Driver attention analysis and drowsiness detection using mobile devices

Master Thesis in ICT for Internet and Multimedia

Supervisor
Simone Milani

Master Candidate
Simone Ceccato

Academic Year 2018/2019
2 december 2019

A chi mi sostiene da sempre.

Abstract

Drowsiness and lack of attention are some of the most fatal and underrated accident causes while driving. With the increasing of fatal crashes during the last years and the advent of new technologies in the machine learning and computer vision fields, different strategies for drowsiness detection have been developed by car manufacturers, but they are not capable to detect early symptoms of drowsiness and fatigue yet.

In this thesis a non intrusive classifier based on features from drivers' facial movements has been developed, focusing on detection strategies that could be deployed on low-complexity devices, like smartphones.

A general overview of the different types of features will be given, motivating their effectiveness in the final classification task.

A detailed description of the features acquisition phase will also be presented, looking for a good trade off between optimal performances and accuracy.

Moreover, different classification architectures will be proposed and studied in order to understand which implementation performed the best in terms of detection accuracy.

Finally an overview of the developed Android application will be given, based on the concept of Clean Architecture and separation of concerns.

Sommario

La stanchezza e la mancanza di attenzione durante la guida fra le principali e maggiormente sottovalutate cause di incidente stradale. Negli ultimi anni, con l'aumentare degli incidenti mortali e con l'avvento di nuove tecnologie nell'ambito del *machine learning* e della *computer vision*, sono state sviluppate diverse strategie per la rilevazione della stanchezza da parte dei produttori dai produttori di automobili. Tuttavia, attualmente, nessuna di queste è capace di rilevare i sintomi di fatica e stanchezza con largo anticipo.

In questa tesi è stato sviluppato un classificatore non intrusivo basato sulle caratteristiche dei movimenti facciali del guidatore , ponendo particolare attenzione a soluzioni che possono essere implementate su disposivi a bassa capacità computazionale come gli *smartphones*

Nell'elaborato viene data una panoramica generale delle varie feature *facciali* e non che possono essere usate nella classificazione, motivando la loro efficacia nella procedura finale di classificazione.

Verrà presentata una descrizione dettagliata della fase di acquisizione delle caratteristiche, con riguardo alla ricerca di un compromesso tra prestazioni e accuratezza.

Inoltre saranno proposte e studiate diverse architetture per la classificazione allo scopo di determinare quale implementazione offre le migliori prestazioni in termini di precisione.

Infine sarà presentata la struttura dell'applicazione Android sviluppata, basata sui concetti di *Clean Architecture* e separazione dei moduli.

Contents

Listing of figures

Listing of tables

1 Introduction

Fatigue, microsleep and lack of attention at the wheel often cause serious accidents on the road as reported from the National Highway Traffic Safety Administration (NHTSA), which has estimated that only in 2017, 91,000 motor vehicle crashes involved drowsy drivers. These crashes led to an estimated 50,000 people injured and nearly 800 deaths [16], but there is broad agreement across the traffic safety, sleep science, and public health communities that the impact of drowsy driving is underestimated.

People often undervalue these types of impediments, but they are as fatal as alcohol and drugs. In fact, driver's reaction time increases and its environment awareness decreases leading to a serious increment of crashing risk [17].

Drowsy driving is a very common problem. According to some polls conducted by the National Sleep Foundation and others, about 60 percent of drivers admit driving while feeling sleepy, about 40 percent have nodded off or fallen asleep while driving during the prior year, and about one-quarter report drowsy driving at least once per month [18].

Lack of attention while driving is exponentially growing cause of car accidents during the last years due to the advent of smartphones in people lives. The National Safety Council reports that cell phone use while driving leads to 1.6 million crashes each year and, according to a AAA poll, 94 percent of teen drivers acknowledge the dangers of texting and driving, but 35% admitted to doing it

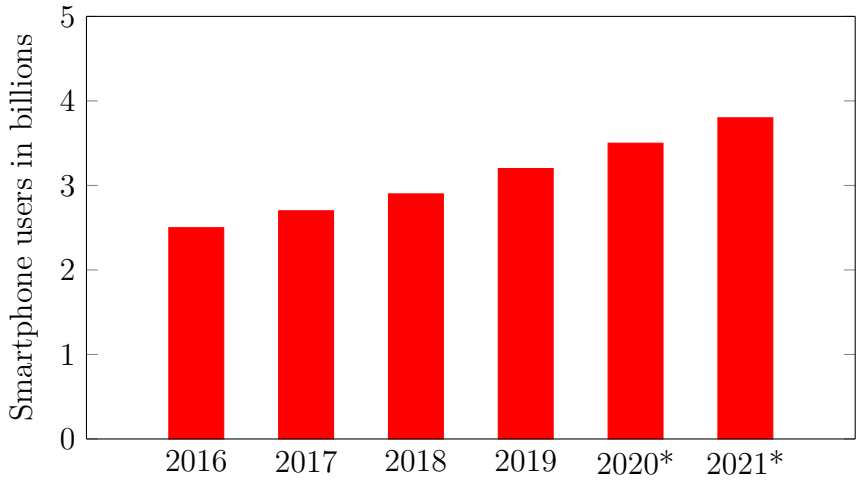


Figure 1.1: Number of smartphone users worldwide from 2016 to 2021 (in billions) (figure adapted from [1])

anyway.

For these reasons it is important to develop solutions that are able to detect drowsiness in drivers and act accordingly to some deterministic rules in order to preserve the safety of the drivers and the other people on the road.

With the roaring progress made in the autonomous driving field, the search for innovative solutions to these problems may seem unnecessary. On the contrary, the highest level in driving automation (i.e. level 5), where the vehicle is capable of executing all the driving functions under all the possible conditions without the supervision of a physical person, is expected to be available at least from 2025 and the transition between manual and automated cars will be slow and difficult [19].

The final purpose of this thesis is to develop a phone application that can be deployed on mobile devices (i.e. devices with limited computational power), run on real time and under any condition and be implemented in a product that can be used by the majority of people due to the lack of necessity to buy an additional device and to the fact that the number of people owning a mobile devices is growing and it is projected to be grown in the next years (as it is possible to see in Figure 1.1).

The thesis is composed by other 6 chapters in addition to the current one that serves as introduction to the topic.

Chapter 2 is a review of the current state of the art techniques in the drowsiness

and lack of attention detection field, starting from a physiological analysis of sleep in order to identify the possible measures to utilize in a classification model. A review of the current solutions adopted by car manufacturers and external companies is also provided.

Chapter 3 explains the features acquisition pipeline from the computer vision point of view, focusing on the various techniques used in order to improve the quality of the classification data that will be fed into the classification models.

Chapter 4 describe the different models used to perform the classification of the raw data extracted in the previous phase, focusing on the machine learning part of the thesis.

In chapter 5, the training results of the classification models are presented and analyzed.

Finally chapter 6 is composed by a brief summary of an Android application functioning, which follows a description of the programming architecture used to structure the entire code.

2

Literature review

2.1 Physiologic analysis

Sleep is an active period in which a lot of essential processing, restoration, and strengthening occur [20]. This physiological process is made of several repeating cycles with a mean duration of 90 minutes, where the body alternates between two different states (see Figure 2.1):

- Rapid Eye Movement (REM) sleep: the last part of the cycle in which muscle tone decreases and dreams occur.
- Non-REM (NREM) sleep: containing three sub-level of sleep (i.e. transition from being awake, light sleep and deep sleep), characterized by a low-probability of dreaming and a more active muscular activity.

The need of sleep is regulated by the circadian rhythm, a 24-hour internal clock that cycles between sleepiness and alertness at regular intervals and controlled by the hypothalamus, a portion of the brain responsible for many other physiological functions [21]. Other factors could affect this clock like darkness and bad sleeping habits [21].

The need for sleep is typically referred as **drowsiness** and can be defined as a transitional state between wakefulness and the entering into Stage I of the sleeping cycle, in which the body prepares to begin the sleeping process [22].

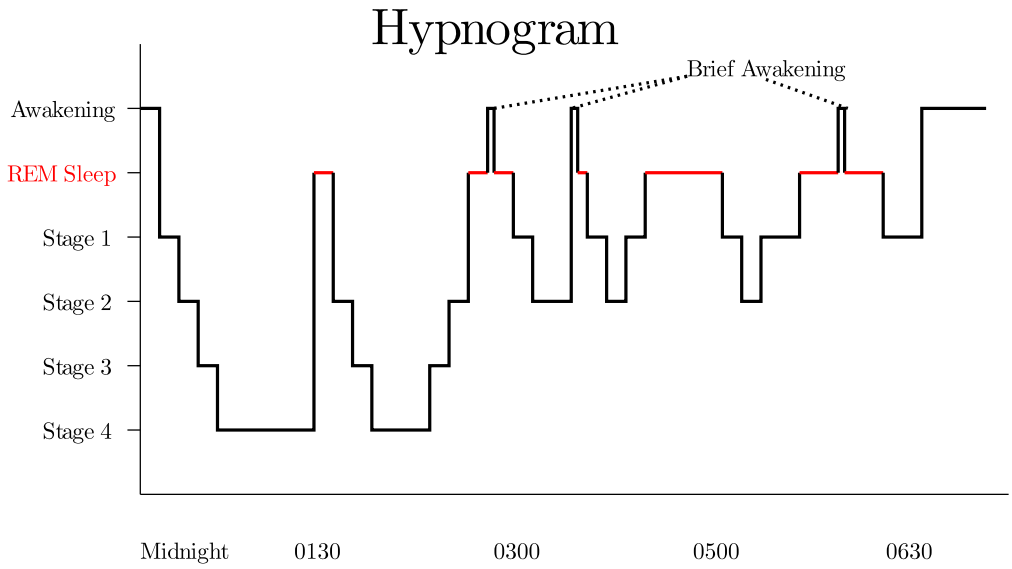


Figure 2.1: Typical sleeping cycle representation (figure from [2])

Fatigue can be scientifically defined as the progressive loss of processing efficiency, requiring a greater volume of neural tissue to perform normal tasks. This can be used with reference to the drowsiness concept, even though it has a have different clinical meaning and may be caused by different factors [23], [24]. A person can be fatigued without being sleepy/drowsy, but a person cannot be sleepy without being fatigued [25]. For this purpose detecting the levels of drivers' drowsiness has a key role in reducing the number of fatal injuries in traffic accident.

Driver fatigue can be divided into two class based on the factors: **sleep-related (SR)** and **task-related (TR)** fatigue [26]. The first one is strictly related to the person circadian rhythm, to sleep deprivation and to sleep disorders, as sleep apnea or narcolepsy. The second one is caused by the driving task and driving environment and can be further subdivided into active or passive fatigue [27], [28]:

- **Active fatigue** is related to the high demand of mental resources needed for the driving task, like driving amid high density traffic or with poor visibility.
- **Passive fatigue** is produced when the environment surrounding the driver

does not give him enough stimuli to stay alert. Examples of these situations are driving in monotonous roads like motorways.

Despite the different causes of fatigue, independently of being sleep related or not, the perceptible signs are essentially the same and includes [23]:

- difficulty on keeping the eyes open;
- frequent head movements away from the road;
- frequent yawning;
- daydreaming;
- drifting from lane, reckless driving and missing signs or exits;
- a general irritating sensation;
- slower reaction times.

This implies that a person experiencing these symptoms while driving represents a risk for himself/herself and for the entire surrounding environment. The development of technical solutions to this problem is therefore extremely useful.

2.2 Measures analysis

From the studies conducted in [29] and [30], it is possible to describe four types of measures in the drowsiness detection task:

1. **subjective measurements**;
2. **physiological measurements**;
3. **performance measurements**;
4. **behavioral measurements**.

In the next sections, a review of all the measures will be given, focusing more on the last one because the detection system build for this thesis is mainly based on these type of features. An hybrid method integrating behavioral and performance measurements is the final goal of my project with Texa and will be developed in the next months during my job hours at the company.

2.2.1 Subjective measures

Subjective measures rely on the evaluation, through a dedicated questionnaire, of the drowsiness state performed by the person itself or by an external supervisor. The common trait in these questionnaires is the representation of the level of drowsiness using a conformed scale [31].

The most used drowsiness scale is the Karolinska Sleepiness Scale (KSS) [32], a 9 level scale where the subjects rank their attention level ranging from completely alert to extremely drowsy as stated in Table 2.1

Rating	Description
1	*Extremely alert*
2	*Very alert*
3	*Alert*
4	*Fairly alert*
5	*Neither alert nor sleepy*
6	*Some sign of sleepiness*
7	*Sleepy, but no effort to keep alert*
8	*Sleepy, some effort to keep alert*
9	*Very sleepy, great effort to keep alert, fighting sleep*

Table 2.1: Karolinska Sleepiness Scale ratings description

Studies like [33] have already focused in validating the KSS, showing, as an example, that there is a relatively strong correlation between the KSS and electroencephalographic measurements.

The Epworth Sleepiness Scale (ESS) [34] measures a person's general level of daytime sleepiness, or their average sleep propensity in daily life, through the compilation of a questionnaire. Questions focus on retrospectively reporting the likelihood of dozing off or falling asleep in a variety of different situations.

A different approach is given by the Visual Analogue Scales (VAS) [31] where the rating is performed by asking subjects to give a rating to their level of drowsiness using a linear scale spread along a 100 mm wide horizonatl line where the

alert state stands on right end whilst the drowsy one stands on the left side. The final sleepiness level is measured by the distance in millimeters from one end of the scale to the mark placed on the line. The VAS is convenient since it can be rapidly administered as well as easily repeated.

A major drawback of this measurement type is the difficulty of implementation in real world driving conditions due to their subjective nature. Asking a driver to rate their arousal level may stimulate alertness, thus biasing the ratings. Variations in self-rated drowsiness can also be caused by stress or the use of drug substances. In addition, the average estimates have to be interpreted with caution, as there are considerable individual differences in the relationship between subjective ratings [32].

2.2.2 Physiological measurements

Physiological measurements offer a more objective and precise way to measure sleepiness. They are based upon the fact that physiological signals start to change in earlier stages of drowsiness, which could allow a potential driver drowsiness detection system some extra time to alert a drowsy driver and, thereby, preventing road accidents [31].

Main used signals include heart rate provided by electrocartiogram (ECG) and brain activity coming from an electroencephalogram (EEG);

The ECG is the process that records the electric activity of the heart and it is controlled by the autonomic nervous system (ANS), composed by the the sympathetic nervous system and the parasympathetic nervous system [35].

It is possible to describe the current state in a person by checking these two system: a wakefulness is characterized by an increase in the sympathetic activity or a decrease of the parasympathetic activity, whilst a drowsy or fatigued state is characterized by an increase of the parasympathetic activity or a decrease of the sympathetic one [35].

The EEG instead records electrical activity of the brain providing information in form of waves and it is often used in studies [36].

Drowsy states can be detected watching the frequency spectrum of waves in determined band [37], [36]:

- 8 - 12 Hz (i.e. alpha waves), which are considered when studying the transi-

tion from alert to drowsy and are a good indicator of the capacity of reaction to predetermined stimuli;

- 13 - 30 Hz (i.e. beta waves), which are a good indicator for a person increment in term of alertness and excitement;
- 4 - 7 Hz (i.e. theta waves), which are commonly regarded as a clear indicator of lack of attention and the onset of sleep;
- $<$ 4 Hz (i.e. delta waves),which are generated during deep sleep.

These measures provide a reliable source of information at the cost of building a system that is intrusive and not practical in a car environment.

In addition to that, signals like heart rate and EEG are excellent to distinguish between awake and sleep states, but encounter some problems in the identification of the intermediate ones.

2.2.3 Performance measures

Performance measurements are extracted monitoring the data coming from the main Electronic Control Unit (ECU) of the vehicle like steering wheel small movements, driving speed, lane deviations and brake patterns [37].

Steering wheel analysis relies on the fact that driver in a drowsy state performs a lower number of micro corrections during the driving rather than an alert one [38] and [39]. In order to remove all possible noise or the effect of line changing, only a small range of steering angles (i.e. from 0.5° to 5°) are considered [40]. The extraction of the data is performed using a sensor mounted on the steering column [41] but newer car models provide the data in the CAN bus, allowing external devices connected to the On-Board Diagnosis (OBD) port to read the values.

Driving speed also can be extracted from the values stored in the car ECU, but with the recent coming of high precision localization system into mobile devices, it is possible to perform an accurate estimation using a smartphone. A driver that is driving dangerously with abrupt speed changes is more likeable to be considered as a drowsy or distracted driver with respect to an alert one that, typically, drives in smoother way. Naturally these considerations cannot be implemented

in a generic software as the driving style is a subjective characteristics and can change significantly for different subjects.

Another good indicator of the driver drowsiness level is given by the standard deviation of lane positioning (SDLP) [42], obtained monitoring the car position within the lane is travelling using a lane detection system (typically implemented using a camera facing the road). A drowsy driver is less prone to follow the designated lane rather than an alert one, being more inclined to cross into an opposing traffic lane or off the road.

These technologies are often implemented on expensive car models since their development costs make them sustainable only for expensive cars. In addition, they tend to be too dependent on the geometric characteristics of the road (i.e. road marking, climatic and lighting conditions) [41], to the driving style of the subject and to other driver's states that are not related to drowsiness, like cognitive distraction and visual distraction (i.e. texting task) [43].

These disadvantages could possibly be overcome by the development of a robust system capable of adapt to the driver's typical behavior and car type in order to avoid false alarms [44]. Furthermore with the advent of automated driving functions, these type of measures becomes less and less correlated with real driver state because of the interactions of artificial intelligence driving systems to counteract the driver's errors [45].

2.2.4 Behavioral measurements

In this dissertation, the most relevant measurements are the behavioral ones because they meet the requirements imposed in the project. These types of features can be extracted using non intrusive and accessible devices, like a smartphone camera, and processed in real time given the high computational capabilities of modern phone processors.

Previous works in this field mostly focused on detecting extreme drowsiness with explicit signs such as yawning, nodding off and prolonged eye closure [44]. However, for drivers and workers, such explicit signs could appear when driver's drowsiness has already reached a dangerous level. For this reason is important to detect it at an early stage to avoid any type of incident [30].

The lack of large, public, and realistic datasets has been pointed out by re-

searchers in the field [44], [30].

It is possible to divide these features in two categories:

1. eye-related,
2. face-related

In this work, we do not take into consideration the movements of other parts of the body, although a relevant contribution in the classification task could be given by the motion of the hands. It is normal that a drowsy subject rub its eyes, as a natural consequence due to the lack of lubrication of the pupils. The detection of these events could bring a significant increase in detection accuracy but, due to the lack of computational resources, it could be unfeasible for mobile devices.

Eye-related features

The most informative region in a drowsy driver face are the eyes and their behaviour.

A good indicator of drowsiness in a subject is given by the Percentage of Eyelid Closure (PERCLOS) that can be described as the proportion of time that the subject's eyes are closed over a specific period of time.

In subjects experiencing drowsiness, PERCLOS measurements are higher than in alert drivers, as the eyes are closed for longer periods of time and more often than in alert drivers [46].

It is one of the most robust and used feature used in the drowsiness detection task but proper lighting conditions are needed and use of glasses and sunglasses limit this feature accuracy. Also, it might happen that a driver who is trying to stay awake is able to fall asleep with his eyes open [47], [48], [49], [50].

McIntire, McKinley, Goodyear and Nelson show, in their research, how blink frequency and duration normally increase with fatigue, by measuring the reaction time and using an eye tracker [51].

Svensson has shown that the amplitude of blinks can also be an important factor [52].

Friedrichs and Yang [53] investigated many blinking features like eye opening velocity (EOV), average eye closure speed, blink duration, micro sleeps and energy of blinks.

Also slow eyelid movement [54], decreased eyes openness level [55] and limited gaze movement area [56] have been noted in subject experiencing a sleepy or drowsy condition.

The gaze movement of a driver might be a great source of information of driver's mental condition because looking at other directions for an extended period of time may indicate fatigue or inattention.

Face-related features

Overall face behavior can also be used to detect sleepiness. Several studies have reported that in drowsy condition, a driver experience various changes in the face appearance like frequent yawning [57],[58], sudden nod and poor body posture [30].

Yawning is a very common feature extracted by a the driver face and it is a good indicator of a person fatigue. However, the detection implementation experiences several false negatives. It is common to detect yawning when the mouth opens [42], regardless of the fact that the driver is talking or singing. Therefore, we have to be careful in interpreting which open-mouth condition represents a real yawning.

Moreover when a driver is in a drowsy condition, the head may nod frequently. This is often followed by eye closure movements.

A key process to perform, in order to get information about driver's state is the head pose estimation, that is capable to provide a strong indicator of a driver's field of view and current focus of attention [59]. Intuitively, it might seem that looking at the driver's eyes might provide a better estimate of gaze direction, but in the case of lane-change intent prediction, for example, head dynamics were shown to be a more reliable cue [60].

