Università degli studi di Torino

SCUOLA DI SCIENZE DELLA NATURA

Master's Degree in Computer Science

Master's thesis

DRIVER DISTRACTION IDENTIFICATION FROM UNOBTRUSIVE SENSORS AND VEHICLE DYNAMICS

Supervisors:
Prof. Marco Botta
Prof. Elvio Amparore

Candidate:
Giuseppe Mazzone
938528

Academic year 2020/2021

We can live a wonderful life in the world
if we know how to work and love,
work for those we love and love what we work for.

Lev Tolstoy

Originality Declaration

Acknowledgements

My masters work and this thesis as a result would not be possible without the support of a lot of people.

Firstly, I would like to thank the University of Torino for making a scholarship available for this project and I thank Prof. Marco Botta, Prof. Elvio Amparore, Prof. Susanna Donatelli and Eng. Idilio Drago for welcoming me into their work team and for helping me to grow professionally and knowledge during the entire period. I am grateful for their support, feedback and comments on this thesis to help and improve my research.

I thank my family, my mom and dad, who allowed me to study and continue my studies in another city and their continued support over the years, without ever missing me anything despite the distance.

To my girlfriend who always supported and followed me throughout my master university career. To my friends and my fellow students, in particular Davide and Matteo. Thanks to all of you for always being there.

Abstract

With the increasing use of mobile phones, interacting with complex infotainment systems of modern cars increases the driver's distraction notably. Consequently, this induces many road accidents and has become an increasingly relevant matter to discuss in recent traffic safety research.

Analyzing driver behavior using machine learning methods is one of the most promising solutions to detect driver distraction. The aim of the research in this thesis is to study whether it is possible to determine distraction using only unobtrusive sensors and vehicle dynamics, with a focus on cognitive distraction. Experiments are carried out in the context of the European project "NextPerception".

In this thesis, we have been using driving data collected on a driving simulator from 26 different participants, consisting in RGB video and vehicle dynamics. We have manually and automatically selected the most influential features from this dataset, such as gaze, Action Units of facial muscles, lane gap, acceleration, and other vehicle dynamics data that, according to the literature, are relevant to classify distracted and non-distracted drivers.

We used Machine Learning methods for classification and regression, such as Random Forest, K-NN, Multi-Layer Perceptron, Long Short Term Memory, and 1D Convolutional Neural Network. The results show that eyes-off-road condition is a very simple task with the front camera data; in fact, even with the very simple MLP model, an F1-Score in the distracted class of about 92% is obtained. For cognitive distraction, on the other hand, we reach poor performance, in fact, the maximum accuracy obtained is 61% and it is even worse for all other models tested, as if it were a random guessing model. We verified that the dataset does not provide enough information to better model cognitive distraction.

Finally, the dominant features that most influence the classification (global features importance) have been identified, thus allowing us to explain the classification and to compare them with recent research.

In conclusion, the classifier was implemented in an embedded system capable of interfacing with the simulator which will allow a fine-tuning of the system.

Italian abstract

Con il crescente utilizzo dei telefoni cellulari e l'interazione con i complessi sistemi di infotainment delle auto moderne, la distrazione del conducente è aumenta notevolmente. Di conseguenza, questo provoca molti incidenti stradali ed è diventato un argomento sempre più rilevante da discutere nelle recenti ricerche sulla sicurezza stradale.

L'analisi del comportamento del conducente utilizzando metodi di apprendimento automatico è una delle soluzioni più promettenti per rilevare la distrazione del conducente. Lo scopo della ricerca in questa tesi è studiare se è possibile determinare la distrazione utilizzando solo sensori non invasivi e la dinamica del veicolo, con particolare attenzione alla distrazione cognitiva. Gli esperimenti sono condotti nell'ambito del progetto europeo "NextPerception".

