Researches were carried out in order to develop a system that detect signs of drowsiness in drivers and it was found that fatigue can be detected by using a lot of techniques. It was concluded that PERCLOS was the one that presented the best results in most of the works analyzed of other authors. So several solutions were considered, but the one that stood out the most was one that used two models, one for tracking key regions of the face, and another to determine whether the driver is in a drowsy state or not. So for the first model, it was chosen to use the object detection technique and after some research, it was concluded that YOLO, was the most suitable architecture by the time this work was carried out, because of its excellent results. YOLO has at this moment five major versions, and the last one was chosen because of its accuracy and high detection speed.

For this purpose, a dataset had to be built and restructured as needed, and various trainings were carried out to reach good results for the object detection model. Concluding that the results were satisfactory at the third and fourth training sessions, the drowsiness detection model was designed. Experimental results were obtained to see the best threshold values in order to give the warning of drowsy. Tests were carried out and it was concluded that the final results were satisfactory, however, the object detection model still can be improved by adjusting the dataset, training it in a better machine and giving it more time.

As for the deployment in the Raspberry Pi the designed system successfully runs on it, however, it still needs some improvements. The type of camera, the size of images of the dataset and the arguments used to train the object detection model have to be in consideration.

For future work, a hardware has to be built to implement the system in vehicles. An IR camera is a must-use on because it can give more night quality images for better detection. This hardware could communicate to a cellphone, and this is more purposefully for truck companies. Also, the system can be improved by improving the object detection model, like it was mentioned.

References

Caffier, P. P., Erdmann, U., & Ullsperger, P. (2003). Experimental evaluation of eye-blink parameters as a drowsiness measure. *European Journal of Applied Physiology*, *89*, 319-325. doi: 10.1007/s00421-003-0807-5

Cao, D., Chen, Z., & Gao, L. (2020, 12). An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks. *Human-centric Computing and Information Sciences*, *10*. doi: 10.1186/s13673-020-00219-9

Chehrehgosha, A., & Emadi, M. (2016). Face detection using fusion of lbp and adaboost. *Journal of Soft Computing and Applications*, *2016*, 1-10. doi: 10.5899/2016/jsca-00064

Cui, Z., Sun, H. M., Yin, R. N., Gao, L., Sun, H. B., & Jia, R. S. (2021, 7). Real-time detection method of driver fatigue state based on deep learning of face video. *Multimedia Tools and Applications*, *80*, 25495-25515. doi: 10.1007/s11042-021-10930-z

Deng, W., & Wu, R. (2019). Real-time driver-drowsiness detection system using facial features. *IEEE Access*, *7*, 118727-118738. doi: 10.1109/ACCESS.2019.2936663

Devi, M. S., & Bajaj, P. R. (2008). Driver fatigue detection based on eye tracking. In (p. 649-652). doi: 10.1109/ICETET.2008.17

Dinges, D. F., & Grace, R. C. (1998). Perclos: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance..

Edenborough, N., Hammoud, R., Harbach, A., Ingold, A., Kisačanin, B., Malawey, P., ... Zhang, H. (n.d.). *Driver state monitor from delphi proposal for demonstration at ieee cvpr 2005, san diego, ca.*

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*, 1627-1645. doi: 10.1109/TPAMI.2009.167

Gonçalves, M., Amici, R., Lucas, R., Åkerstedt, T., Cirignotta, F., Horne, J., ... Grote, L. (2015, 6). Sleepiness at the wheel across europe: A survey of 19 countries. *Journal of Sleep Research*, *24*, 242-253. doi: 10.1111/jsr.12267

Horng, W. B., Chen, C. Y., Chang, Y., & Fan, C. H. (2004). Driver fatigue detection based on eye tracking and dynamic template matching. In (Vol. 1, p. 7-12). doi: 10.1109/icnsc.2004.1297400

Jabbar, R., Al-Khalifa, K., Kharbeche, M., Alhajyaseen, W., Jafari, M., & Jiang, S. (2018). Real-time driver drowsiness detection for android application using deep neural networks techniques. In (Vol. 130, p. 400-407). Elsevier B.V. doi: 10.1016/j.procs.2018.04.060

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2021). A review of yolo algorithm developments. In (Vol. 199, p. 1066-1073). Elsevier B.V. doi: 10.1016/j.procs.2022.01.135

Lu, S., Wang, B., Wang, H., Chen, L., Linjian, M., & Zhang, X. (2019, 7). A real-time object detection algorithm for video. *Computers and Electrical Engineering*, *77*, 398-408. doi: 10.1016/j.compeleceng.2019.05.009

Manu, B. N. (2017, 3). Facial features monitoring for real time drowsiness detection. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/INNOVATIONS.2016.7880030

MRL. (2017). *Mrl eye dataset, http://mrl.cs.vsb.cz/eyedataset.*

of Electrical, I., Section, E. E. C., de Control Automático, A. C., of Electrical, I., & Engineers, E. (n.d.). *Ieee chilecon2017 proceedings : Pucón, chile, october 18-20, 2017*.

PRP. (2022). *Fatores de risco, https://prp.pt/prevencao-rodoviaria/fatores-de-risco/fadiga/, access: 03-01-2022.*

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, 6). You only look once: Unified, real-time object detection. Retrieved from `http://arxiv.org/abs/1506.02640`

Sulaeman, D., & Fauzi, R. (2022). *Factors related to work fatigue on truck drivers at pt kordon putra in sumedang regency in 2022* (Vol. 1). Retrieved from `https://ejournal.unsap.ac.id/index.php/phsaj`

Sun, Y., & Yu, X. (2014, 11). An innovative nonintrusive driver assistance system for vital signal monitoring. *IEEE Journal of Biomedical and Health Informatics*, *18*, 1932-1939. doi: 10.1109/JBHI.2014.2305403

Ultralytics. (2021). *Yolov5, https://github.com/ultralytics/yolov5readme.*

Viola, P., & Jones, M. J. (2004). *Robust real-time face detection* (Vol. 57).

Wang, Q., Yang, J., Ren, M., & Zheng, Y. (2006). Driver fatigue detection: A survey. In (Vol. 2, p. 8587-8591). doi: 10.1109/WCICA.2006.1713656

Works, M. (2022). *https://www.mathworks.com/discovery/object-detection.html, access: 04-02-2022.*

Yan, J. J., Kuo, H. H., Lin, Y. F., & Liao, T. L. (2016, 8). Real-time driver drowsiness detection system based on perclos and grayscale image processing. In (p. 243-246). Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/IS3C.2016.72

Yang, J., & Waibel, A. (2002, 12). A real-time face tracker. In (p. 142-147). Institute of Electrical and Electronics Engineers (IEEE). doi: 10.1109/acv.1996.572043

Zhang, C., & Eskandarian, A. (2021). A survey and tutorial of eeg-based brain monitoring for driver state analysis. *IEEE/CAA Journal of Automatica Sinica*, *8*(7), 1222-1242. doi: 10.1109/JAS.2020.1003450