One of the main drawbacks of these types of features is the need to have a proper lightning condition in order to get satisfactory results.

Normal cameras do not perform well at night and, in order to overcome this limitation, some researchers have tried active illumination using an infrared (IR) Light Emitting Diode (LED). Although working fairly well at night, LEDs are considered less robust during the day [41], [61].

A possible solution to this problem might be a system integrating both type

of cameras, the normal one working during the day (when they are capable to capture enough light) and the infrared during the night.

2.2.5 Hybrid methods

As described in the previous sections, each measurement method has its strengths and weaknesses, as we can see in Table 2.2.

Measure	Pros	Cons
Subjective	Important for ground truth	Not realtime
Physiological	Superb precision	Extremely intrusive
Performance	Good accuracy	Depend on too many external factors
Behavioral	Great accuracy	Depend on the camera quality. Not working with occlusions

Table 2.2: Review of the pros and cons of the various measures

To improve the classification task, a promising approach is to mix various measures in order to create an hybrid system which combines the advantages and robustness of all the measurements. Although some works have already combined different drowsiness detection methods, there is still a lot of possibilities for improvements [44].

Drowsiness analysis with driver's facial data and steering wheel data was also performed by [62]. Vehicle-based and behavioral measures are also combined in [63]. Works as [57] have combined all the measurement types, using indicators such as heart rate variability, respiration rate, head and eyelid movements (blink duration, frequency and PERCLOS), time-to-lane-crossing, speed, steering wheel angle and position on the lane.

2.2.6 Other factors considerations

It is important to consider the environment related to the behavior. When driving, a driver may face a certain condition that is challenging, or on the other hand,

boring. That condition may affect certain natural and habitual behavior of the driver.

It is possible to separate these conditions in two different type:

- **Inside-car conditions:** all the conditions inside the car that may affect driver's behavior. Clear examples of this are:
  - cockpit temperature. Comfortable conditions may soothe the driver smoothing and calming down his/her reactions but can also induce a sleepy state. Driving in uncomfortable condition could generate anger and annoyance in the driver.
  - sounds produced by music, people voice (from radio, talking, etc.). Calm, slow, and soft sounds (or even silent condition) make drivers feel comfortable and sleepy. When it is combined with warm temperature, it may increase the drowsiness.

- **Outside-car conditions:** all the conditions outside the car that may affect driver's behavior.
  - Long driving. Long and monotonous tracks can reduce the concentration and may cause distractions. In this case, the driver is not really fatigued, but monotony progressively reduces his/hers concentration and his/hers level of attention in controlling the vehicle.
  - Road conditions. Bumpy road, traffic jam, or challenging track may be something undesirable for the driver. But these conditions can increase his/her attention and make him/her more alert. On the other hand, a smooth and easy track, with light traffic, tends to make driver comfortable, raising the car speed and falling asleep more easily.
  - Driving time: the length of the trip may be a key factor in increasing driver fatigue due to the high mental workload generated in a subject by the driving task.

2.3 Current solutions

Currently on the market there are many solutions, created both by car manufacturers and by external companies, emplying the measurements presented in the previous section. Most of these use performance measures but more recently, behavioral ones focused the attention of researchers and engineers leading to the implementation of systems that integrates signals coming from the car and signals coming from the driver.

Figure 2.2: Mercedes Attention Assist alert example (figure from [3])

Mercedes Attention Assist [64] checks as many as 90 indexes, such as steering wheel angle and lane deviation, as well as external factors as weather and road surface conditions. After the drivers begin their journey, the system creates an individual profile of the driver, recognizing drivers' fully-alert condition. If the system detects drowsiness, it will send an audible and visible alert letting the driver know it is time to take a break as in Figure 2.2.

Driver Alert Control (DAC) by Volvo is a camera-based vehicle system that detects ideal road trajectory and compares it to steering wheel movements. The system provides sound alert and visual notification on control board when drowsiness is detected. Notification is repeated if driver behavior does not improve. The system is designed to function on highways and is activated when the speed exceeds 65 km/h [65], [66].

Bosch [67] has developed solutions as a steering-angle sensor that is sold to car manufacturers. In addition, from driver's steering behavior, the system algorithm evaluates approximately 70 signals such as the length of a trip, use of turn signals,

Figure 2.3: Bosch drowsiness detection system (figure from [4])

and the time of day. The function calculates the driver's level of fatigue. If that level exceeds a certain value, an icon such as a coffee cup flashes (as in Figure 2.3) on the instrument panel to warn drivers that they need a rest.

NVidia developed an artificial-intelligence tool, called Co-Pilot, that can learn the behaviors of individual drivers and determine when they are acting differently from their standard behaviour [68]. The system will evaluate driver's standard posture, head position, eye-blink rate, facial expression and steering style, among other indexes. Based on a vehicle's capabilities, the driver will be warned or automatically driven to a safe spot.

3

Acquisition framework

This chapter is focused on presenting the framework for the data acquisition phase and the relative feature extraction part. First an overall panoramic of the acquisition pipeline is given in order to introduce the reader to each specific description of the process.

The main goal of the acquisition phase is to collect the data needed to perform the classification of the current driver status, working with different machine learning and computer vision techniques in order to obtain some relevant features for the various classification models.

This stage can be divided in 3 major components as we can see from Figure 3.1:

1. A **preprocessing part**, where a series of image transformation are made in order to enhance and denoise the input and allow a more accurate detection.

2. A **raw features extraction part**, where the driver is detected using a deep learning approach and various features are extracted starting from the associated region of interest (ROI) in the image.

3. A **post processing part**, where the raw features are processed in order to create more complex characteristics of the driver behavior.

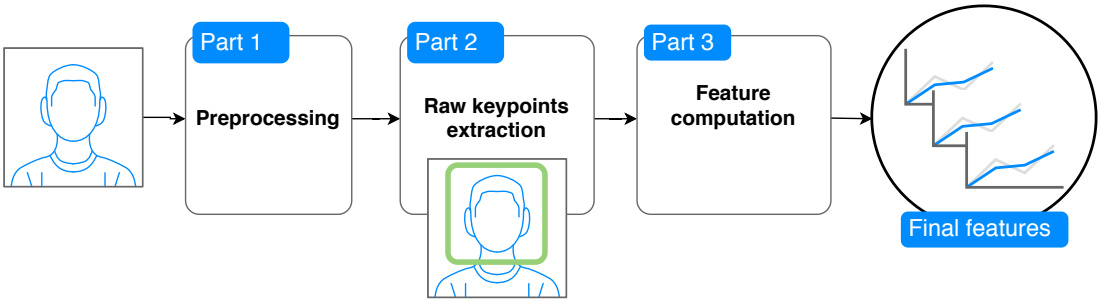


Figure 3.1: Acquisition pipeline flowchart

All these steps and the general system, were implemented in both a desktop and a mobile version using OpenCV methods and implementation, in compliance with the specifications required by the product.

Both versions of the software are capable of performing in real-time and are designed to mitigate the issues caused by the low illumination in the environment and the presence of obstructions in the subject face.

3.1 Frame acquisition

Smartphone cameras, like all today digital cameras, acquire images using an electronic image sensor rather than films.

There are two main type of image sensor in the market, but the general functioning idea is the same: the sensor converts the intensity of the incoming light rays into electric tension or current generating the final pixel value. A matrix of such photoresponsive sensors produces the final image.

The active-pixel sensor (APS) are mounted on most devices due to their low cost. They are made of a set of sensor cells containing a photodetector and one or more MOSFETs in order to amplify the input signal.

One of the main problem of the acquisition phase is the lightning conditions during a normal driving session. As said in the previous chapter, most of the drowsiness episode occurs during certain hour intervals during the day, including periods of time when the cockpit is not enough illuminated and the acquisition of an image with a quality that enables the feature computation is a hard task.

In order to tackle this problem, the first thing to do is to tune the camera physical parameters in such a way that the sensor is capturing the greatest possible

amount of light.

These two parameters are called exposure time and ISO: the first one refers to the length of time when the film or digital sensor inside the camera is exposed to light, whilst the second one controls the sensitivity of the image sensor (i.e. how much the sensor is sensible to light).

In automatic mode, camera firmware decides these parameter values in order to obtain the best "theoretical" shot, but, usually, it does not exceed the suggested *security range* to avoid the eventuality of capturing noisy or blurred shots.

Instead, when the manual mode is active, the photographer has full freedom in choosing the parameter values, as described in Figure 3.2

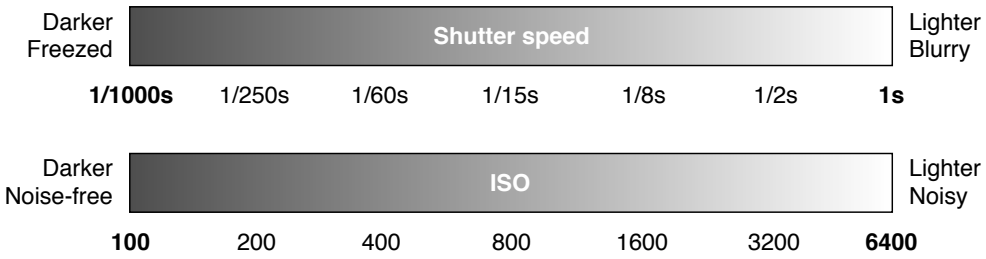


Figure 3.2: Camera parameters chart

An higher value of ISO generates a bright image, at the cost of acquiring a lot of noise as well. Similarly, the image looks brighter decreasing the shutter speed at the cost of depicting blurred objects, reducing drastically the number of images per second acquired by the camera.

In the problem investigated in this dissertation, each frame needs to represent a well-illuminated scene, regardless of noise. In fact, a noise reduction filter is applied to the image during the acquisition phase; moreover, in our conditions, either the face detector or the landmarks detector still work reasonably.

3.2 Preprocessing task

In the first part of the chain, the algorithm performs a preprocessing in order to improve the quality of the image and the accuracy of the detection.

A Gaussian blur operation is first performed on the input image, in order to reduce noise and reflections due to the eyeglasses eventually worn by the driver.

The Gaussian blur can be modelled by the filter

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

where x and y are the horizontal and the vertical pixel coordinates, respectively; σ is the standard deviation of the Gaussian distribution.

Then, the image is analyzed from the intensity point of view in order to select the brightest frames from the input flow. This is necessary, in the mobile implementation, due to the fact that the next steps in the pipeline are computationally expensive and the performance rate of the system is smaller than the actual frame rate delivered by the camera (i.e. in the majority of the smartphones equal to 30 frames per second). As a matter of fact, it is necessary to process only the most informative frames to save computational power and allow a real-time detection,.

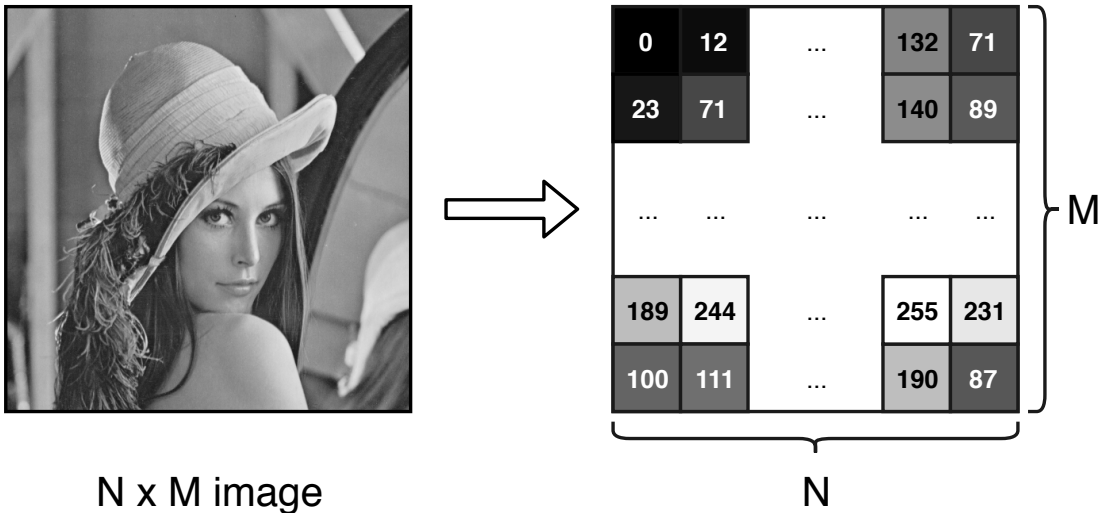


Figure 3.3: Grayscale image representation

Considering a gray-scale image, it could be described as a matrix similar to the one in Figure 3.3, where each cell represents a single pixel and contains a value ranging from 0 to 255 where the former stands for black and the latter for white.

To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB color-space). As a consequence, the pixel value is actually a vector of three numbers. The three different components are stored as three separate grayscale images known as color planes (one for each of red, green and blue).

Given an image with resolution NxM, the algorithm compute the total brightness of the image as

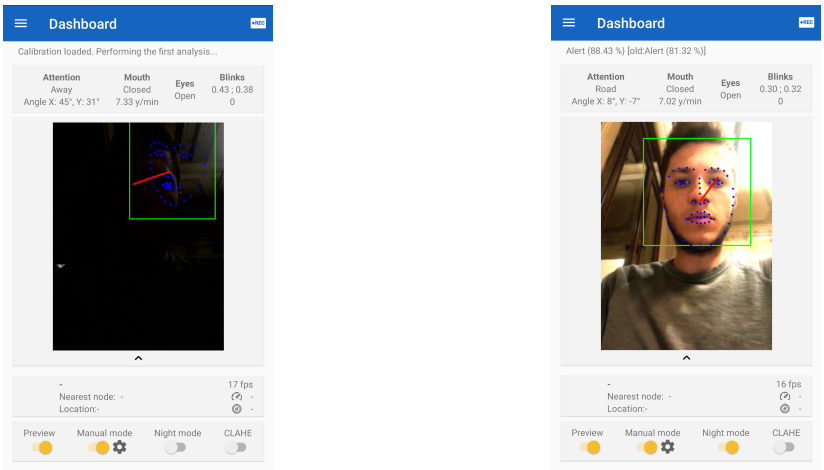
$$I_{frame} = \sum_{x=1}^{N} \sum_{y=1}^{M} g(x, y) \tag{3.2}$$

, where g(x,y) is the pixel intensity (in greyscale). It is possible to increase the brightness and the contrast of an image simply performing the following operation called **contrast and brightness adjustment**:

$$g_{output}(x, y) = \alpha \cdot g_{input}(x, y) + \beta \tag{3.3}$$

where $\alpha > 0$ is called *gain* parameters whilst β is called *bias* parameter and they represent relatively the change in image's contrast and brightness.

This step is called **Night mode step** and aim to artificially increase the frame luminosity after the shot was already taken. This is useful when the environment is very dark and the face detection task is not capable to extract the driver location due to the low information contained in the image. Despite the great noise introduction, the filtering techniques leads to the resolution of the problem in the majority of the cases. Naturally, using an infrared camera could lead to better performance, but excludes almost all the smartphones from the range of usable devices.



(a) Night mode disabled

(b) Night mode enabled

Figure 3.4: Comparison in dark environment

Better accuracy could have been reached using machine learning techniques in order to detect the portion of image where the driver is and increase the brightness of the selected part only. But, due to the computational constraint of the problem, the naive implementation has been used.

The processed frame is finally feed into the next step of the pipeline.

3.3 Raw features extraction

In the central part of the pipeline, the framework has to extract some raw features from the preprocessed input frame and the entire section can be subdivided in many different machine learning subproblems.

3.3.1 Face detection

Face detection is used in all the applications that require the identification and localization of human faces in a digital image.

Many approaches have been tried since 1964 when Bledsoe, along with Helen Chan and Charles Bisson, created the first semi-automated system. The initial approach involved the manual marking of the location of various landmarks on the face like eye center, mouth and nose. After that, the landmarks were mathematically rotated by computer to compensate for pose variation [69] and the distances between them were automatically computed. These distances were useful to compare different images in order to determine identity [70].

One of the most ground breaking development in this field was the work of Paul Viola and Michael Jones [5]. The Viola and Jones algorithm was created in 2001 it was the first object detection framework to provide competitive object detection rates in real-time.

It can be described as a machine learning approach based on three key contributions.

1. Use of integral image representation, which allowed a fast computation of the feature needed by the detector.

2. Implementation of a learning algorithm based on AdaBoost, which was capable of extract a small number of important features from a larger set.

3. The cascade combination of increasingly complex classifiers, which allows to discard background region of the frame spending more computational power on regions where faces are more likely to be detected.

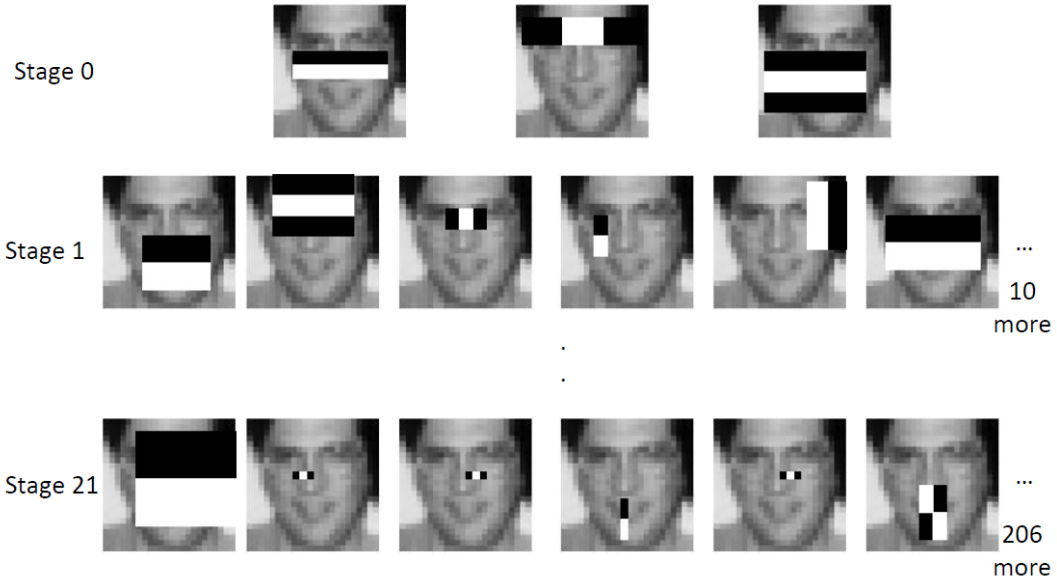


Figure 3.5: Viola and Jones features extraction phase (adapted from [5])

Although these methods works well in a controlled environment, they present some problems and limitations in a noisy environment like a car. The face detection task is, in fact, made complex by variability of poses, expressions, positions, orientations, skin colour, presence of glasses or facial hair, differences in camera gain, lighting conditions, image resolution and moving backgrounds.

In order to tackle these problems, deep learning has been applied to various implementations leading to results that outperform, in terms of accuracy, the previous ones and consolidating itself as the state of the art in the face detection task.

Dataset

As stated before, the challenges associated with face detection can be attributed to variations in pose, scale, facial expression, occlusion, and lighting conditions.

The first step in order to create a robust face detector is to choose an appropriate dataset for the training.

WIDER dataset [6] consists of 32,203 images with 393,703 labeled faces, which is 10 times larger than the second largest face detection dataset, with a high degree of variability in scale, pose and occlusion as depicted in Figure 3.6.