In questa tesi, abbiamo utilizzato i dati di guida raccolti su un simulatore di guida da 26 diversi partecipanti, costituiti da video RGB e dinamica del veicolo. Abbiamo selezionato manualmente e automaticamente le features più influenti da questo set di dati, come lo sguardo, le unità di azione dei muscoli facciali, il gap di corsia, l'accelerazione e altri dati sulla dinamica del veicolo che, secondo la letteratura, sono rilevanti per classificare i conducenti distratti e non distratti.

Abbiamo utilizzato metodi di Machine Learning per la classificazione e la regressione, come Random Forest, K-NN, Multi-Layer Perceptron, Long Short Term Memory e 1D Convolutional Neural Network. I risultati mostrano che gli "occhi fuori strada" sono un compito molto semplice da individuare con i dati della fotocamera anteriore; infatti, anche con il semplicissimo modello MLP, si ottiene un F1-Score nella classe distratta di circa 92%. Per la distrazione cognitiva, invece, si arriva a prestazioni scadenti, infatti la massima accuratezza ottenuta è del 61% ed è anche peggio per tutti gli altri modelli testati, come fosse un modello casuale. Abbiamo verificato che il dataset utilizzato non fornisce informazioni sufficienti per modellare meglio la distrazione cognitiva.

Infine, sono state individuate le features dominanti che maggiormente influenzano la classificazione (features importance a livello globale), permettendo così di spiegare la classificazione e di confrontarle con ricerche recenti.

In conclusione, il classificatore è stato implementato in un sistema embedded in grado di interfacciarsi con il simulatore che consentirà una messa a punto del sistema.

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous vehicles seem to be closer than expected on their timeline. However, there is still a decade of driving manual as well as semi-autonomous vehicles before we can experience completely automated vehicles on the road. Where the driver has been in charge for more than a century to ensure that the vehicle navigates safely through traffic, more and more automated driving functions are nowadays integrated into our vehicles in order to ensure major safety through traffic.

The promise is to drastically reduce the number of traffic accidents due to human error, but there are clear technological challenges ahead before that can be accomplished.

In recent years, traffic accidents caused by distracted driving have been on the rise with the popularization of smartphones and IVISs. In fact, distraction is one of the greatest risk factors for road crashes. In the United States there are approximately 6 million reported car accidents annually, each day more than 9 people are killed due to distracted driving, more than 1060 people are injured in crashes that involve a distracted driver . According to the National Highway Traffic Safety Administration (NHTSA), in 2018, 8% of all fatal crashes and 14% of all police-known traffic crashes in the USA were caused by distraction [1] .

NHTSA defines distracted driving as any activity that diverts attention from driving, including talking or texting on the phone, eating and/or drinking, talking to people in your vehicle, fiddling with the stereo, entertainment or navigation system or anything else that takes driver's attention away from the task of safe driving.

According to the 100-Car Naturalistic Driving Study " conducted by National Highway Traffic Safety Administration (NHTSA) the secondary tasks that contributed to the highest number of crashes or near-crashes were cell phones, internal distraction, and passenger related secondary tasks (primarily conversations). An explicative example is that sending or reading a text takes the driver's eyes off the road for a minimum of 5 seconds. At 55 mph, this is the same as driving the length of an entire football field with the eyes closed.

Distraction occurs when a driver engages in a secondary activity that removes attention from the primary task of driving safely. Distractions can be both insides (e.g., using a mobile phone or wearable technology, talking with passengers) or outside a vehicle (e.g., looking at roadside advertising or navigating complex road contexts) and in general it can be divided into three types: Visual Distraction, Manual Distraction and Cognitive Distraction.

Recent research suggests that limiting technology-based secondary tasks will reduce driver distraction. There is a suggestion that higher levels of vehicle automation can induce boredom and, consequently, cause drivers to divert their attention towards competing secondary tasks.

The most common secondary task examined in literature was mobile phone use and in-vehicle information systems (IVISs). Mobile phone use while driving is common, with nearly half of the drivers reported sending texts while driving. 1 out of 3 people text while driving. But these tasks refer to visual or manual distraction and not specifically for 'mind-off-road' condition of cognitive distraction.