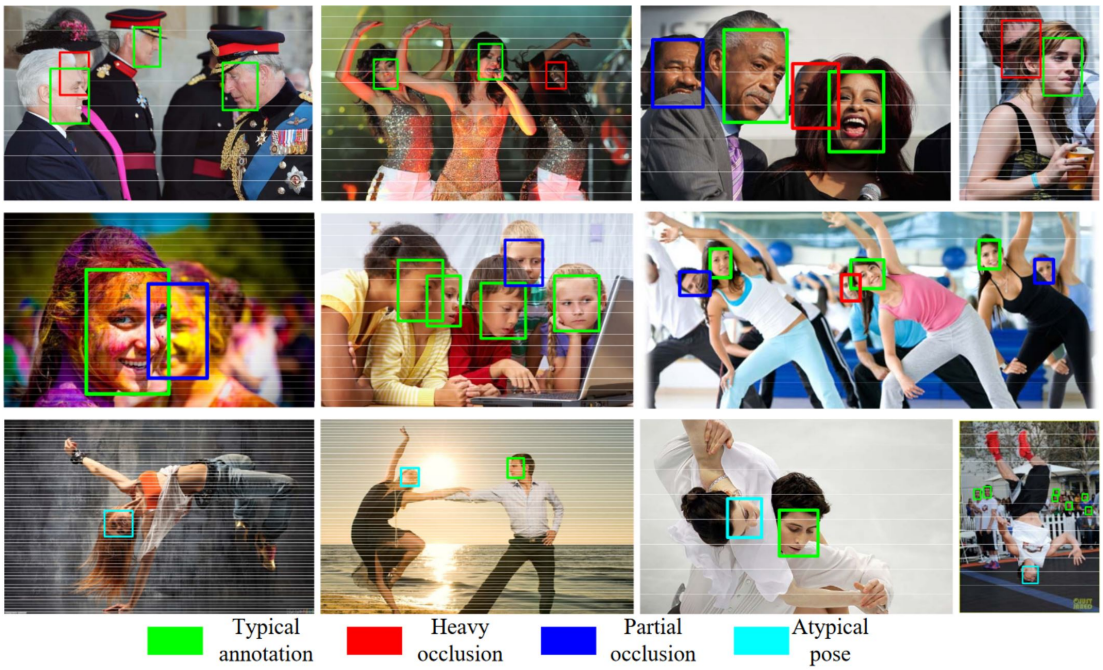


Figure 3.6: Wider dataset example (figure from [6])

As stated in the results of [6], the use of this dataset for the training allow the face detector to be more robust against extreme environments like the ones in a car cockpit, where the driver could move its head frequently and be exposed to different kind of illuminations.

Object detection task

Face detection is a specific task that can be related to the object detection field, where the main goal is to detect instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.

The key concept behind object detection is that every object class has its own special features. There are two main types of approaches in this field:

1. **Machine learning approaches** where it is necessary to first define a set of features like Histogram of Oriented Gradient features and then use a Support Vector Machine (SVM) to perform the classification.
2. **Deep learning approaches**: where the network (typically a CNN) is capable of automatically extract the right features to classify. Most famous are region proposal (R-CNN) methods, You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD).

In the first implementations, the key idea was using windows of fixed sizes from input image at all the possible locations feed these patches to an image classifier.

This method is simple but present some problems like size and aspect ratio for the subject to be detected. It is possible to solve this problem by resizing the image at multiple scales: the object will be encapsulated by the chosen window in one of these resized images.

Histogram of Oriented Gradient features (HOG)
On each window a Histogram-Of-Gradients (HOG) feature vector is calculated in order to fed to a Support Vector Machine (SVM) classifier. HOG features are computationally inexpensive and work well in many real-world problems.

Region-based Convolutional Neural Networks (R-CNN)
In order to improve the accuracy of the detections, HOG features are replaced in favour of recursive CNN-based classifier, named R-CNN.

Due to the fact that the sliding window search of possible objects locations is a extremely slow process and is unfeasible in low computational power devices, R-CNN introduces selective search [71] to generate about 2,000 region proposals.

Selective search uses, in the first step, color similarities, texture similarities, region size, and region filling to perform a non-object-based segmentation, obtaining a set of small segmented areas. Subsequently, these areas are merged together to form larger ones that represent the candidates in which the algorithm will compute CNN features, following a processing pipeline similar to the one in Figure 3.7

For each proposal region, a feature vector is computed with dimension that depends on the CNN used (e.g. AlexNet, Inception, GoogleNet) and classified using a SVM.

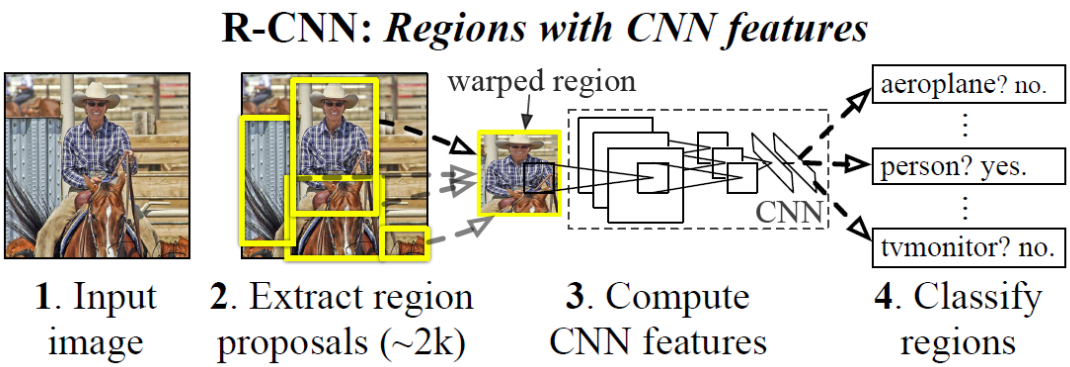


Figure 3.7: R-CNN processing pipeline (figure from [7])

Various optimizations of this algorithm has been developed (e.g. Fast R-CNN, Faster R-CNN) but they all rely on the concept of generation of region proposal and feature computation on these them.

You Only Look Once (YOLO)

YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities [72].

As we can see from Figure 3.8, YOLO divides every image into a grid of SxS and, for every cell in the grid, predicts N bounding boxes and confidence. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

YOLO works way faster than the previously described object detection algorithms, but finds difficulties in detecting small objects within the image (e.g. a flock of birds). This is due to the spatial constraints of the algorithm.

Single Shot Detector (SSD)

Single Shot Detector (SSD) algorithm provides a better balance between performances and precision.

The algorithm runs a convolutional network on input image only one time and then it computes a feature map. After that, a small 3×3 sized convolutional

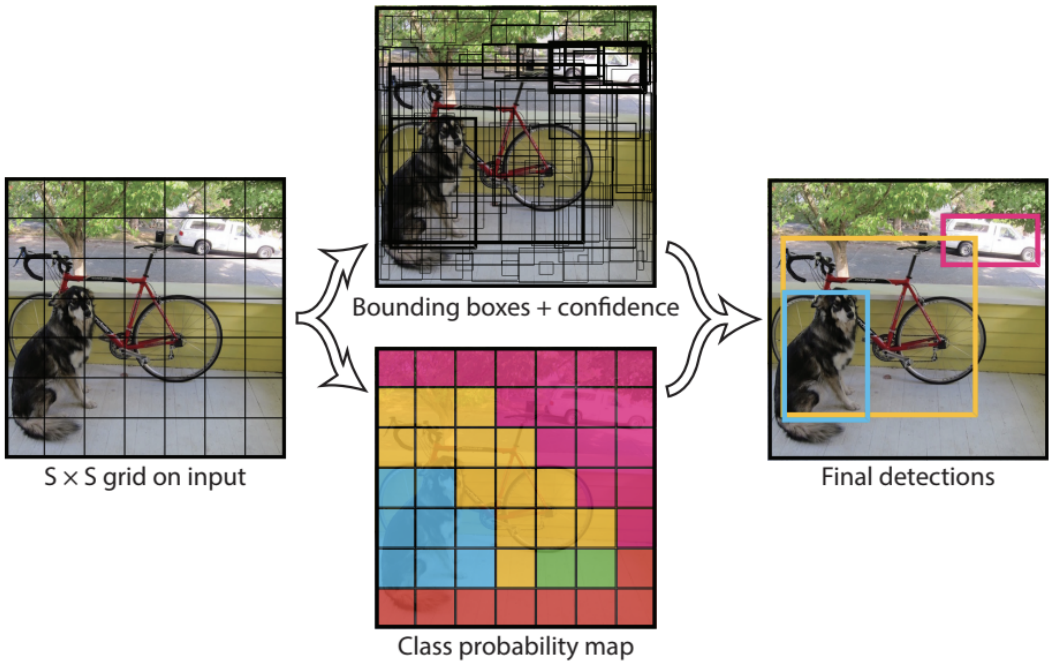


Figure 3.8: YOLO pipeline (figure from [8])

kernel processes the output feature map in order to compute the bounding boxes and the categorization probability.

SSD also uses anchor boxes at a variety of aspect ratio comparable to Faster-RCNN and learns the off-set to a certain extent than learning the box. In order to hold the scale, SSD predicts bounding boxes after multiple convolutional layers. Since every convolutional layer operates at a different scale, it is able to detect objects at different resolutions.

As we can see in Figure 3.9, the SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences.

SSD algorithm with a 300 $\times$ 300 input size significantly outperforms its 448 $\times$ 448 YOLO counterpart in accuracy on VOC2007 test dataset while also improving the speed [9].

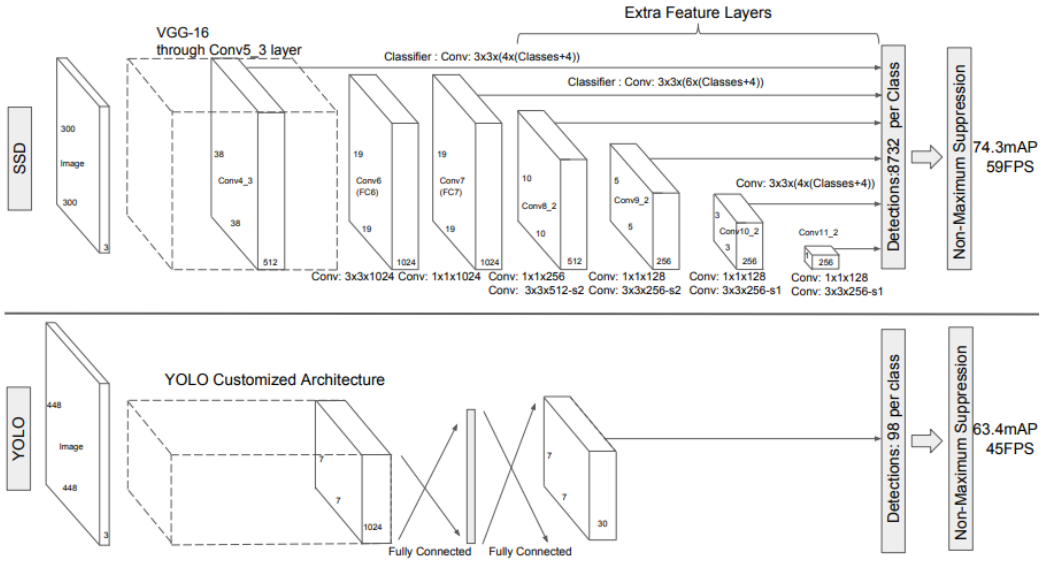


Figure 3.9: SSD vs YOLO architecture comparison (figure from [9])

Deep learning approach

As stated before, the introduction of convolutional neural networks significantly improved the classification accuracy in computer vision related problems. In those networks, the learned parameters are the filters that are applied iteratively through sliding windows to the input image. The filter is implemented by a matrix multiplication that, thanks to the modern GPUs, requires a limited computational power speeding up the training and inference phases of neural networks.

One theorem that justifies the importance of neural networks is the theorem of universal approximation, which states that a single layer neural network can approximate every existing function [73].

However, a single layer neural network could become wider and wider as the difficulty of the problem grows. This generates a massive network with over-fitting trends and high computation demand. The path taken by neural network research during these years is the development of a deeper and narrow network instead of a wider one, in order to simplify calculations and to allow the network to learn abstract representations of the data.

One of the main disadvantages of the use of deep learning is the appearance of the vanishing gradient problem. Whenever the gradient reaches a value close

to zero, the network can not update the weights any more. This effect is more evident as the back-propagation algorithm reaches the top of the network.

Several solutions were proposed to tackle this problem such as introducing various activation functions, like ReLU [74], or the usage of batch normalization.

In this thesis, a deep learning approach has been followed, implementing two different architectures, both based on SSD.

Residual Neural Network (ResNet)

ResNet is an artificial neural network (ANN) that it is based on the structure of pyramidal cells in the cerebral cortex.

The main innovation brought by this type of network is the introduction of the *residuals* inside the neural network through the use of an identity mapping that adds/subtracts the output of a block with its input. Residuals are one of the newest presented solutions for the vanishing gradient problem.

The main block in Figure **??** of the developed residual deep learning framework allows the gradient to flow directly from input to output without being degraded by the ongoing process.

This block has two parallel flows of data: one is a sequence of convolutional layers with activation functions and batch normalization, the other is an identity map function.

The second exists only if the input and the output have the same dimension and are summed at the end of the block.

This trick allows the network, which focuses on residual data, to reach a depth of hundreds of layers without facing the vanishing gradient problem and improving the features' learnability.

The stacking layers should not degrade the network performance, because it is possible to simply stack identity mappings (i.e. layer that are not active in the network) upon the current network, and the resulting architecture would perform the same [75].

This implies that the deeper model should not produce a training error higher than its shallower counterparts.

The authors of [76] demonstrated with experiments, that is possible to build a 1001-layer deep ResNet capable of outperform its shallower counterparts. Because of its compelling results, ResNet quickly became one of the most popular

architectures in different computer vision tasks.

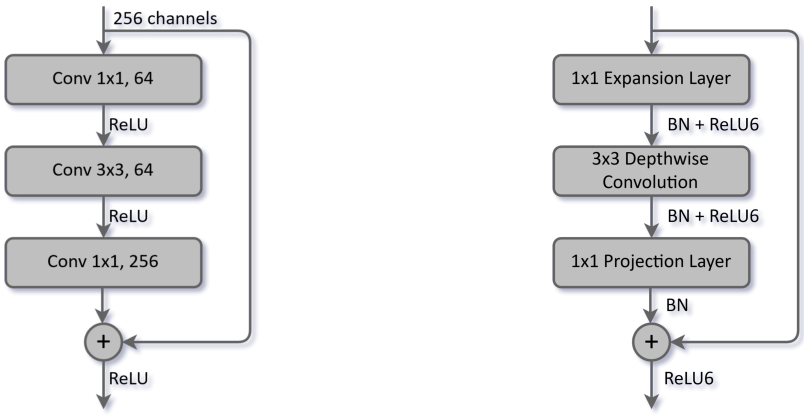
Even though deeper networks perform better in speed than wider ones, the ResNet architecture, which implies 58M of parameters, suffers from heavy computations during training even using modern and accelerated libraries or hardware. This would not be a problem for supercomputers with many GPUs connected in parallel but needs to be taken into account when it has to be implemented on mobile consumer devices.

MobileNet

MobileNets are a family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application.

These networks are small, low-latency and low-power models parameterized to meet the resource constraints of a variety of use cases in order to perform the classification and detection tasks on mobile devices.

They have in common with the ResNets the use of residuals, but they do not use any pulling layer and the stride parameter of the convolutional layer are employed to reduce the dimension of the input (see Figure 3.10b). The main module that is repeated throughout the network changing the parameter values is the bottleneck residuals block.



(a) ResNet residual block **(b)** MobileNet V2 residual block

Figure 3.10: Residuals block differences between ResNet and MobileNet

MobileNetV1 [77] is the first version of these types of network and introduced

depth-wise separable convolutions as an efficient replacement for traditional convolution layers.

Depth-wise separable convolutions effectively factorize traditional convolution by separating spatial filtering from the feature generation mechanism. Depth-wise separable convolutions are defined by two separate layers: light weight depth-wise convolution for spatial filtering and heavier 1x1 point-wise convolutions for feature generation.

With this trick, the number of parameters drastically diminishes together with the computation requirements.

MobileNetV2 [78] introduced the linear bottleneck and inverted residual structure in order to make even more efficient layer structures by leveraging the low rank nature of the problem. This structure is shown on Figure 3 and is defined by a 1x1 expansion convolution followed by depth-wise convolutions and a 1x1 projection layer. The input and output are connected with a residual connection if and only if they have the same number of channels. This structure maintains a compact representation at the input and the output while expanding to a higher-dimensional feature space internally to increase the expressiveness of nonlinear per-channel transformations.

MnasNet [79] built upon the MobileNetV2 structure, introduced lightweight attention modules based on squeeze and excitation into the bottleneck structure.

In the last version MobileNetV3 [80], the main goal is to improve the network efficiency, including all the improvements that the previous versions of the network have brought.

Both squeeze and excitation as well as the swish non-linearity use the sigmoid function which can be inefficient to compute as well challenging to maintain accuracy in fixed point arithmetic so MobileNetV3 replaces this with the hard sigmoid.

MobileNetV3 is defined as two models: MobileNetV3-Large and MobileNetV3-Small. These models are targeted at high and low resource use cases respectively.

Comparison

Both models were trained on the WIDER dataset with an input shape equals to (224, 224, 3) for both the networks. As it is possible to see in Table 3.1, both

	Input shape	Accuracy	Mean inference time			
			Google Pixel	Xiaomi MI 9T	Honor 8	Nokia 6.1
ResNet	(224, 224, 3)	72.05%	80.2 ms	73.5 ms	75.4 ms	134.2 ms
MobileNet	(224, 224, 3)	74.87%	45.8 ms	40.1 ms	41.8 ms	89.8 ms

Table 3.1: Face detector accuracy and inference speed comparison

networks achieve a good accuracy in the validation set but MobileNet outperform ResNet in terms of inference time.

For this reason, the chosen network for the face detection task is the MobileNet.

3.3.2 Performance improvement

In order to improve the performance in the face detection task that is, as it is possible to see in the previous section, the most expensive in terms of computational duration, an hybrid detection method has been deployed.

The key idea of this method is to detect the driver's face using a face detector implementation as the ones seen and use the region of interest (ROI) obtained to initialize a video tracking algorithm.

Video tracking is the process of locating a moving object (or multiple objects) over time using a camera. The objective of these types of algorithms is to associate target objects in consecutive video frames.

The main problem of tracking is that the association can be especially difficult when the objects are moving fast relative to the frame rate, leading to a misleading in the estimation of the driver's face.

To perform video tracking an algorithm analyzes sequential video frames and outputs the movement of targets between the frames. There are a variety of algorithms, each having strengths and weaknesses.

There are two major components of a visual tracking system:

- Target representation and localization: a bottom-up process (i.e. it process information as it is coming in) with low computational complexity.
- Filtering and data association: mostly a top-down process (i.e. it process information with regard to the context the information is surrounded) with an higher computational complexity.

Generally, the object tracking process is composed of four main steps:

1. Target initialization: the algorithm define the initial position of the target by selecting a box around it. The idea is to initialize the bounding box of the target in the initial frame of the video in order to let the tracker estimate the target position in the remaining frames in the video.

2. Appearance modeling: the algorithm has to learn the visual appearance of the object by using some learning techniques.

3. Motion estimation: in this part the algorithm tries to learn to predict a zone where the target is most likely to be present in the successive frames.

4. Target positioning: in the last part, the algorithm scans all the new possible object's location regions that were given by the previous step by applying the visual model generated before in order to estimate the new object position

In some cases where the visual appearance of the object is not clear, the tracking task can outperform the detection task:

- Occlusion: the object is partially or completely occluded.
- Identity switches: two objects cross each other.
- Motion blur: the object is blurred due to the motion of the object or camera. Hence, visually the object does not look the same anymore.
- Viewpoint variation: different viewpoint of an object may look very different visually and without the context, it become very difficult to identify the object using only visual detection.
- Scale Change: huge changes in object scale.
- Background Clutters: background near object has similar color or texture as the object. Hence, it may become harder to separate the object from the background.
- Illumination Variation: illumination near the target object is significantly changed. Hence, it may become harder to visually identify it.
- Low resolution: when the number of pixels inside the ground truth bounding box is very less, it may be too hard to detect the objects visually

Many different algorithms with very different implementations have been presented over the years and it is not the scope of this dissertation to analyze them. After a careful research, the tracker selected for the purpose was the Minimum Output Sum of Squared Error (MOSSE) tracker given the fact that represent a good trade-off between accuracy and performance.

MOSSE Tracker

Minimum Output Sum of Squared Error (MOSSE) tracker [81] uses adaptive correlation for the object tracking track in order to produce a set of stable correlation filters during the initialization with the help of a single frame.

The target is initially selected based on a small tracking window centered on the object in the first frame. From this point on, tracking and filter training work together.

The target is tracked by correlating the filter over a search window in next frame; the location corresponding to the maximum value in the correlation output indicates the new position of the target. An online update is then performed based on that new location.

To create a fast tracker, correlation is computed in the Fourier domain using the Fast Fourier Transform (FFT).

MOSSE tracker is robust to variations in lighting, scale, pose, and non-rigid deformations. It is also capable of detecting occlusion based upon the Peak-to-Sidelobe Ratio (PSR), which measures the strength of a correlation peak and enables the tracker to pause and resume where it left off when the object reappears.

To add to the positives, it is also very easy to implement, is as accurate as other complex trackers and much faster.

3.3.3 Hybrid implementation

The combination of the face detector and the object tracker leads to the creation of a new type of face provider that is suitable for a mobile implementation and that achieves a comparable accuracy with respect to a only deep learning solution.