With the increase in the use of artificial intelligence in every sector, the manual use of the mobile phone is diminished. This is thanks to virtual personal assistants, for example, 'Ok Google' or to the new technologies introduced in cars such as 'Hey Mercedes' or 'Hey BMW' that allow the hands-free use of mobile phones and infotainment systems.

However, hands-free does not necessarily mean safe-driving. In fact these, just like talking to a passenger or listening / singing on the radio, cause the condition called 'mind-off-road' commonly called cognitive distraction.

Cars need to build an understanding of the driving environment, beyond the capabilities of individual on-board sensors, which are always limited. The vehicle safety aspect has been enabled through vehicle sensors that observe the surrounding environment as well as the driver behaviour, which is further used to give alerts to drivers for potential collisions.

While the driver assistance aspect is equipped through In-Vehicle Information System (IVIS) and systems such as Cruise Control which enable drivers to relax.

Vehicle technology has been enriched with an advanced observation of the human driver in the vehicle to bring about a more personalized and safe experience.

In fact, a number of today's new motor vehicles have technology that helps drivers avoid drifting into adjacent lanes or making unsafe lane changes, or that warns drivers of other vehicles behind them when they are backing up, or that brakes automatically if a vehicle ahead of them stops or slows suddenly, among other things.

These and other safety technologies use a combination of hardware (sensors, cameras, and radar) and software to help vehicles identify certain safety risks so they can warn the driver to act to avoid a crash.

1.2 Context of this research: Next Perception

Smart complex systems have become of great significance particularly in healthcare and autonomous driving domains. The precision and timeliness of the decisions depend on the systems' capacity to accurately understand individuals and their environment.

Next Perception aims to bring perception sensing technologies to the next level, enhancing their features to allow for more accurate detection of human behaviour and physiological parameters. Ranging technologies like Radar, LiDAR and Time of Flight cameras have been applied for detecting objects and assist in navigation in recently developed ADAS solutions.

The next generation of these perception sensors will be able to successfully detect humans and accurately monitor their behaviour and physiological parameters. Besides more accurate automotive solutions ensuring driver vigilance and pedestrian and cyclist safety, this innovation will open up new opportunities in health and wellbeing to monitor elderly people at home or unobtrusively assess their health state.

The goal of this project is to develop next-generation smart perception sensors and enhance the distributed intelligence paradigm to build versatile, secure, reliable, and proactive human monitoring solutions for the health, wellbeing, and automotive domains.

The solutions developed in the project are key to realize the foreseen revolutions in Health and Well-Being - patient-centric care through advanced monitoring solutions, and Transport and Smart Mobility - automated driving made safe by drivers, and vulnerable road user monitoring.

The project brings together major industrial players and research partners to address top challenges in health, wellbeing, and automotive domains

The project is further organized in use cases, which help to provide a context for the technology development and allows for requirement solicitation from the target groups as well as small scale pilots.

The three use cases are:

1. Integral vitality monitoring for elderly and exercise

2. Driver monitoring
3. Safety and comfort for vulnerable road users at intersections

The use cases are concrete, feasible and potentially commercially viable examples of the deployment of the new technologies and will steer the development of business models and planning of exploitation steps. Figure 1.1 provides an overview of the use cases.

Use case 2, named 'Driver monitoring', focuses on the definition of innovative sens-