As we can see in Figure 3.11, the algorithm checks if a face has already been discovered and, depending on the response, decides to use either the deep-learning

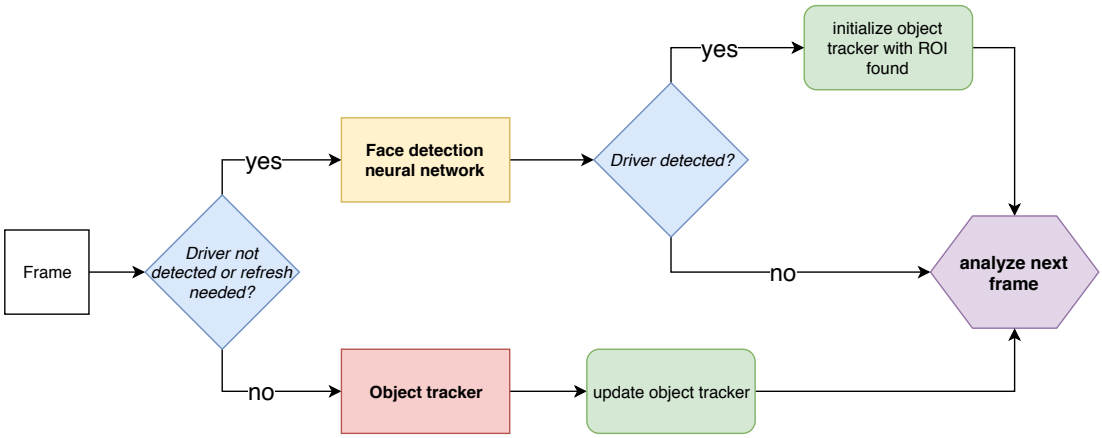


Figure 3.11: Hybrid face detection algorithm pipeline

detector or the object tracker.

In the first case, the algorithm uses the neural network to locate the driver's face: if it finds an acceptable result, it initialize the MOSSE tracker with the ROI just found, else it discard the frame and try the successive one.

In the second case, the algorithm update the current ROI of the driver's face using the object tracker.

In order to avoid the accuracy problems brought by the introduction of a less accurate detection performed by the tracker, the current driver's face ROI is updated using the neural network every 30 frames computed by the algorithm, even if the driver is continuing to be detected by the object tracker.

This approach leads to a big improvement in the computational speed of the pipeline, allowing it to perform the face detection on all the frames provided by the camera and drastically reducing the time needed for the face detection task.

3.3.4 Face landmarks localization

Face landmark detection is the process of finding points of interest in an image of a human face. It has recently seen rapid growth in the computer vision community because it has many compelling applications like the ability to detect emotion through facial gestures, to estimate gaze direction and to change facial appearance (face swap).

In this work they are particularly useful in order to obtain some key features from the driver, as we can see from Figure 3.12 but they are also fundamental in

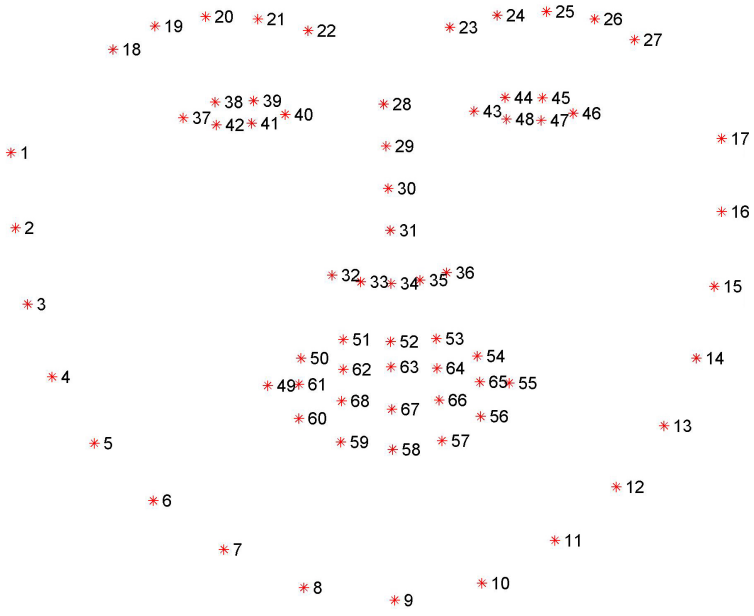


Figure 3.12: 68 points generic face representation

the computation of more abstract features regarding the its driving behavior.

The key idea in this task, is to define a set of 68 points (called *landmarks*) that are present in every human face and represent the location of different parts of the subject's face in a two dimensional space, in order to train a machine learning algorithm capable of detecting these points in the input image.

Those key points are either the dominant points describing the unique location of a facial component (e.g., eye corner) or an interpolated point connecting those dominant points around the facial components and facial contour.

Formally, given a facial image denoted as I, a landmark detection algorithm predicts the locations of D landmarks $\mathbf{x} = \{(x_1, y_1), (x_2, y_2), ..., (x_D, y_D)\}$, where x and y represent the image coordinates of each facial landmark.

Many facial landmark detection algorithms have been developed to automatically detect those key points over the years and it is possible to divide them into three major categories that differ in the way they utilize the shape information and the facial appearance:

- **Holistic methods**, which explicitly build models to represent the global facial appearance and shape information.

- **Constrained Local Model (CLM) methods**, which leverage the global shape model but build the local appearance models.
- **Regression-based methods**, which implicitly capture facial shape and appearance information.

Several studies like [82] and [83], stated that regression-based models outperform the other models both in speed and accuracy metrics.

Regression-based methods

The regression-based methods, as explained before, directly learn the mapping from image appearance to the landmark locations, differentiating themselves from the Holistic Methods and Constrained Local Model methods, which usually do not explicitly build any global face shape model.

Instead, the face shape constraints may be implicitly embedded. In general, the regression-based methods can be classified into three subtypes of methods:

- **Direct regression methods**, which predict the landmarks in one iteration without any initialization.
- **Cascaded regression methods**, which perform cascaded prediction and usually require initial landmark locations.
- **Deep learning based regression methods**, which follow either the direct regression or the cascaded regression.

Among different regression methods, cascaded regression method achieves better results than direct regression. Cascaded regression with deep learning can further improve the performance.

One main issue for the regression-based methods is that, since they learn the mapping from the facial appearance within the face bounding box region to the landmarks, they may be dependent on the adopted used face detector and the quality of the face bounding box. Because the size and location of the initial face is determined by the face bounding box, algorithms trained with one face detector may not work well if a different biased face detector is used in testing.

Facial landmark detection is challenging for several reasons:

1. Facial appearance changes significantly across subjects under different facial expressions and head poses.

2. Challenging environmental conditions such as the extreme illumination would affect the appearance of the faces on the processed images.

3. Facial occlusion by other objects or self-occlusion due to extreme head poses would lead to incomplete facial appearance information.

Local Binary Features model

The selected landmarks detector for this dissertation is the one described in [84]. It is an highly efficient and accurate regression model based on the concept of Local Binary Features, a set of features extracted using a simple texture operator which labels the pixels of an image by t the neighborhood of each pixel and considering the result as a binary number, as we can see in Figure 3.13

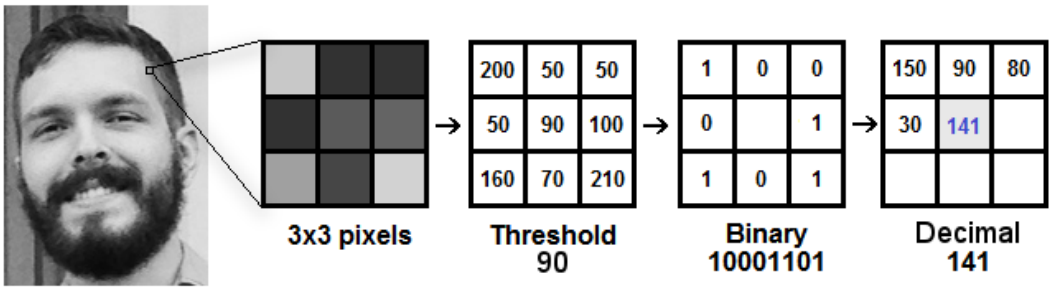


Figure 3.13: Local Binary Operation (figure from [10])

Thanks to the *locality principle* that states that an object is directly influenced only by its immediate surroundings, the algorithm learns a set of highly discriminative local features for each facial landmark independently.

The obtained local binary features are then used to jointly learn a linear regression for the final output. This learning process is repeated for a determined number of stages in a cascaded fashion, in order to improve the quality of the point localization.

Given the fact that extracting and regressing local binary features is not computationally demanding, this algorithm is capable of process over 3,000 frames

per second on a desktop computer or 300 frames per second on a mobile phone in the task of locating the selected landmarks.

The paper's authors discussed on the validity of choosing to perform the regression task independently for each landmarks, because such approach could miss a good feature that can be shared by multiple landmarks.

They motivated their architecture choices in three different points:

1. the feature pool in local learning is less noisy with respect to a global learning, leading to an easier feature selection.

2. performing local learning does not mean that a local prediction will be executed. In their approach, the linear regression in the second step exploits all learned local features to make a global prediction.

3. the local learning is adaptive in different stages. In the early stage, the local region size is relatively large and a local region actually covers multiple landmarks. The features learned from one landmark can indeed help its neigh-boring landmarks. At the late stage, the region size is small and local regression fine-tunes each landmark.

The model returns the set of 68 landmarks described in Figure 3.12 and with the components described in Table 3.2

Landmark	**Identified by**
Chin	1 - 17
Right eyebrown	18 - 22
Left eyebrown	23 - 27
Nose	28 - 36
Right eye	37 - 42
Left eye	43 - 48
Mouth	49 - 68

Table 3.2: Landmarks - ID correspondence

As we can see from Table 3.2 it is possible to retrieve set of points describing different parts of the driver face.

Two important values to extract from these set of points are the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR) that represent how much the eyes and the mouth, respectively, are open.

Eye Aspect Ratio (EAR)

Eye Aspect Ratio is described in [85] as the ratio between the height and the width of the eye and it is computed as follow:

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||} \tag{3.4}$$

where p_1 represent the external corner, p_4 the internal one, p_2, p_3 the upper external and internal, respectively, superior eyelid and p_6, p_5 the upper external and internal, respectively, superior eyelid

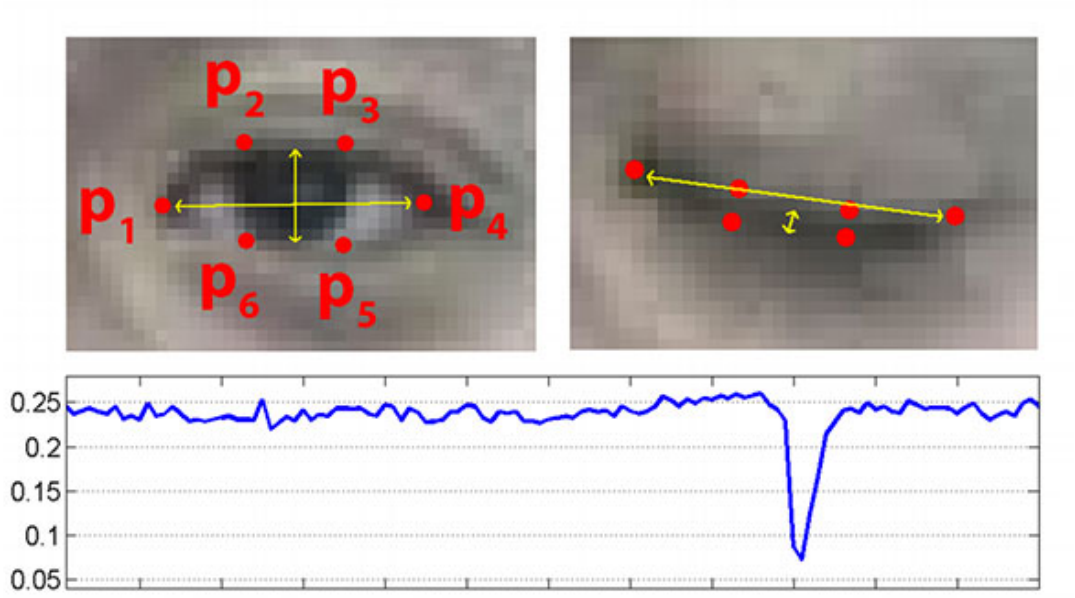


Figure 3.14: Eye Aspect Ratio typical behavior (figure from [11])

EAR is mostly constant when an eye is open and is getting close to zero while closing an eye as we can see in Figure 3.14. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face.

EAR is computed for both left and right eye and then averaged in order to obtain a general indicator of the eyes closeness.

$$EAR_{mean} = \frac{EAR_{left} + EAR_{right}}{2} \tag{3.5}$$

Mouth Aspect Ratio (MAR)

Similarly to EAR, it is possible to use this kind of approach to determine the order of opening of the driver's mouth.

Differently from the eyes, the facial landmarks detector returns a greater number of points that can be split into two set of points representing the upper and the lower lip respectively.

The key idea is to focus only on the lower part of the upper lip (i.e. points in the range [62, 64]) and the upper part of the lower lip (i.e. points in the range [66, 68]). These three points will represent the height of the mouth aperture. The width will be given by points 61 and 65.

It is possible to derive an equation similar to Equation (3.4)

$$MAR = \frac{||p_2 - p_8|| + ||p_3 - p_7|| + ||p_4 - p_6||}{3||p_1 - p_5||} \tag{3.6}$$

where p_1 represent the right corner of the mouth, p_5 the left one, p_2, p_3, p_4 the upper lip points starting from right and p_8, p_7, p_6 the lower lip points starting from right.

MAR shares the majority of characteristics with EAR (naturally the value is mostly constant when the mouth is closed and increases when the mouth is open) but, as explained in the previous chapter, represent a greater set of possible driver's behavior than just the yawning task.

For this reason, MAR is not a reliable parameters to determine the driver's yawns and needs to be integrated in a more complex system.

3.3.5 Head pose estimation

Having obtained the facial landmarks, it is possible to find the direction of the driver's face. The 2D face landmark points essentially fits to the shape of the head. So, given a 3D model of a generic human head, an approximate set of corresponding 3D points can be estimated.

This process, called head pose estimation, can be used to retrieve a generalized set of drivers' head movements with respect to that described in Chapter 2, checking whether the driver is looking at the road or if he/she is nodding.

The pinhole camera model describes the mathematical relationship between the coordinates of a point in three-dimensional space and its projection onto a two-dimensional plane called **image plane**. This model could be seen as the simplest approximation of a camera where distortions, blurring or unfocused object are not included. While some effects are sufficiently small to be neglected, others can be compensated applying different coordinate transformation on the image coordinates.

Despite these heavy approximations, the model can be reasonably be used to parameterize how a normal camera acquire a three-dimensional scene.

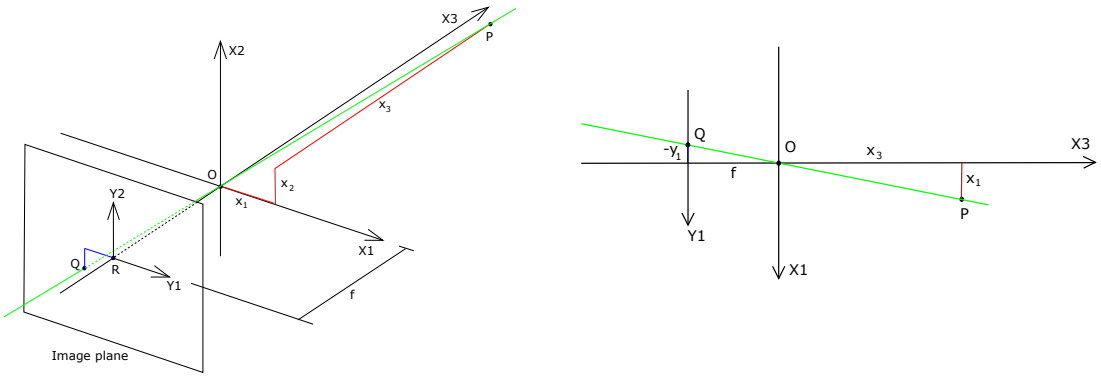


Figure 3.15: Pinhole model representation

As we can see from Figure 3.15 it is possible to mathematically describe the world with a three dimension orthogonal coordinate system with its origin at O, and the three axes referred as X, Y and Z. Axis Z is defined to be the *optical axis* (or *principal ray*) and represents the viewing direction of the camera.

The plane which is spanned by axes X and Y is the front side of the camera, or *principal plane*. The origin correspond to the center of projection.

The *image plane*, located at distance f from the origin, in negative direction with respect to y axis and parallel to *principal plane*, is the plane containing the projected 2D points from the 3D world and its intersection with the *optical axis* is called *principal point* R. It has its own two-dimensional coordinate system with axes U and V parallel to X and Y respectively.

The pinhole aperture of the camera O, through which all projection lines must pass, is assumed to be infinitely small.

Given a point $\boldsymbol{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, we denote as $\boldsymbol{Q} = (u, v)$ its projection into the *image plane* through the origin. The line connecting these two points is called *projection line*.

In order to understand the relationship between the coordinates of $\boldsymbol{Q}$ and $\boldsymbol{P}$, Figure 3.15 shows the 3D coordinate system from above, looking down in the negative direction of the Y axis. The *projection line* creates two similar triangles that allows us to state that

$$\frac{-u}{f} = \frac{x}{z} \implies u = \frac{-fx}{z} \tag{3.7}$$

Similarly, along the Y axis we have

$$\frac{-v}{f} = \frac{y}{z} \implies v = \frac{-fy}{z} \tag{3.8}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.9}$$

It is useful to adopt the *homogeneous coordinates* allowing the implementation of common transformations such as translation, rotation, scaling and perspective projection as matrix operations. Points P and Q can be refactored as:

$$\bar{\boldsymbol{P}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \bar{\boldsymbol{Q}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.10}$$

Rewriting (3.9) in terms of homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{f}{z} \begin{bmatrix} x \\ y \\ \frac{z}{f} \end{bmatrix} \sim \begin{bmatrix} x \\ y \\ \frac{z}{f} \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.12}$$

$$\bar{\boldsymbol{Q}} \sim \mathbf{C}\bar{\boldsymbol{P}} \tag{3.13}$$

where $\mathbf{C}$ is called *camera matrix*

It is possible to factorize the camera projection matrix as the multiplication of a 3x3 and a 3x4 matrices, which are respectively related to the intrinsic and extrinsic parameters. Intrinsic parameters model the optical features of the device and the extrinsic ones model the camera position and orientation in space.

From these 2D–3D correspondences, it is possible to calculate the 3D pose (rotation and translation) of the head, with respect to the camera, by way of the Point-n-Perspective (PnP) algorithm. The solving strategies estimate the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image.

The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. It is possible to write the full relationship between points on the image and the object as follows:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = s \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{3.14}$$

where equation (3.14) is made of a rotation and a projection matrix, governed by the camera intrinsic parameters (focal length f and center point C), mapping the 3D points to 2D image points, up to scale s. In our application set-up, it is possible to assume the camera is already calibrated so the intrinsic properties are already known (focal length, principal image point and skew parameter).

Given the intrinsic parameters we need to find 12 coefficients for the rotation and translation.

In this dissertation, the PnP algorithm used for the head pose estimation is the

Efficient PnP (EPnP) developed by Lepetit, et al. [86]. This solution is based on the assumption that each of the n points (which are called reference points) can be expressed as a weighted sum of four virtual control points.

From this point of view, the coordinates of these points become the unknowns of the problem. The estimation of these control points permits inferring the final pose of the camera.

3.4 Feature computation

The final step in the acquisition pipeline is the computation of the features that will be used by the classification model.

It is possible to divide these features in various categories:

- blink related;
- mouth related;
- driving style related.

The main idea is to create a set of features coming from different measurements in order to build an hybrid classification model that relies on different sources of information to predict the driver's drowsiness state.

Blink related

As explained before, the landmarks detection system returns a set of points, which permits computing a set of raw features called EAR. This describe the opening width of the eyes in a accurate and almost standardized way for each analyzed frame.

Initially we introduced, a threshold that discriminates whether subject's eyes are open or closed. This parameter was determined in an heuristic way from a set of experimental results in different conditions. The estimate parameter proved to work effectively for the majority of the analyzed subjects, but revealed unreliable for persons with a naturally-narrow eye aperture.

For this reason, we introduced a calibration phase, that will be explained later in Chapter 5. This routine performs a statistical analysis on the fetched EAR values determining an upper and lower bound to normalize the eye aspect ratio

in the range [0, 1]. More precisely, the value of the EAR feature was converted as follows

$$EAR_{standardized} = \frac{EAR_{current} - EAR_{min}}{EAR_{max} - EAR_{min}} \tag{3.15}$$

From this metric, it is possible to detect blinking from the following set of operations.