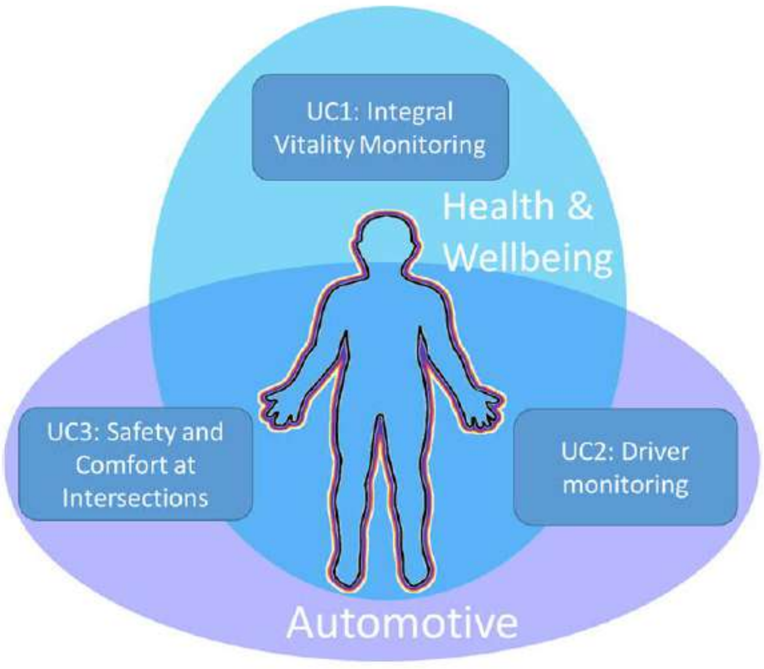

UC1: Integral Vitality Monitoring
Health & Wellbeing
UC3: Safety and Comfort at Intersections
UC2: Driver monitoring
Automotive

Figure 1.1: Use cases in NextPerception mapped to domains

ing solutions to monitor driver state and behaviour in order to mitigate the risk of accidents and their consequences on individuals and, indirectly, on society.

This Use case is the context in which this thesis work takes place.

It considers both public and private vehicles, and trucks with differences in driver monitoring. A relevant goal is to develop a Driver Monitoring System (DMS), which can classify both the driver's distraction states and the driver's emotional states, as well as the activities and positions of occupants (including driver) inside the vehicle cockpit. Examples of distraction states are:

1. **Visual distraction**: refers to the Eyes-off-road situations where driver takes his/her eyes off the road for some other task;
2. **Manual distraction**: refers to situations where driver is doing something else

in the car cabin with his/her hands not being on the steering wheel, for example, drinking water.

3. **Cognitive distraction**: refers to the "look but not see" situations when the drivers' eyes are focused on the forward roadway, but his/her mind is not. Typically, cognitive distractions can result from fatigue, conversation with a co- passenger, listening to the radio, and drowsiness.

4. **Emotions distraction**: refers to situations where driver is unable to drive such as anxiety, panic attack, illness, anger/aggressiveness, and so on.

This use-case proposes a driver cognitive and emotional detection system, considering different machine learning techniques, compared to each other.

In order to achieve that, several sensors and source of information will be considered:

- The use of internal camera, that produce image, audio processing and computer vision (such as Eye-tracking to understand where the driver is looking at, face and expression recognition, etc.).
- Physiological and bio-metric driver signals (heart rate, blood pressure, ECG/EEG, skin conductance and so on).
- Vital signs monitoring and health-related analysis based on mm-wave radar(s), in combination with other sensor types.
- Vehicle dynamic data and driving style (speed, yaw-rate, steering wheel, accelerator and braking behavior, etc.) acquired by ad hoc sensors and by data already available from vehicle (CAN-BUS) in order to detect "abnormal behavior" and so identify tiredness or drowsiness.
- Traffic data (obstacles speed and velocity, position of ego-vehicle in the lane, and so forth), possibly supplied by a simulator.

This information will be used for autonomous driving functions (through Machine Learning techniques), in order to identify the driver's status. The Autonomous Driving Functions that will be implemented are primary Take-Over-Request, in which the system can know the status of the driver and select the most appropriate strategy to adopt, and driver support in that the system can offer "its support" if the driver behaviour appears inappropriate.

Demonstrators in this use case are: driving simulator (RELAB), passenger car "Marylin" (VTT) and Heavy good vehicle TTS. The pilots will be conducted in both a simulation environment and on real vehicles, and will be located in Italy and Finland. First innovations of this use case include:

1. in-vehicle vital signs monitoring by means of UWB/FMCW radar, ECG, GSR and other suitable sensors, as well as camera and acoustic sensors for in-vehicle behaviour monitoring.
2. Driver cognitive and emotional state recognition algorithms based on vital sign and behavioural input illustrate analytics and decision-making innovations.