Whenever the current $EAR_{standardized}$ value is below the threshold, the eye is classified as closed. If this happens, the current frame index and the value are stored in the array *instant_blink*, which represent all the blinking instants. When the condition is not verified, the algorithm checks whether the eyes in the previous analyzed frame were closed or not. If they were not closed, the driver is keeping his/her eyes opened and no actions are needed. Instead, if they were closed, a blink has just terminated. As a matter of fact, the standardized EAR features for each frame of the blink period can be computed using the function *compute_blink()* that will be later described. Finally the algorithm will add the output of the computation to the *blinks* array, which contains all the blinks occurred during the last minute.

Currently tested mobile devices are not capable of performing all the processing steps of the acquisition pipeline within the time between two consecutive frames (i.e. for a camera that acquire images at 30 frames per second, approximately 33 ms); so, some frames will be discarded by the application.

For this reason, the *compute_blink()* function has, in first place, to estimate the missing values from the discarded frames performing a linear interpolation.

Linear interpolation employs some linear polynomial functions to construct new data points within the range of a discrete set of known values.

Given two points with coordinates (x_0, y_0) and (x_1, y_1), for a value x in the interval (x_0, x_1), the correspondent interpolated y value is given from the equation:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \tag{3.16}$$

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{3.17}$$

After the interpolation, the algorithm computes the following features:

Frame ID	...	123	125	128	129	131	133	135	...
$EAR_{standardized}$	...	0.89	0.93	0.91	0.9	0.6	0.2	0.15	...

Table 3.3: Mocked EAR values array

Frame ID	...	123	124	125	126	127	128	129	130	131	132	133	134	135	...
Int. $EAR_{standardized}$	...	0.89	0.91	0.93	0.925	0.915	0.91	0.9	0.75	0.6	0.4	0.2	0.175	0.15	...

Table 3.4: Interpolated mocked EAR values array

- **Frame duration**, which represent the number of frames that elapsed between the beginning and the end of the blink

$$\text{Duration} = Frame\text{ID}[end] - Frame\text{ID}[start] \tag{3.18}$$

- **Amplitude**, which represent the EAR excursion from the upper value to the lower one

$$\text{Amplitude} = \frac{EAR[start] - 2EAR[bottom] + EAR[end]}{2} \tag{3.19}$$

- **Eye Opening Velocity (EOV)**, which represent how quickly the eye opens after the blink

$$\text{EOV} = \frac{EAR[end] - EAR[start]}{Frame\text{ID}[end] - Frame\text{ID}[start]} \tag{3.20}$$

- **Interframe**, which measures the number of frames since the last blink

$$\text{Interframe} = Frame\text{ID}[start] - Frame\text{ID}[end]_{last\_blink} \tag{3.21}$$

The main idea is to build a sequence of blinks (given the fact that normally an alert person blinks 20 times per minute) where each row contains the four blinks' features. This will provide a fixed vector of shape $(n, 4)$ containing the last n driver's blinks. This array of data is generated for classification every minute.

If during the last minute, the algorithm captures less than n blinks, the sequence will be padded with zeros.

Works like [87], [38] and [88] reported that one of the biggest challenges in using blink features for drowsiness detection is the difference in blinking pattern across individuals

So features should be normalized across different subjects performing a statistical evaluation during the calibration phase.

In the calibration phase, the program captures one minute of data in order to compute the mean and the standard deviation of each features. These two values will be used in the previous features computation to standardize all the subsequent blinks.

$$\text{StandardizedFeature}_n = \frac{\text{Feature}_n - \mu_n}{\sigma_n} \tag{3.22}$$

The standardized features will allow the network to work with normalized data and not be biased by different subjects behaviors.

Mouth related

Given the set of landmarks extracted by the algorithm at each frame, a raw feature called MAR describes the opening width of the mouth in a accurate and almost standardized way.

Unfortunately these type of features are not implemented in this dissertation, but will be included in future development.

Driving-style related

The last type of features that the application is computing, are the one related to the current driving behavior and they can be divided into three categories:

- attention feature;
- speed feature;
- steering wheel feature.

The fetching of these features has currently been implemented only for the first two types, since they can be extracted using sensors that are inside smartphones.

An attempt to compute an estimation of the steering wheel values using the accelerometer, gyroscope and magnetometer has been done but the accuracy of the estimates made them useless for the final classification task.

One of the future steps in the project is to implement a Bluetooth direct connection with Texa Care, a bluetooth OBD unit that is capable of reading values from

the various car's electoric control units and transmit them to the smartphone in a readable way.

Attention feature

In the previous step of the acquisition pipeline, the algorithm estimated the head pose of the driver and returned a series of angles that represent the face direction in three dimensional space, as we can see from Figure 3.16.

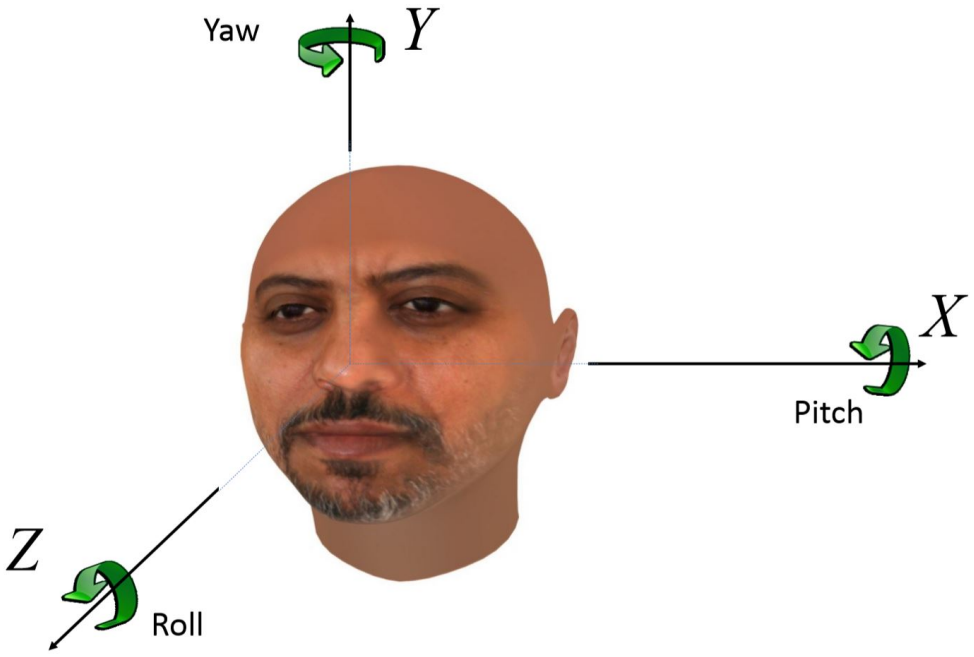


Figure 3.16: Euler's angles explaination (figure from [12])

We define the various angles as the following:

- Pitch is the angle obtained rotating the head around the X axis.
- Yaw is the angle obtained rotating the head around the Y axis.
- Roll is the angle obtained rotating the head around the Z axis.

During the initial calibration phase, the application requires the driver to place the smartphone in the typical place and look at the road. This will initialize the starting angles and the application will localize the camera in the space.

This operation will enable the estimation of the current pose for the head of the driver with respect to the road and the current camera position and orientation.

Speed feature

Some good indicator of the driver conditions, as stated in Chapter 2, are the type of road the driver is travelling and the difference between the current car speed and the actual speed limit on a determined road.

In order to grab these features, two raw data are necessary:

1. Current car velocity, which is computable from the inertial or localization sensors, as well as from the localizing applications that are installed on the smartphone.

2. Dataset of roads, which includes the GPS position, the type of road and, eventually, the road speed limit.

Modern devices running Android, are capable of achieving a very accurate position combining different localization apps or sensors (i.e. GPS, Network, Passive).

The information concerning speed limits can be provided by OpenStreetMap (OSM), a collaborative project to create a free editable map of the world.

OSM uses a topological data structure, with four core elements:

- Nodes, which are points with a geographic position. Stored as coordinates (pairs of a latitude and a longitude) according to World Geodetic System (WGS) 84. The WGS is a standard notation for cartography, geodesy, and satellite navigation including GPS;

- Ways, which are ordered lists of nodes, representing a polyline or possibly a polygon (when they form a closed loop). They are used both for representing linear features such as streets and rivers, and areas, like forests, parks, parking areas and lakes;

- Relations, which are ordered lists (called ”members”) of nodes, ways and other relations. Each member can optionally have a ”role”, represented using a string, which describes the scope of the object in the relation;

- Tag, which are key-value pairs (both arbitrary strings). They are used to store metadata about the map objects (such as their type, their name and their physical properties). Tags are not free-standing, but are always attached to an object: to a node, a way or a relation.

First of all it is necessary to download the data of the map. Due to network and server constraints, the algorithm download only the data belonging to an area of radius of 2 kilometers with center in the current position of the car.

When the map is available, the algorithm checks the accuracy of the current localization: if the accuracy is below 20 meters, the algorithm computes the possible roads where the driver may be.

The algorithm rely on the haversine formula to calculate the great-circle distance (i.e. the shortest distance over the earth's surface) between two GPS coordinates point.

$$a = \sin^2(\Delta\phi/2) + \cos\phi_1 \cos\phi_2 \sin^2(\Delta\lambda/2) \tag{3.23}$$

$$c = 2a \tan 2(\sqrt{a}, \sqrt{1-a}) \tag{3.24}$$

$$d = R \cdot c \tag{3.25}$$

where ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km) with all the angles expressed in radians.

The use of this formula allow the algorithm to compute the distance between the current location and all the nodes in the downloaded map.

Starting from the closest, the algorithm checks if the current position is between two consecutive points. In order to do that, it calculates an abstract rectangle generated expanding the two consecutive points in latitude and longitude terms, like in Figure 3.17.

We suppose to have two points expressed in GPS coordinates $p_1 = (\text{lat}_1, \text{lon}_1)$ and $p_2 = (\text{lat}_2, \text{lon}_2)$. It is possible to find the four corners of the rectangle performing the following steps:

1. find the slope and intercept of the line expressed as $l : y = mx + q$ passing

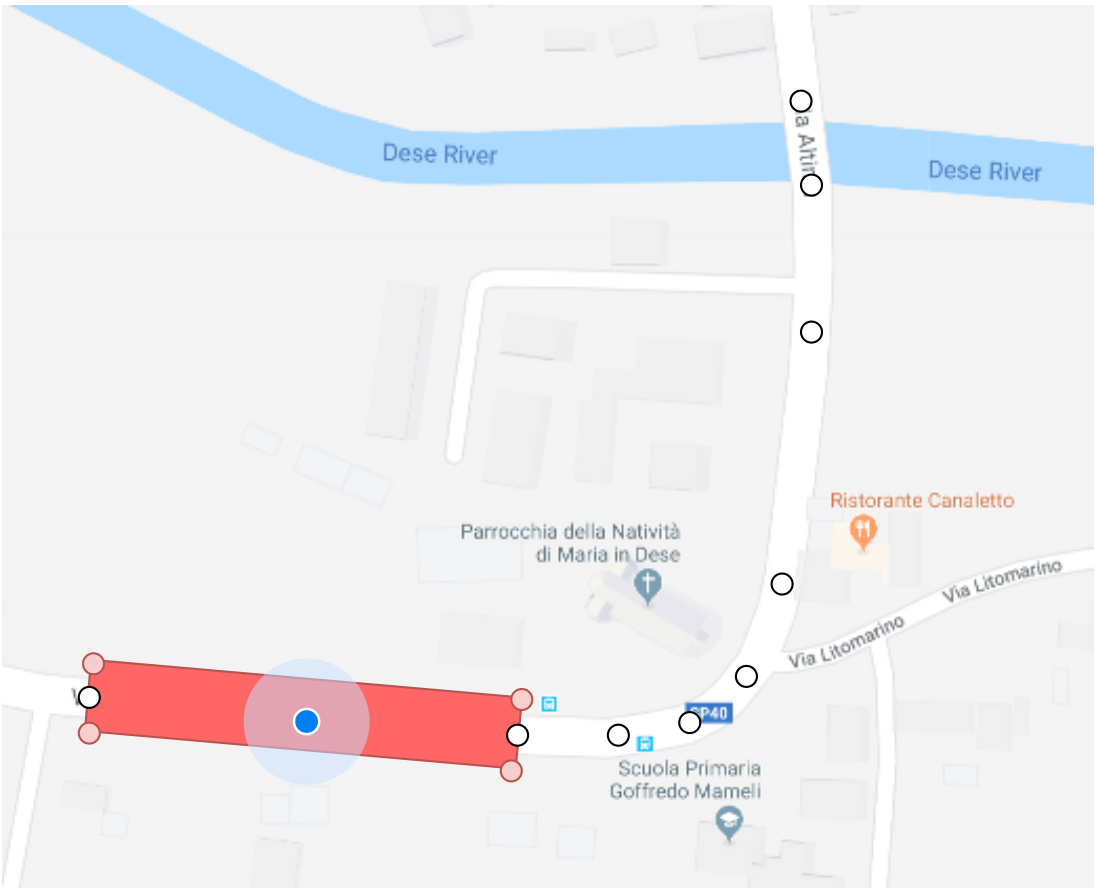


Figure 3.17: Road street algorithm representation

between the two points as follow:

$$m = \frac{\text{lat}_2 - \text{lat}_1}{\text{lon}_2 - \text{lon}_1} \tag{3.26}$$

$$q = \text{lon}_1 - m\text{lat}_1 \tag{3.27}$$

2. compute the slope and intercept of $l_\perp$ as

$$m_\perp = -\frac{1}{m} \tag{3.28}$$

$$q_\perp = \text{lon}_1 - m_\perp\text{lat}_1 \tag{3.29}$$

3. compute the longitudes of the four corners as

$$c_1 = (\text{lat}_1 + \epsilon, m_\perp(\text{lat}_1 + \epsilon) + q_\perp) \tag{3.30}$$

$$c_2 = (\text{lat}_1 - \epsilon, m_\perp(\text{lat}_1 - \epsilon) + q_\perp) \tag{3.31}$$

$$c_3 = (\text{lat}_2 + \epsilon, m_\perp(\text{lat}_2 + \epsilon) + q_\perp) \tag{3.32}$$

$$c_4 = (\text{lat}_2 - \epsilon, m_\perp(\text{lat}_2 - \epsilon) + q_\perp) \tag{3.33}$$

where ϵ is an arbitrary value that represent the width of a road lane (in our case $\epsilon = 7m = 0.00007$ in geographic coordinates)

After that, the slope and intercept of the four line passing for each rectangle corner points are calculated.

In order to determine if the current locations lies within the current rectangle, the algorithm checks whether latitude and longitude values are included withing the respective upper and lower bounds.

Every time the location provider gives a new GPS location, the algorithm estimate the new road path within the possible roads viable from the previous position.

This algorithm, despite the fact of being under heavy development, seems to return a pretty accurate estimate of the current road.

Given the current road, the relative speed limit is obtained looking at the *max-speed* tag or, if this is not present, using the speed limits in accordance to the law.

4

Classifiers

In the last section of the previous chapter, the set of features used by the classifier were presented. In this chapter a description of the training and test datasets will be given, focusing on the four models architecture and motivating the structure of layers that was chosen.

4.1 Training dataset

The first problem in the drowsiness detection task is the collection of an extensive and reliable dataset. To this purpose, a generalized and diversified dataset does not exist. This is because the acquisition of data that enable the computation of high-quality and and noiseless features is a complex problem.

Data acquisition process can be basically divided into four different types of setups, each of one differing from the other by two main features: controllability and validity [89].

The first one refers to the possibility to control every aspect of the acquisition process, like the speed to maintain, the track to travel and the current driver condition, which is affected by the driving scenario, the characteristics of the participants, their preparation before the study and their experience during and after the study.

The second one refers the possibility to generalize the results for real life situ-

ations. It implies having a high level of realism in the definition of the driving set-up, taking into account aspects as obtrusive/unobtrusive instrumentation, the driving scenario, and the appropriateness of the collected data for the classification task.

In most scientific work, a controlled experimental set-up allows minimizing the noise level and the probability of misunderstanding the acquired data. Control is often necessary for reliable repeatability within a reasonable time frame. In the end, a controlled and simulated scenario permit preserving the drivers' safety since falling asleep while driving on a road can lead to severe injuries for both drivers and people passing by.

- Driving simulators: middle to high fidelity simulators that provide the driver with an at least somewhat genuine feeling of sitting in a real car. The environment is computer generated, and it is possible to log a host of variables [89].
- Test tracks: performed using instrumented vehicles on a controlled environment, closed to public traffic, which can easily be adjusted to the desired needs [89]. Examples of drowsiness and also distraction related studies performed on test track include [90] work.
- Road tests: quite limited in time, the driven routes are predetermined, and there may be an experimenter in the car. However, the study is conducted in real traffic [89].
- Naturalistic studies: conducted in real traffic, but no experimenter is present in the car, and the studies are usually long-term, lasting for a month or more. The drivers have free choice of route and use the vehicles for their daily lives.

A simulator has a high degree of control, it is possible to control test conditions, dissuading and stimulating drowsiness. Reducing the need to change lanes and gear, ask drivers to drive at constant speed and not introducing environmental disturbances as cross winds are all examples of factors that stimulate drowsiness. However, in opposition, there is a lower degree of external validity (the drivers' behavior will not be the same under simulated driving since there is not any risk involved).

An experiment on a test track or on a road with an experimental vehicle still provides a high degree of control on the participants, but decreases in the control

of events like interactions with other road users, animals, but also of the weather and road constructions.

Naturalistic driving studies will have a low degree of control.For example, it is not possible to use electrodes to acquire physiologic signals, to use a subjective measure as the KSS and no prior information is available about the participants' conditions, as the number of hours slept. By contrast it has a high degree of external validity, as it depicts real life situations. Regardless of the experimental setting the quality of the results is highly dependent on the data quality.

4.1.1 UTA-RLDD dataset

The RLDD dataset was created for the task of multi-stage drowsiness detection, targeting not only extreme and easily visible cases, but also subtle cases of drowsiness and has been acquired using the first type of process.

Detection of these subtle cases can be important for detecting drowsiness at an early stage, so as to activate drowsiness prevention mechanisms.

The RLDD dataset is the largest to date realistic drowsiness dataset and consists of around 30 hours of RGB videos of 60 healthy participants.

For each participant there is one video for each of three different classes: alertness, low vigilance, and drowsiness based on the KSS table, for a total of 180 videos for around ten minutes each.

There were 51 men and 9 women, from different ethnicities (10 Caucasian, 5 non-white Hispanic, 30 Indo-Aryan and Dravidian, 8 Middle Eastern, and 7 East Asian) and ages (from 20 to 59 years old with a mean of 25 and standard deviation of 6). The subjects wore glasses in 21 of the 180 videos, and had considerable facial hair in 72 out of the 180 videos. Videos were taken from roughly different angles in different real-life environments and backgrounds.

Each video was self-recorded by the participant, using their cell phone or web camera. The frame rate was always less than 30 fps, which is representative of the frame rate expected of typical cameras used by the general population.

All videos were recorded in such an angle that both eyes were visible, and the camera was placed within a distance of one arm length from the subject. These instructions were used to make the videos similar to videos that would be obtained in a car, by phone placed in a phone holder on the dash of the car while driving.

Figure 4.1: Sample frames from the UTA-RLDD dataset in the alert (first row), low vigilant (second row) and drowsy (third row) states (figure from [13]).

Rating	Description
1	*Extremely alert*
2	*Very alert*
3	*Alert*
4	*Fairly alert*
5	*Neither alert nor sleepy*
6	*Some sign of sleepiness*
7	*Sleepy, but no effort to keep alert*
8	*Sleepy, some effort to keep alert*
9	*Very sleepy, great effort to keep alert, fighting sleep*

Table 4.1: Quantization of Karolinska Sleepiness Scale into three different classes

The three classes were explained to the participants as follows

1. **Alert:** One of the first three states highlighted in the KSS table in Table 4.1. Subjects were told that being alert meant they were experiencing no signs of sleepiness.

2. **Low Vigilant:** As stated in level 6 and 7 of Table 4.1, this state corresponds to subtle cases when some signs of sleepiness appear, or sleepiness is present but no effort to keep alert is required.

3. **Drowsy:** This state means that the subject needs to actively try to not fall asleep (level 8 and 9 in Table 4.1).

4.1.2 Feature extraction

Each of these videos has to be processed in order to compute all the features needed to train the classification models. A Python implementation of the acquisition framework described in Chapter 3, has been deployed with some modifications.

Fold	Subjects ID	Dataset size											
		win_len = 30			*win_len = 40*			*win_len = 50*			*win_len = 60*		
		training	**validation**	**test**	**training**	**validation**	**test**	**training**	**validation**	**test**	**training**	**validation**	**test**
1	01 - 12	9687	2421	2256	9621	2401	2248	9580	2388	2245	9561	2371	2240
2	13 - 24	8532	2133	2679	8520	2121	2658	8512	2109	2650	8502	2095	2639
3	25 - 36	9656	2413	2295	9648	2405	2287	9640	2397	2280	9618	2382	2268
4	37 - 48	9437	2359	2568	9411	2332	2543	9403	2325	2531	9395	2310	2515
5	49 - 60	8655	2193	3546	8643	2181	3534	8638	2176	3528	8630	2168	3520

Table 4.2: Dataset division

Given the fact that these videos could only provide a set of behavioural measures, a limited training dataset based only on the eyes features has been generated. A future objective in the project is to record a set of personal videos with subjects that are driving in real-world or in a simulator, that will comprise data coming from other type of measures.

After the extraction of the features, the algorithm generates, for every video, a batch of input arrays following the rules explained in Chapter 3 and containing the number of blinks detected in a minute, cycling all the blinks contained in the video with a stride of 2.

A parameters analysis has been performed, in order to choose the best shape of the input array to feed into the classification module. Model training and test has been done with different input shape of $(30, 4), (40, 4), (50, 4), (60, 4)$, where the first coordinate represent the maximum number of blinks that can be inserted in the input array during the acquisition phase.

In addition to that and following the rules imposed by the dataset's creators, all the classification models have been trained on a subset of subjects and tested on another one containing different subjects.

The division of the subjects and the size the training, validation and test sets is described in Table 4.2.

In addition to original labels given by the videos that allow a multiclass classification problem, a binary classification one has been also studied: given the dataset explained above, all the sequences labeled as the mid state of drowsiness were assigned at the alert one if the number of blinks detected in the sequence is less than 10, or at the drowsy one if the condition was not satisfied.

Trivially, the dimension of the dataset for the binary classification task has the same dimension of the one created for the multiclass problem.

4.2 Classifiers' Architectures

The model built for the abstract features classification task are three: the first two are based only on Convolutional layers and differs only by the numbers of fully connected layers after the convolutional part, the second one adds, after the convolutional part, a Bidirectional Long Short Term Memory (LSTM) layer, useful to avoid the long-term dependency problem, that is one of the problems that afflict RNN and that will be explained in the next subsection.

In order to optimize the networks, all the model uses the Adam version of stochastic gradient descent together.

4.2.1 LSTM networks

One of the main problems of the traditional neural networks when tackling some time-based classification, is the lack of information persistance during the analysis of the sequence. As a matter of fact, it is not clear how the information at a given instant could be affected by the data from the past.

In order to avoid this, recurrent neural networks (RNN) were created; their peculiarity is that the input of a RNN layer is made by the output of previous layers and the output of the current layer at previous instants. This recursive links create loops in the flowing of the information and require storing part of the information.

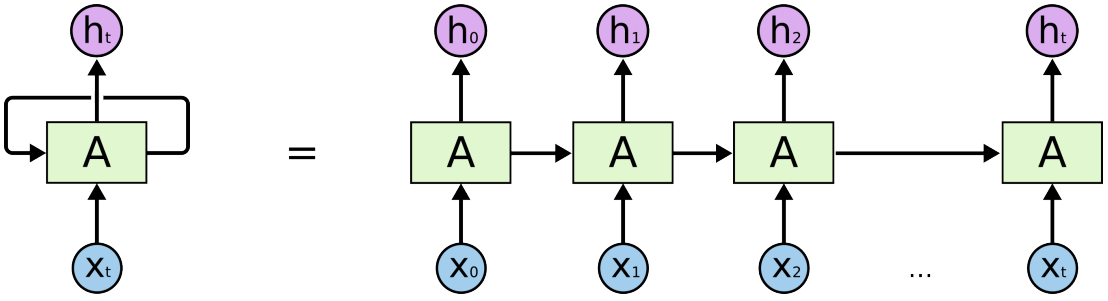
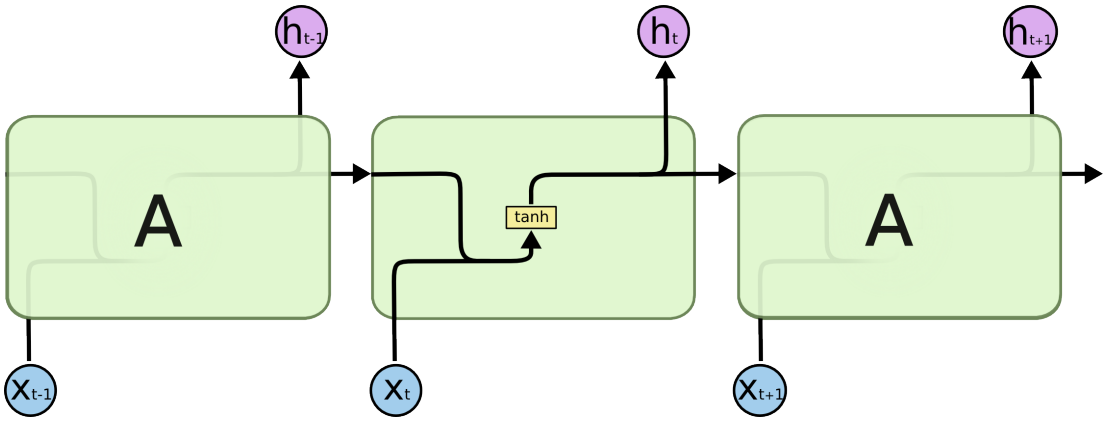


Figure 4.2: Recurrent neural network (figure from [14])

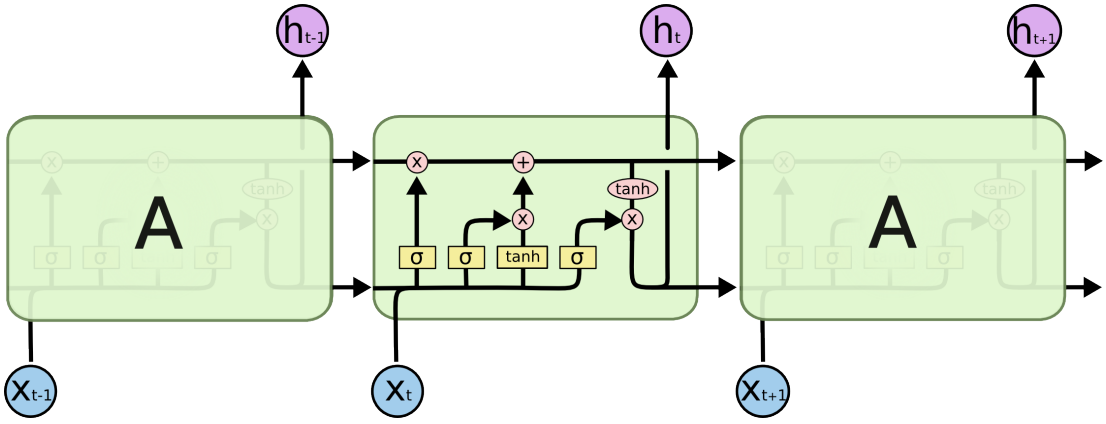
As it is possible to see in Figure 4.2 a chunk of neural network A looks at some input x_t and outputs a value h_t while the loop allows information to be shared between one step of the network and the next one. But these type of networks, as we stated before, are not able to handle the so-called **"long-term dependencies"**, that are the retrieval of relevant information for the current prediction that was learned a long time ago.

A special kind of RNNs that can easily tackle this problem are the **Long Short Term Memory networks**: these type of networks are nowadays used on a large variety of problems and are explicitly designed to remember information for long periods of time.

The main difference between a regular RNN and a LSTM network is in the repeating module A:



(a) RNN repeating module



(b) LSTM repeating module

Figure 4.3: Differences in the repeating modules (figures from [14])

As we can see from Figure 4.3, the standard RNN have a very simple structure with a single tanh layer, while the LSTM uses four interactive neural network layer where the horizontal line running through the top of the diagram represent the **cell state** (i.e. the key idea of LSTM).

The LSTM can change the cell state, adding or removing information to it, using **gates**:

1. The first one is called *forget gate layer* and is useful to decide what information we're going to throw away from the cell state using a sigmoid layer.
2. The second (called *input gate layer*) and the third one are useful to create an update to the state.
3. The third one is useful to decide what the network is going to output, basing the choice on the cell state.

4.2.2 A first drowsyness classifier based on CNN

The first classifier is a feedfoward network made of a set of convolutional layers that extract relevant features from the feature vector generated by the pipeline described in the chapter 3.

The convolutional part is a typical CNN composed by three consecutive two-dimensional convolutional layers with a ReLU activation function (chosen in order to avoid the vanishing gradient problem) and an increasing number of filters.

The first one has 30 filters and a window size of $(3, 4)$ in order to detect possible correlations between subsequent features.

The second one and the third one have a window size of $(3, 1)$ that searches correlations in the time domain and 60 and 120 filters respectively in order to augment the dimensionality of the network.

Every layer is followed by a MaxPooling operation in two dimension with pool size equal to $2, 1$ and a Dropout layer.

At each training stage, the Dropout layer drops individual nodes out of the net with probability $1 - p$ or keeps them with probability p, in order to build a reduced network where the incoming and outgoing edges to a dropped-out node are also removed.

This approach is useful in order to allow the network to generalize better the context it is trying to learn and to reduce overfitting.

The output of the last convolutional layer is flattened in order to feed it to a single fully connected layer containing 128 neurons.

The fully connected part is followed by another Dropout and finally by a set of 3 or 2 neurons, depending on the type of problem, that represent the probability of the input to belong to a certain class that have a Softmax activation function.

4.2.3 A drowsiness classifier based on CNN and Dense fully-connected layers

In the second model, the approach is the same as the previous one with one main difference: the introduction of a dense Fully Connected part at the end of the network.

The new part is composed by a set of three different fully connected layers, each one with decreasing number of neurons as it is possible to see in Figure 4.4.

This part has been inserted into the network because of its capability to map a lower dimension output as the one of the last convolutional layer into a greater one, in order to fetch new features the convolutional part has generated.

The convolutional part, as stated before, is exactly the same as the previous network.

4.2.4 A third drowsiness classifier based on LSTM

In this network, a different approach has been followed, introducing into the network a Bidirectional LSTM part capable of finding time correlations between the various timesteps.

The model has a **single Bidirectional LSTM hidden layer** with a number of neurons proportional to the window length. The idea of Bidirectional Recurrent Neural Networks (RNNs) is straightforward and it involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second.

Then, the layer is followed by a **Dropout layer** used to reduce overfitting during the training session and a **Dense fully connected layer**, with the same

number of neurons of the LSTM and a ReLU activation function, in order to interpret the features extracted by the LSTM hidden layer.

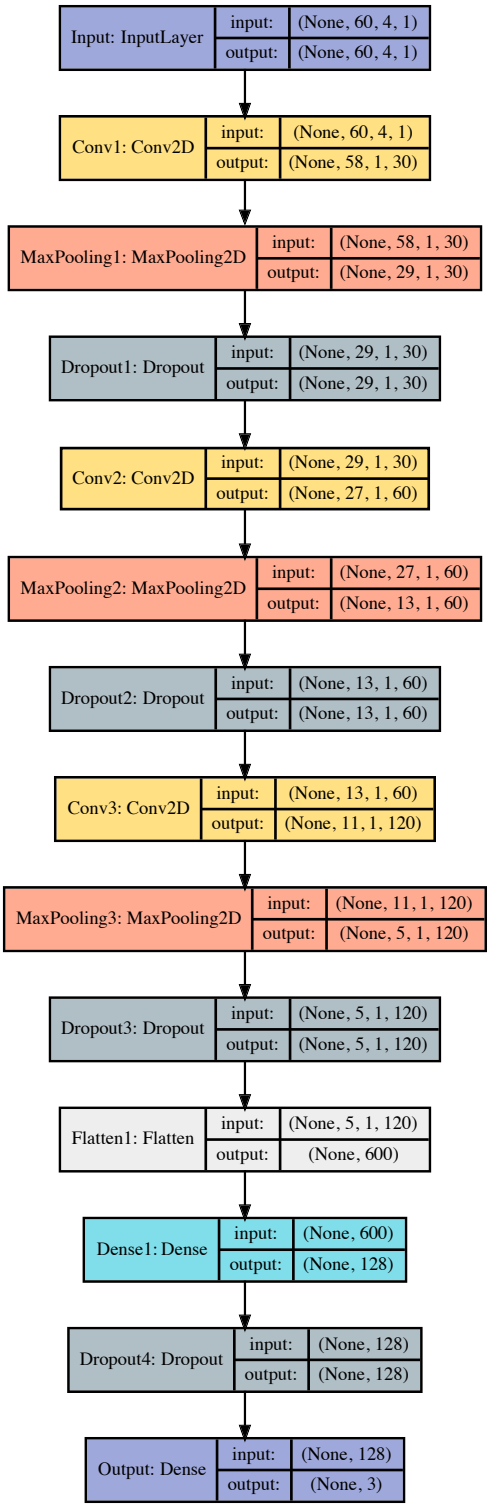
Finally another **Dense fully connected layer** with number of neurons equal to the classes size and a softmax activation function, in order to make predictions.

4.2.5 A fourth classifier inytegrating CNN with LSTM

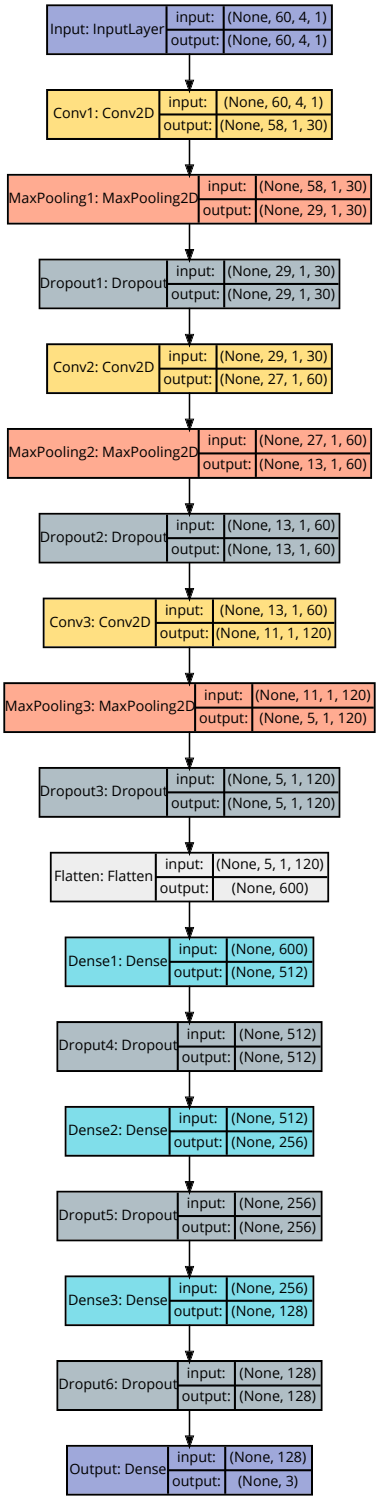
The last model merges the presented architectures into a unique one.

The convolutional part has been added before the LSTM layer, to read subsequences of the main time series as blocks, extract features from each block and then allow the LSTM to interpret the features extracted.

The rest of the network, including the LSTM part, is equal to the one presented in the previous network.

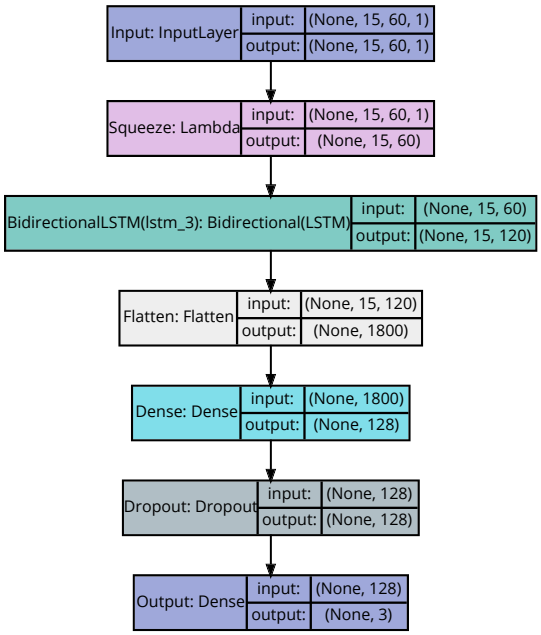


(a) First model: CNN

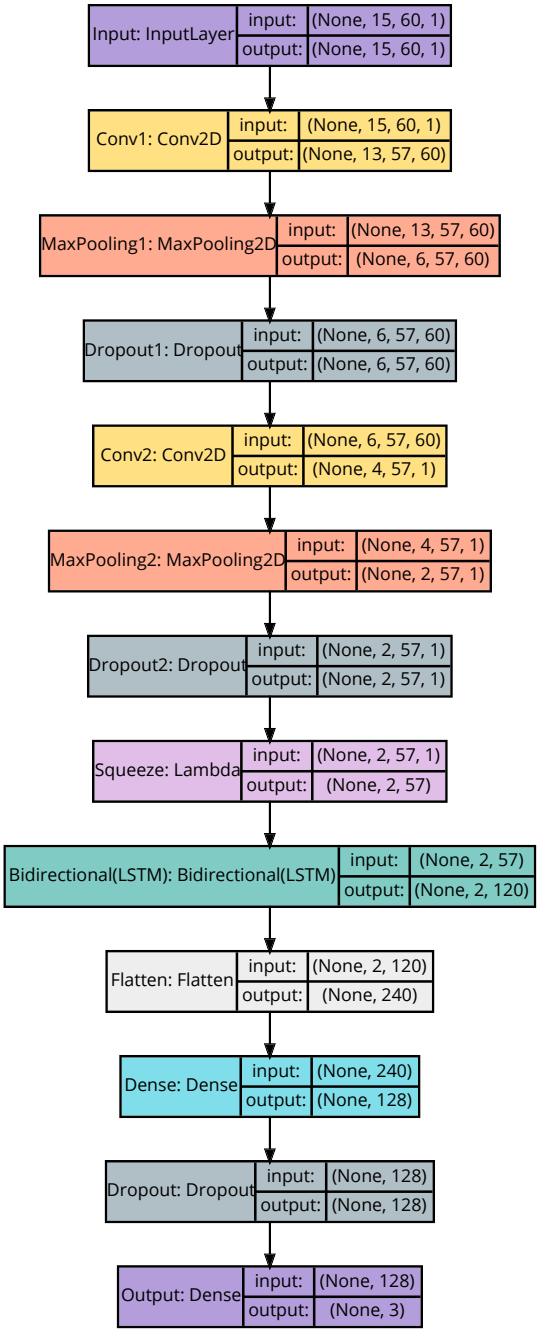


(b) Second model: CNN + Dense FC

Figure 4.4: First and second models architecture



(a) Third model: LSTM



(b) Fourth model: CNN + LSTM

Figure 4.5: Third and fourth models architecture

5 Results

5.1 Performance evaluation

In order to compare the different models and to choose the best performing parameters for the networks, a prediction quality measure has to be introduced.

The first thing to compute is the confusion matrix, a table that is used to describe the performance of each classifier on a set of test data for which ground truth values are known.

From the confusion matrix it is possible to extract different measures like

- accuracy, which represents the percentage of correctly-classified samples;
- precision, which represents percentage of true positive samples, i.e., drowsy states that are classified as ”drowsy”;
- recall, which represent the ratio between the total number of correctly classified positive examples divide by the total number of positive examples.

Given the fact that the drowsiness detection problem has been divided into two different sub problems, one regarding a binary classification and the other a three class classification, it is necessary to define the measures for each classification output.

For the binary classification problem, it is possible to represent the confusion matrix as in Table 5.1

		Predicted	
		Alert	*Drowsy*
Actual	*Alert*	TN	FP
	Drowsy	FN	TP

Table 5.1: Binary classification confusion matrix

and the various measures can be mathematically defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

$$\text{Precision} = \frac{TP}{TP + FN} \tag{5.2}$$

$$\text{Recall} = \frac{TP}{TP + FP} \tag{5.3}$$

Only in binary classification, recall of the positive class (i.e. drowsy) is also known as sensitivity, whilst recall of the negative class is called specificity.