Next Perception will provide innovative technologies for perception sensors, distributed intelligence and proactive monitoring. Driver drowsiness and distraction are some of the major causes of mortality in traffic accidents worldwide but only recently with the developments on autonomous vehicles, the research on driver status monitoring has received a positive stimulus to explore methodologies to apply and study prototypal reliable and feasible implementations for them.

The starting point is, as for each system for the autonomous vehicle: no sensor type works well for all tasks and for all conditions; so sensors fusion and complex algorithms and intelligent elaboration capabilities are required to provide redundancy for autonomous functions. These research activities have been focused in several methodological strategies, from advanced image-based analysis up to physiological monitoring, to exploit them on drowsiness detection and driver fatigue recognition.

Human Factors related to driver's state are a major cause of traffic crashes. Public and standardization bodies are strongly supporting research to develop in-vehicle systems to continuously monitor driver alertness and performance. Scientific support for the feasibility of this countermeasure concept is provided by research showing that:

1. The use of direct, unobtrusive driver psychophysiological monitoring (e.g., of eye closure) could potentially enhance distraction and drowsiness detection significantly.
2. Drowsy drivers typically do not “drop off’ instantaneously. Instead, there is a preceding period of measurable performance decrement with associated psychophysiological signs.
3. Distraction can be detected with reasonable accuracy using driving performance measures such as “drift-and-jerk” steering and fluctuations in vehicle lateral lane position.
4. The use of secondary/subsidiary auditory tasks (e.g., auditory recognition tasks presented to the driver via recorded voice) could further enhance detection accuracy.

An envisioned vehicle-driver-based state detection system would continuously and unobtrusively monitor driver's condition at a “micro-performance” level, such as minute steering movements, and “micro-behavioral” level, such as driver psychophysiological status, in particular eye closure and facial expressions. The system may enable the handling (and the personalization) of immediate warning signals when driver's state is detected with high certainty, or, alternatively, to present a verbal secondary task via recorded voice as a second-stage probe of driver status in situations of possible drowsiness.

Moreover, the detection of emotion can further increase the driver's safety. Traditionally, emotion recognition has been extensively used to implement the so-called Sentiment Analysis. It has been widely used by different companies to gauge consumer mood towards their product or brand in the digital world. However, it can be also used to recognize different emotions on an individual's face automatically, to make cars safer and more personalized.

In fact, car manufacturers around the world are increasingly focusing on making cars more personal and safe to drive. In their pursuit to build smarter car features, it makes sense for carmakers to use AI to help them understand human emotions.

For example, using facial emotion detection, smart cars can interact with the driver to adapt the level of automation or support the decision-making process if emotions that can affect the driving performance are detected. In fact, (negative) emotions can alter perception, decision making and other capabilities eminent for driving and can even affect physical capabilities.

Therefore, the main innovations to be achieved through NextPerception are:

1. in-vehicle vital signs monitoring by means of UWB/FMCW radar, ECG, GSR and other suitable sensors, as well as camera and acoustic sensors for in-vehicle behaviour monitoring.
2. Driver and passenger cognitive and emotional state recognition algorithms based on vital sign and behavioural input illustrate analytics and decision-making innovations.
3. Monitoring of occupants to detect they are healthy too
4. Monitoring of social behaviour in the cockpit to predict any risk-prone behaviour (e.g. engaging discussions while driving, cognitive distraction of the driver)

The proposed solutions integrate different data sources, collecting a large set of relevant information and processing it in real time.

Extending the interaction with several sources of information, including sensors integrated in wearable devices of occupants and from IoT platform, will improve the fusion of collected data enhancing the accuracy of DMS, ensuring faster detection, increased number of monitored situations, improved privacy with local data processing, and the overall cost reduction of the DMS. Figure 1.2 contains the concepts explained so far.