For the multiclass classification problem, it is possible to represent the confusion matrix as in Table 5.2

		Predicted		
		Alert	*Mid*	*Drowsy*
Actual	*Alert*	$\text{value}_{0,0}$	$\text{value}_{1,0}$	$\text{value}_{2,0}$
	Mid	$\text{value}_{0,1}$	$\text{value}_{1,1}$	$\text{value}_{2,1}$
	Drowsy	$\text{value}_{0,2}$	$\text{value}_{1,2}$	$\text{value}_{2,2}$

Table 5.2: Multiclass classification confusion matrix

and the various measures, with n equal to the number of classes, can be mathematically defined as:

$$\text{Accuracy} = \frac{\sum_{i=0}^{n} value_{i,i}}{\sum_{i=0}^{n} \sum_{j=0}^{n} value_{i,j}} \tag{5.4}$$

For each class is also possible to describe the relative precision and recall:

$$\text{Precision}_i = \frac{value_{i,i}}{\sum_{j=0}^{n} value_{i,j}} \tag{5.5}$$

$$\text{Recall}_i = \frac{value_{i,i}}{\sum_{j=0}^{n} value_{j,i}} \tag{5.6}$$

In order to obtain a more general result, it is possible to average the results across classes:

$$\text{Average Precision} = \frac{\sum_{i=0}^{n} \text{Precision}_i}{\text{n}} \tag{5.7}$$

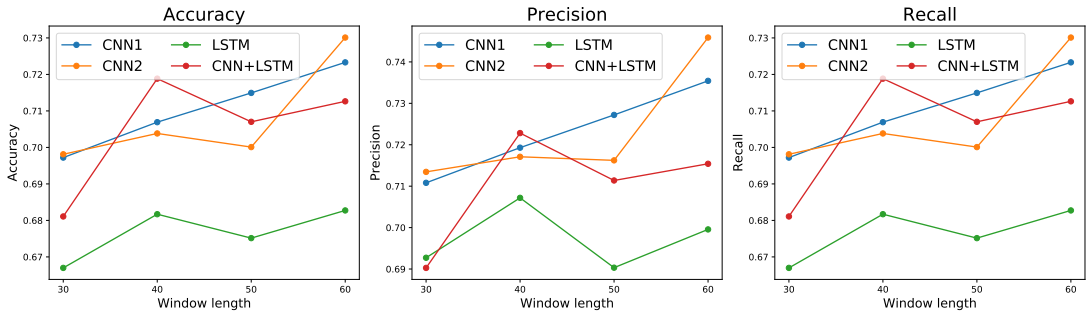
$$\text{Average Recall} = \frac{\sum_{i=0}^{n} \text{Recall}_i}{\text{n}} \tag{5.8}$$

5.2 Results report

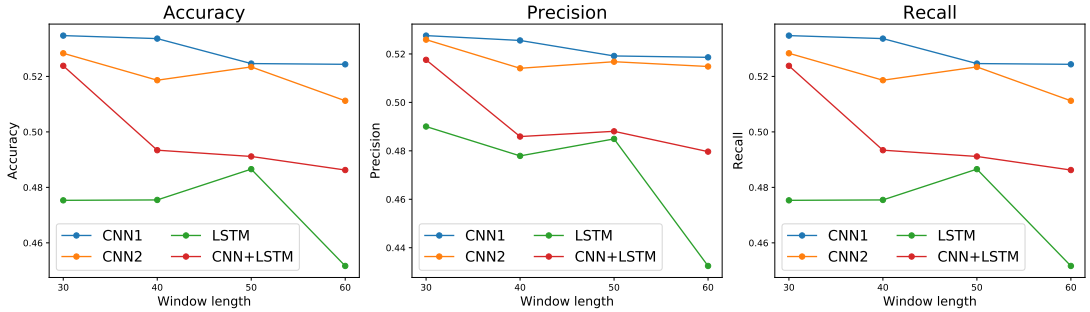
As it is possible to see in Figure 5.1, the obtained results are different in the binary classification problem and the multiclass problem. Surely the pure LSTM network performs worst in both cases and with every combination of window length, compared to the others. This means that the convolutional part is useful to create new features that are extremely-informative for the classifier.

In the binary classification results (see Figure 5.1a), a major improvement for all the networks can be seen with the increase of the window length. The best results, in fact, are achieved using a window length equals to 60 blinks, with the second CNN model achieving the best results. Both CNN1 and CNN+LSTM models perform similarly to CNN2 with the main difference that CNN+LSTM model seems to obtain better results with a window size equals to 40 blinks.

Looking at the confusion matrices (see Figure 5.2), it is clear that all networks are capable of predicting the alert state without any problem. It is more difficult to detect drowsiness, with a lot of false negatives predicted by the first three models. This could be caused by the division operated in the binary dataset creation. The only model that seems to be able to correctly detect different types of drowsiness is the CNN + LSTM, which unfortunately struggles a little bit in the detection of the alert state, producing more false positive than the others.



(a) Binary classification



(b) Multiclass classification

Figure 5.1: Performance measures of the classification models

In the multiclass classification results (see Figure 5.1b), the situation is inversed. CNN1, CNN2 and CNN+LSTM models perform better with a lower window length, degrading their performance as the size of the window increases. This is particularly evident in the CNN+LSTM model.

Looking at the confusion matrices (see Figure 5.4), it is possible to see how the CNN only models perform better than the LSTM-based ones. CNN1 especially is capable of differentiate well between the "Alert" and the "Drowsy" state.

All the networks struggles in the classification of "Mid" states as to be expected. Looking at videos belonging to that class, it is very difficult to classify the subject state even by a human judge (this was also stated in [13]).

(a) Window length = 30 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(b) Window length = 40 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(c) Window length = 50 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(d) Window length = 60 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

Figure 5.2: Binary classification confusion matrices

(a) Window length = 30 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(b) Window length = 40 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(a) Window length = 50 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

(b) Window length = 60 (from left to right: CNN1, CNN2, LSTM, CNN+LSTM)

Figure 5.4: Multiclass classification confusion matrices

6

Android Application

The presented classifier was implemented in an Android application that is supposed to run on drivers' cellphone. the current chapter overviews the different features of such applications.

6.1 Main components

There are some necessary building blocks that an Android application consists of. These loosely coupled components are bound by the application manifest file which contains description of each component and how they interact. The manifest file also contains the app's metadata, its hardware configuration and platform requirements, external libraries and required permissions.

It is possible to subdivide an Android application into four main components:

- **Activities:** they dictate the UI and handle the user interaction to the smart phone screen.
- **Services:** they handle background processing associated with an application.
- **Broadcast Receivers:** they handle communication between Android OS and applications.
- **Content Providers:** they handle data and database management issues.

There are additional components which will be used in the construction of above mentioned entities, their logic and wiring between them. These components are:

- **Fragments:** they represents a portion of user interface in an Activity.
- **Views:** they represents UI elements that are drawn on-screen including buttons, lists forms etc.
- **Layouts:** they represents view hierarchies that control screen format and appearance of the views.
- **Intents:** they represents messages wiring components together.
- **Resources:** they represents external elements, such as strings, constants and drawable pictures.
- **Manifest:** configuration file for the application.

6.2 Application architecture

The application was developed using an architectural pattern called Clean Architecture that is based on different other architecture proposed in the last years like

- Hexagonal Architecture
- Onion Architecture
- Screaming Architecture

The key concept in these architectures is the separation of concerns, i.e., the division of software into modules. Each one has at least one layer for business rules, and another for interfaces.

The actuation of the separation of concerns leads to the development of systems that present the following qualities:

- **Independence of Frameworks.** The architecture does not depend on the existence of some external library.

- **Testability.** The business rules can be tested without the UI, Database, Web Server, or any other external element.
- **Independence of UI:** the UI can be changed easily, without applying changes to the rest of the system. A Web UI could be replaced with a console UI, for example, without changing the business rules.
- **Independence of Database.** The database can be updated flexibly without modifying the rest of the system. A SQL database could be replaced with a non-relational one, for example, without changing the business rules.
- **Independence of any external agency.** The business rules don't know anything at all about the outside world.

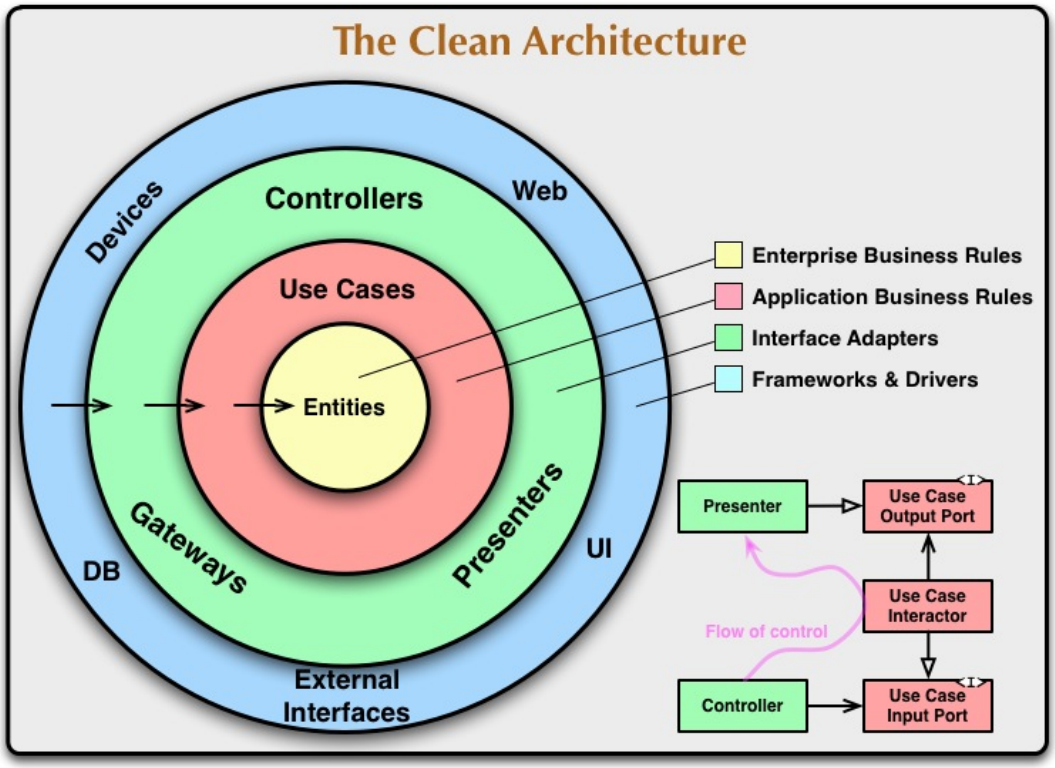


Figure 6.1: Clean architecture diagram from [15]

Figure 6.1 represents architecture and functionalities for an Android app.

The concentric circles represent different areas of software. As we move inwards, the level of abstraction increases and encapsulates higher level policies. The

outermost circle is low level concrete detail. The inner most circle is the most general.

The Dependencies Rule that lies behind this architecture state that source code dependencies can point inwards only. An inner circle does not have access to the variables and the states of an outer circle. In particular, the name of something declared in an outer circle must not be mentioned by the code in the an inner circle. That includes, functions, classes. variables, or any other named software entity.

By the same token, data formats used in an outer circle should not be used by an inner circle, especially if those formats are generate by a framework in an outer circle.

In the lower right of Figure 6.1, there is an example of how the crossing of the circle boundaries is performed.

It is possible to identify the following fundamental components:

- **Entities**, which encapsulate enterprise wide business rules (i.e. the most general and high-level rules of the application). They are the least likely to change when something external changes and no operational change to any particular application should affect the entity layer.

- **Use Cases**, which encapsulate application specific business rules implementing all of the use cases of the system. These use cases orchestrate the flow of data to and from the entities, and direct those entities to use their enterprise wide business rules to achieve the goals of the use case. This layer has not to be affected by changes to coming from outer world such as the database, the user interface (UI), or any of the common frameworks.

- **Interface Adapters**, which contains set of adapters that adapt data from a format suitable for use cases and entities, to a format to be used by some external agency, like the database or the Web. The Presenters, Views, and Controllers can be found here.

- **Frameworks and Drivers**, which is generally composed of frameworks and tools such as the Database, the Web Framework, or, like in our case, the Android environment. All the

6.3 Application organization

Following the Clean Architecture structure, the application is organized in features, representing, as the name suggest, different functionality of the application.

- Dashboard: the main activity where a general overview of the entire system is rendered.
- Calibration: the activity that is responsible to perform the calibration phase for each user.
- Alert sessions: the activity that is responsible to show all the sessions recorded in the phone and on the cloud.
- Classification models: the activity that is responsible to illustrate the selected classification model and all the calibrations done for that model.
- Settings

Each feature is divided into three layers:

- Domain layer: containing all the classes belonging to the core business logic of the application (i.e. entities, use cases, interfaces for the data providers and for the computation workers)
- Data layer: containing the implementations of all the interfaces described in the domain part.
- Presentation layer: containing all the logic for the connection between the inner layers and the user-world. It follows a Model-View-ViewModel (MVVM) architecture.

The application is composed by several classes in order to follow the principles explained by the Clean Architecture and the scope of this dissertation is not to focus on each class. For this reason, only a review of the main feature will be given, due to the fact that is composed by all the components that were explained before.

6.3.1 Dashboard

The application, as stated before, includes multiple modules in order to grab information, preform the computation, store data and render the results; each one of these actions is represented by a different type of module:

- Service modules. These are needed in order to grab information by the sensors inside the smartphone. Frame provider module, location provider module and data vehicle module belong to this class.
- Worker modules. These are needed to perform computation on input data in order to extract data from it. A clear example is the visual computation module that is responsible to compute the raw features from an input frame.
- Repository modules. These are needed to store data on local, cloud and download data from the web. Examples are the sessions recording repository that save a driving session (composed by a video and the related data) either in local and on the personal cloud and the map repository module that downloads maps from a determinate source.

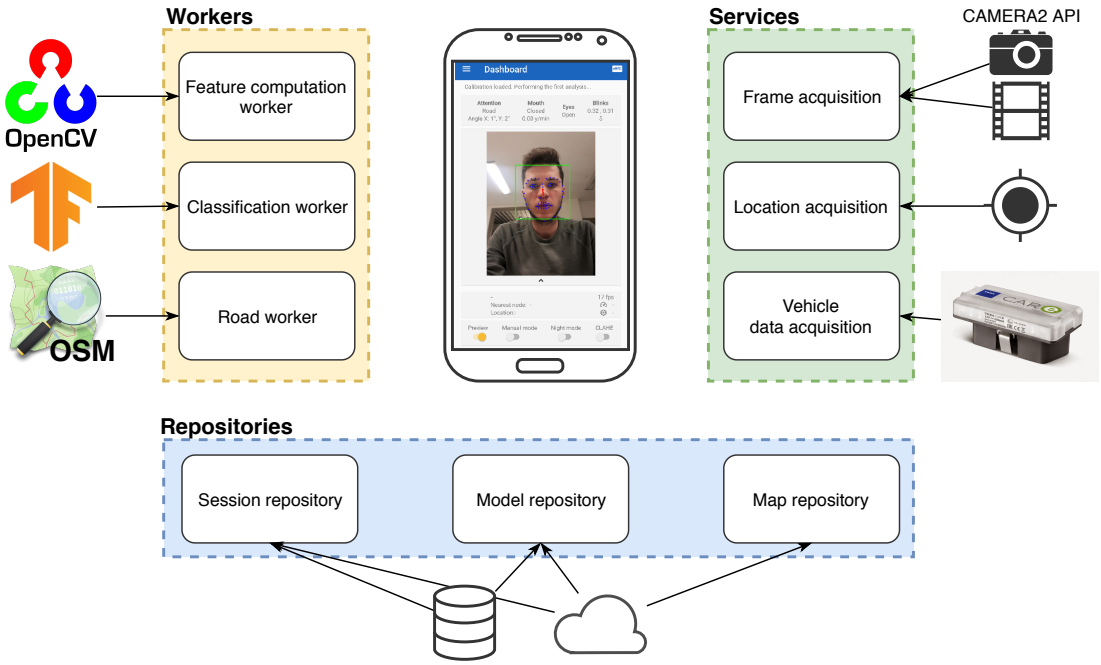


Figure 6.2: Dashboard modules overview with relative implementations

The logic business of these modules is defined in the domain layer of the relative features using an interface that is responsible to declare every function the module has to implement.

Relative implementations of these interfaces are, instead, placed into the data layer of the relative feature.

Using this approach is possible to separate the core business logic of the application from the relative implementation as the Clean Architecture states. In fact, only classes that belong to the data layer have information about classes that are in a inner layer but the ones in the domain layer do not know anything about outer circles.

The practical utility of this architecture is that is possible to swap an old implementation of a module with a new one without having to modify any code in the core business logic of the application.

Another application of this concept is given by the frame acquisition module. In the current application there are two different implementations of this functionality. The first acquires frames from the smartphone camera, the other one reads frames from a video file.

Having declared a generic frame acquisition module, the core business logic does not care about the origin of these frames. This implies that if, in a future implementation, the application need to process video frames from a connected external camera in a remote place, it is only necessary to implement the business rules declared by the interface that describe the module.

As it is possible to see in Figure 6.2, the dashboard is the control center of the application and includes the majority of the modules declared in the business rule.

The Android activity lifecycle (see Figure 6.3) defines a set of callbacks that allow the activity to know that a state has changed. As an example, the system is creating, stopping, or resuming an activity, or destroying the process where the activity resides. The dashboard pipeline can be described as:

1. Initialization: a background process loads all the modules needed by the activity and load the various models (i.e. face detector, landmarks detector, classification model).

2. Start of the acquisition: a start command is given in the *onResume()* function to all the service modules in order to start reading information.

3. Computation: every time the application receive a data entity (i.e. a frame, a location) from a service, the dashboard ViewModel send it to the relative worker in order to obtain a result.

4. Rendering: when a result is received by the ViewModel, an update of a MutableLiveData declared in the ViewModel is performed. This triggers a callback in the Fragment that calls the relative render method.

5. User interaction: different event listeners are declared in the Fragment; when a user interact with the UI, the relative procedure is started. These procedures call a method in the ViewModel in order to perform the needed operation (e.g. changing the ISO value of the camera, selecting the night mode).

6. Pause of the application: when the activity enters in the Pause state, the application pause all the modules that are working in background. This allow, for example, to not continue to read frames from the camera when the user is using another application.

7. Application closing: when the user decides to close the application, tha activity enters in the Stop and Destroy states, where all the data and the modules are destroyed.

As it is possible to see in Figure 6.2, the implementations of the modules are performed using a set of libraries. For example, the visual features computation module is implemented using OpenCV, the classification module uses Tensorflow and the map downloader module using RetroFit.

6.3.2 OpenCV

OpenCV (Open source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is developed using C++ but Python and Java bindings are provided, running on various desktop operating systems such as Windows, Linux, macOS, FreeBSD and on mobile systems like Android and iOS.

The main modules that are used in this dissertation are:

- **Imgproc**: containing various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms.

- **Calib3d**: containing algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction.
- **Features2d**: containing algorithms regarding the world of feature detection and description.
- **DNN**: containing methods and classes useful to perform inference of neural networks in an easy way
- **Face**: included in the *contrib* module (i.e. the container of all the features that are not officially supported) that contains some implementations of face landmarks detection algorithms.

The use of OpenCV allows the Android application and the various methods to run at very high speed respect to a naive implementation of the same method using Kotlin, due to the fact that the core of the framework, as said before, is written in C++, one of the most performing programming languages.

6.3.3 Tensorflow Lite

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

- The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
- The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

TensorFlow Lite is designed to make it easy to perform machine learning on devices, "at the edge" of the network, instead of sending data back and forth from a server, allowing to improve:

- Latency: there's no round-trip to a server
- Privacy: no data needs to leave the device

- Connectivity: an Internet connection isn't required
- Power consumption: network connections are power hungry

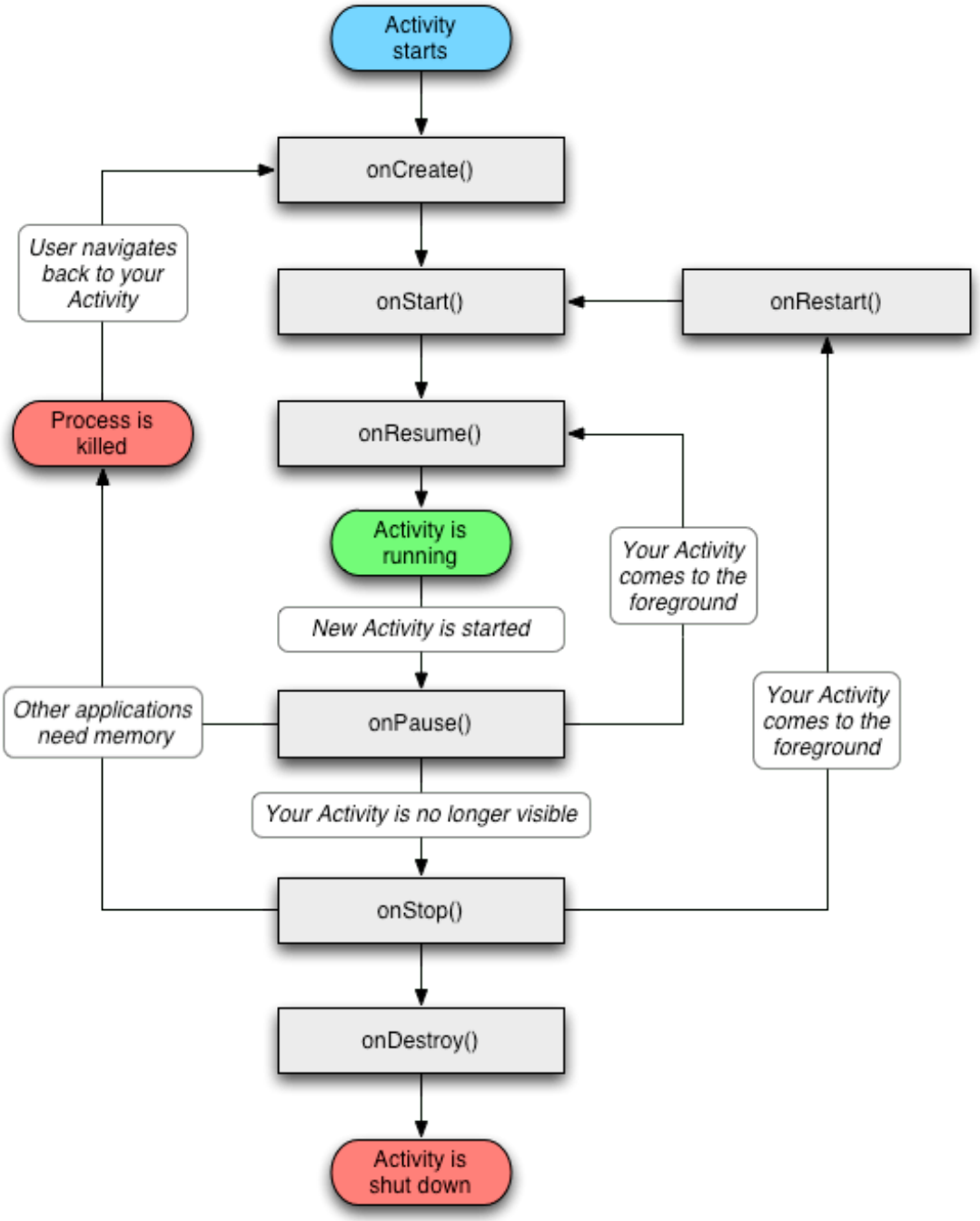


Figure 6.3: Android activity lifecycle flow

7

Conclusions and future work

This thesis described the creation process of a drowsiness detection system working on mobile devices with low computational power, starting from the literature study to the mobile application development.

I developed the system during my traineeship period from March to August 2019 at TEXA s.p.a., a company that is specialists in autodiagnostics, A/C system maintenance, satellite tracking and training courses for vehicle mechanics.

Driver drowsiness presents as an actual and relevant problem that is the cause of many driving accidents. The number of causalities due to drowsy driving accidents is high and has been constant over the last years with associated high economic impact. The high impact of drowsy driving related issues results in a strong motivation for developing a countermeasure for this problem.

After a study on the drowsiness measures currently pros and limitations and a comparison on the different products currently in the market, the feature acquisition phase was described, focusing on the research of speed optimization needed to run efficiently on mobile devices. The acquisition algorithm has been developed to enable the computation of different types of features from different data source in order to create a novel hybrid method.

The computation of a set of abstract features were presented next, together with the algorithms developed to compute them and the integration with OpenCV.

A set of four different networks was presented, relying on different architec-

tures (Convolutional Neural Networks and Recurrent Neural Networks) and the correspondent training results were discussed.

Results showed that drowsiness detection is still a difficult task, due to the lack of a uniform and well-labeled dataset and the difficulty to differentiate small changes between states (e.g. between Mid and Drowsy state).

Despite these problems, the obtained results look promising for future implementations of new classification models based on a larger set of features, as that the best performing classifier was able to differentiate between an Alert and a Drowsy state with high accuracy. Surely new features are needed to improve the performance in the multiclass classification task.

Finally the Android application was presented together with the architecture that was used to build the app structure. Following this coding style allowed to create a scalable, modular and testable application, which can be easily mantained in the future.

7.1 Future work

As future work, it will be interesting to implement new features extracted from the driver into the classification models, in order to improve their accuracy.

Moreover, the creation of a dataset, initially based on a simulator, could also be a key factor in improving the classification accuracy.

Finally, the implementation of the software into embedded devices created with machine learning purpose and the addition of an infrared camera to the system could also be relevant developments for the product success.

References

[1] Statista. Number of smartphone users worldwide from 2016 to 2021 (in billions). [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] Wikipedia. Rapid eye movement sleep. [Online]. Available: https://en.wikipedia.org/wiki/Rapid_eye_movement_sleep

[3] Mercedes-Benz. Attention assist. [Online]. Available: http://assets.mbusa.com/vcm/MB/DigitalAssets/TechnologyVideos/12_TV_ATTENTION-ASSIST_@2x.jpg

[4] M. Authority. Bosch drowsiness detection system to make alertness tech more common? [Online]. Available: https://www.motorauthority.com/news/1075780_ibms-lithium-air-battery-tech-the-500-mile-electric-car

[5] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[6] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] J. Rao, Y. Qiao, F. Ren, J. Wang, and Q. Du, "A Mobile Outdoor Augmented Reality Method Combining Deep Learning Object Detection and Spatial Relationships for Geovisualization," *Sensors (Basel)*, vol. 17, no. 9, Aug 2017.

[8] I. Mansouri. Computer vision part 5: Object detection, when image classification just doesn't cut it. [Online]. Available: https://medium.com/overture-ai/part-5-object-detection-when-image-classification-just-doesnt-cut-it-b4072fb1a03d

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," 2016, to appear. [Online]. Available: http://arxiv.org/abs/1512.02325

[10] G. N. O. M. K. Radha, "Empirical study of artificial neural networks in face-recognition," *IJCSN Journal*, vol. 8, pp. 64–72, 02 2019.

[11] M. Pandey, K. Chaudhari, R. Kumar, A. Shinde, D. Totla, and N. D. Mali, "Assistance for paralyzed patient using eye motion detection," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug 2018, pp. 1–5.

[12] A. Saeed, A. Al-Hamadi, and A. Ghoneim, "Head pose estimation on top of haar-like face detection: A study using the kinect sensor," *Sensors (Basel, Switzerland)*, vol. 15, pp. 20 945–20 966, 09 2015.

[13] R. Ghoddoosian, M. Galib, and V. Athitsos, "A realistic dataset and baseline temporal model for early drowsiness detection," *CoRR*, vol. abs/1904.07312, 2019. [Online]. Available: http://arxiv.org/abs/1904.07312

[14] C. Olah. Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[15] R. C. Martin. The clean architecture. [Online]. Available: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[16] N. H. T. S. Administration. (2017) Drowsy driving reasearch. [Online]. Available: https://www.nhtsa.gov/risky-driving/drowsy-driving

[17] D. Strayer, F. Drews, and D. Crouch, "Fatal distraction?: A comparison of the cell-phone driver and the drunk driver," *Briem and Hedman*, vol. 25, 01 1995.

[18] J. Higgins, J. Michael, R. Austin, T. Åkerstedt, H. Dongen, N. Watson, C. Czeisler, A. Pack, and M. Rosekind, "Asleep at the wheel—the road to addressing drowsy driving," *Sleep*, vol. 40, 02 2017.

[19] N. H. T. S. Administration. Automated vehicles safety. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[20] N. S. Foundation. Why do we need sleep? [Online]. Available: https://www.sleepfoundation.org/articles/why-do-we-need-sleep

[21] ——. What is circadian rhythm? [Online]. Available: https://www.sleepfoundation.org/articles/what-circadian-rhythm

[22] M. Johns, "Rethinking the assessment of sleepiness," *Sleep Med Rev*, vol. 2, no. 1, pp. 3–15, Feb 1998.

[23] N. S. Foundation. Drowsy driving. facts. [Online]. Available: https://drowsydriving.org/about/

[24] D. F. DINGES, "An overview of sleepiness and accidents," *Journal of Sleep Research*, vol. 4, no. s2, pp. 4–14, 1995. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2869.1995.tb00220.x

[25] C. Fors, C. Ahlstrom, V. Ahlström, S. Sörner, J. Eye, V. Kovaceva, E. Cars, S. Hasselberg, M. Eye, S. Krantz, J.-F. Eye, V. Grönvall, K. Cars, K. Kircher, A. Vti, A. Anund, Vti, and M. Krantz, "Camera-based sleepiness detection," 11 2019.

[26] J. F. May and C. L. Baldwin, "Driver fatigue: The importance of identifying causal factors of fatigue when considering detection and countermeasure technologies," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 12, no. 3, pp. 218 – 224, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1369847808001009

[27] P. A. H. P. A. Desmond, "Active and passive fatigue states," *Hum. factors Transp. Stress. Workload. fatigue*, vol. 40, pp. 455–465, 2001.

[28] P. Gimeno, G. Pastor, and M. Choliz, "On the concept and measurement of driver drowsiness, fatigue and inattention: Implications for countermeasures," *Int. J. of Vehicle Design*, vol. 42, pp. 67 – 86, 01 2006.

[29] A. Chowdhury, R. Shankaran, M. Kavakli, and M. Haque, "Sensor applications and physiological features in drivers' drowsiness detection: A review," *IEEE Sensors Journal*, vol. PP, pp. 1–1, 02 2018.

[30] H.-B. Kang, "Various approaches for driver and driving behavior monitoring: A review," 12 2013, pp. 616–623.

[31] O. M. A. Colic and B. Furht, *Driver Drowsiness Detection Systems and Solutions*, 2014.

[32] C. Liu, S. Hosking, and M. Lenné, "Predicting driver drowsiness using vehicle measures: Recent insights and future challenges," *Journal of safety research*, vol. 40, pp. 239–45, 08 2009.

[33] K. Kaida, M. Takahashi, T. Åkerstedt, A. Nakata, Y. Otsuka, T. Haratani, and K. Fukasawa, "Validation of the karolinska sleepiness scale against performance and eeg variables," *Clinical Neurophysiology*, vol. 117, no. 7, pp. 1574 – 1581, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1388245706001428

[34] T. E. S. Scale. About the ess. [Online]. Available: https://epworthsleepinessscale.com/about-the-ess/

[35] M. Hendra, D. Kurniawan, R. V. Chrismiantari, T. P. Utomo, and N. Nuryani, "Drowsiness detection using heart rate variability analysis based on microcontroller unit," *Journal of Physics: Conference Series*, vol. 1153, p. 012047, feb 2019. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F1153%2F1%2F012047

[36] H. D. Rosario, J. S. Solaz, N. Rodríguez, and L. M. Bergasa, "Controlled inducement and measurement of drowsiness in a driving simulator," *IET Intelligent Transport Systems*, vol. 4, no. 4, pp. 280–288, December 2010.

[37] A. Mashko, "Review of approaches to the problem of driver fatigue and drowsiness," in *2015 Smart Cities Symposium Prague (SCSP)*, June 2015, pp. 1–5.

[38] R. Feng, G. Zhang, and B. Cheng, "An on-board system for detecting driver drowsiness based on multi-sensor data fusion using dempster-shafer theory," in *2009 International Conference on Networking, Sensing and Control*, March 2009, pp. 897–902.

[39] S. H. Fairclough and R. Graham, "Impairment of driving performance caused by sleep deprivation or alcohol: a comparative study," *Hum Factors*, vol. 41, no. 1, pp. 118–128, Mar 1999.

[40] S. Otmani, T. Pebayle, J. Roge, and A. Muzet, "Effect of driving duration and partial sleep deprivation on subsequent alertness and performance of car drivers," *Physiol. Behav.*, vol. 84, no. 5, pp. 715–724, Apr 2005.

[41] A. Sahayadhas, K. Sundaraj, and M. Murugappan, "Detecting driver drowsiness based on sensors: a review," *Sensors (Basel)*, vol. 12, no. 12, pp. 16 937–16 953, Dec 2012.

[42] M. Ingre, T. Akerstedt, B. Peters, A. Anund, and G. Kecklund, "Subjective sleepiness, simulated driving performance and blink duration: examining individual differences," *J Sleep Res*, vol. 15, no. 1, pp. 47–53, Mar 2006.

[43] P. Choudhary and N. R. Velaga, "Analysis of vehicle-based lateral performance measures during distracted driving due to phone use," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 44, pp. 120 – 133, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1369847816302273

[44] B. G. Pratama, I. Ardiyanto, and T. B. Adji, "A review on driver drowsiness based on image, bio-signal, and driver behavior," in *2017 3rd International Conference on Science and Technology - Computer (ICST)*, July 2017, pp. 70–75.

[45] J. Schmidt, R. Laarousi, W. Stolzmann, and K. Karrer-Gauss, "Eye blink detection for different driver states in conditionally automated driving and manual driving using EOG and a driver camera," *Behav Res Methods*, vol. 50, no. 3, pp. 1088–1101, 06 2018.

[46] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-Time Imaging*, vol. 8, no. 5, pp. 357 – 377, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077201402902792

[47] M. Bn, "Facial features monitoring for real time drowsiness detection," 11 2016.

[48] I. G. Daza, N. Hern, L. M. Bergasa, I. Parra, J. J. Yebes, M. Gavilan, R. Quintero, D. F. Llorca, and M. A. Sotelo, "Drowsiness monitoring based on driver and driving data fusion," in *in Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, 2011, pp. 1199–1204.

[49] S. Darshana, D. Fernando, S. Jayawardena, S. Wickramanayake, and C. DeSilva, "Efficient perclos and gaze measurement methodologies to estimate driver attention in real time," in *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, Jan 2014, pp. 289–294.

[50] Jian-Feng Xie, Mei Xie, and Wei Zhu, "Driver fatigue detection based on head gesture and perclos," in *2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, Dec 2012, pp. 128–131.

[51] L. K. McIntire, R. A. McKinley, C. Goodyear, and J. Nelson, "A comparison of the effects of transcranial direct current stimulation and caffeine on vigilance and cognitive performance during extended wakefulness," *Brain Stimul*, vol. 7, no. 4, pp. 499–507, 2014.

[52] U. Svensson, "Blink behaviour based drowsiness detection : method development and validation /," 01 2004.

[53] F. Friedrichs and B. Yang, "Drowsiness monitoring by steering and lane data based features under real driving conditions," *European Signal Processing Conference*, 01 2010.

[54] M. Jackson, S. Raj, R. Croft, A. Hayley, L. Downey, G. Kennedy, and M. Howard, "Slow eyelid closure as a measure of driver drowsiness and its relationship to performance," *Traffic Injury Prevention*, vol. 17, 05 2015.

[55] W. Han, Y. Yang, G.-B. Huang, O. Sourina, F. Klanner, and C. Denk, "Driver drowsiness detection based on novel eye openness recognition method and unsupervised feature learning," 10 2015.

[56] A. Rumagit, I. Akbar, M. Utsunomiya, T. Morie, and T. Igasaki, "Gazing as actual parameter for drowsiness assessment in driving simulators," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, pp. 170–178, 01 2019.

[57] C. J. de Naurois, C. Bourdin, A. Stratulat, E. Diaz, and J.-L. Vercher, "Detection and prediction of driver drowsiness using artificial neural network models," *Accident Analysis Prevention*, vol. 126, pp. 95 – 104, 2019, 10th International Conference on Managing Fatigue: Managing Fatigue to Improve Safety, Wellness, and Effectiveness". [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0001457517304347

[58] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *Vehicular Technology, IEEE Transactions on*, vol. 53, pp. 1052 – 1068, 08 2004.

[59] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation and augmented reality tracking: An integrated system and evaluation for monitoring driver awareness," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 300–311, June 2010.

[60] A. Doshi and M. Trivedi, "On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, pp. 453 – 462, 10 2009.

[61] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, "Real-time system for monitoring driver vigilance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 63–77, March 2006.

[62] D. Tran, E. Tadesse, W. Sheng, Y. Sun, M. Liu, and S. Zhang, "A driver assistance framework based on driver drowsiness detection," in *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, June 2016, pp. 173–178.

[63] B. Cheng, W. Zhang, Y. Lin, R. Feng, and X. Zhang, "Driver drowsiness detection based on multisource information," *Human Factors and Ergonomics in Manufacturing Service Industries*, vol. 22, 09 2012.

[64] L. Motors. How does mercedes-benz attention assist work? [Online]. Available: https://www.loebermotors.com/blog/how-does-mercedes-benz-attention-assist-work/

[65] Volvo. Driver alert control (dac). [Online]. Available: https://www.volvocars.com/en-th/support/manuals/v60/2017-early/driver-support/driver-alert-system/driver-alert-control-dac

[66] ——. Volvo cars introduces new systems for alerting tired and distracted drivers. [Online]. Available: https://www.media.volvocars.com/global/en-gb/media/pressreleases/12130

[67] Bosch. Driver drowsiness detection. [Online]. Available: https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/driver-drowsiness-detection/

[68] NVidia. Nvidia drive ix. [Online]. Available: https://developer.nvidia.com/drive/drive-ix

[69] J. Bar-Ilan, "The history of information security: A comprehensive handbook20081edited by karl de leeuw and jan bergstra. the history of information security: A comprehensive handbook . oxford: Elsevier 2007. 887 pp. (hard cover), isbn: 9780444516084," *Library Hi Tech*, vol. 26, pp. 682–683, 11 2008.

[70] M. Ballantyne, R. S. Boyer, and L. Hines, "Woody bledsoe: His life and legacy," *AI Magazine*, vol. 17, no. 1, p. 7, Mar. 1996. [Online]. Available: https://www.aaai.org/ojs/index.php/aimagazine/article/view/1207

[71] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*,

ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 580–587. [Online]. Available: https://doi.org/10.1109/CVPR.2014.81

[72] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 06 2016, pp. 779–788.

[73] B. C. Csáji, "Approximation with artificial neural networks," 2001.

[74] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[76] ——, "Identity mappings in deep residual networks," vol. 9908, 10 2016, pp. 630–645.

[77] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 04 2017.

[78] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[79] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, 2018.

[80] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *CoRR*, vol. abs/1905.02244, 2019. [Online]. Available: http://arxiv.org/abs/1905.02244

[81] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 2544–2550.

[82] Y. Wu and Q. Ji, "Facial landmark detection: A literature survey," *International Journal of Computer Vision*, vol. 127, pp. 115–142, 2018.

[83] B. Johnston and P. d. Chazal, "A review of image-based automatic facial landmark identification techniques," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, p. 86, Sep 2018. [Online]. Available: https://doi.org/10.1186/s13640-018-0324-4

[84] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 FPS via regressing local binary features," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 2014, pp. 1685–1692. [Online]. Available: https://doi.org/10.1109/CVPR.2014.218

[85] T. Soukupová and J. Cech, "Real-time eye blink detection using facial landmarks," 2016.

[86] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, Jul 2008. [Online]. Available: https://doi.org/10.1007/s11263-008-0152-6

[87] J. Kim and H. Shin, *Algorithm & SoC Design for Automotive Vision Systems: For Smart Safe Driving System*. Springer Publishing Company, Incorporated, 2014.

[88] T. Boji, A. Vuckovic, and A. Kalauzi, "Modeling EEG fractal dimension changes in wake and drowsy states in humans–a preliminary study," *J. Theor. Biol.*, vol. 262, no. 2, pp. 214–222, Jan 2010.

[89] A. Anund and K. Kircher, "Advantages and disadvantages of different methods to evaluate sleepiness warning systems," 2009.

[90] H. Zhang, M. R. H. Smith, and G. J. Witt, "Identification of real-time diagnostic measures of visual distraction with an automatic eye-tracking system," *Human Factors*, vol. 48, no. 4, pp. 805–821, 2006, pMID: 17240726. [Online]. Available: https://doi.org/10.1518/001872006779166307

Acknowledgments

Vorrei ringraziare per primi Filippo Camillo e Fabio Marton, per avermi concesso l'opportunità di sviluppare un progetto stimolante e al passo con le più recenti e innovative tecnologie di questo settore.

Desidero ringraziare anche il Prof. Simone Milani per avermi aiutato nella stesura di questa tesi e per avermi fatto appassionare alla materia grazie alle sue lezioni.

Un ringraziamento speciale va ai miei genitori e a mia sorella Elena, per la pazienza che hanno avuto durante questi anni e per le motivazioni che mi hanno sempre dato.

Grazie anche a Bari, Bebbe, Lori, Tommy e Faggio, per tutti questi anni passati insieme e a Yuri e Lollo per essere stati degli ottimi compagni di uni.

Infine vorrei ringraziare Arianna, per avermi sempre sostenuto durante questo percorso e per aver sempre creduto in me, sia in ambito universitario sia nella vita.