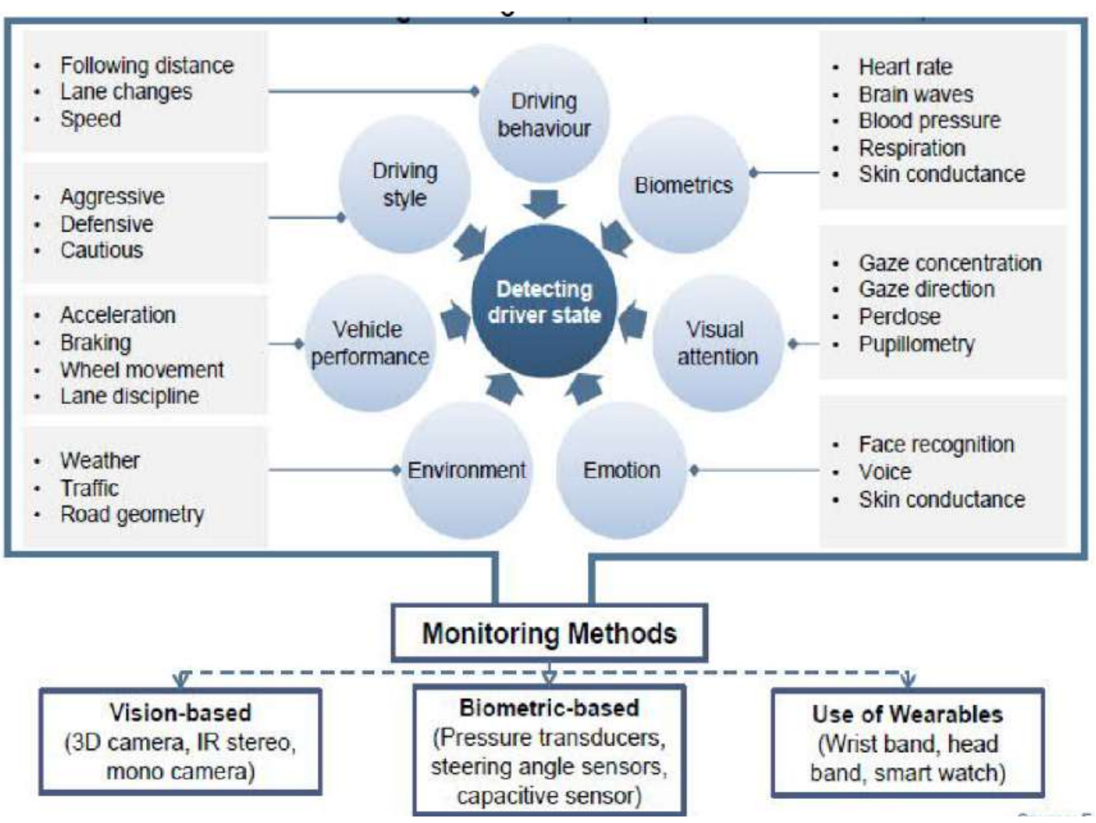


Figure 1.2: The main sensor clusters will be investigated and exploited for the detection of driver status

1.3 Objective of this research thesis

In the context of the 'Next Perception' project, the aim of this research thesis focuses on use case 2, i.e. Driver Status Monitoring.

Specifically, the aim of this thesis is to study whether it is possible to determine distraction using only unobtrusive sensors and vehicle dynamics, with a focus on cognitive distraction.

This study will give as a result a Driver Distraction Classifier (DDC) that will be trained and validated through the analysis and elaboration of data generated from the first pilot of UC1D1 on the Next Perception project from RE:Lab partner.

The focus of the DDC, in Next Perception context, will mostly be on the safety-related motivation (i.e. number of accidents), in the detection of the moments where the driver is distracted, and in its interaction with the decision module to avoid collisions.

1.4 Methodology of this thesis

Based on what is defined in the Next Perception project, and on the two types of distraction (visual and cognitive) induced in the dataset, figure 1.3 shows the idea that in this thesis we want to prove

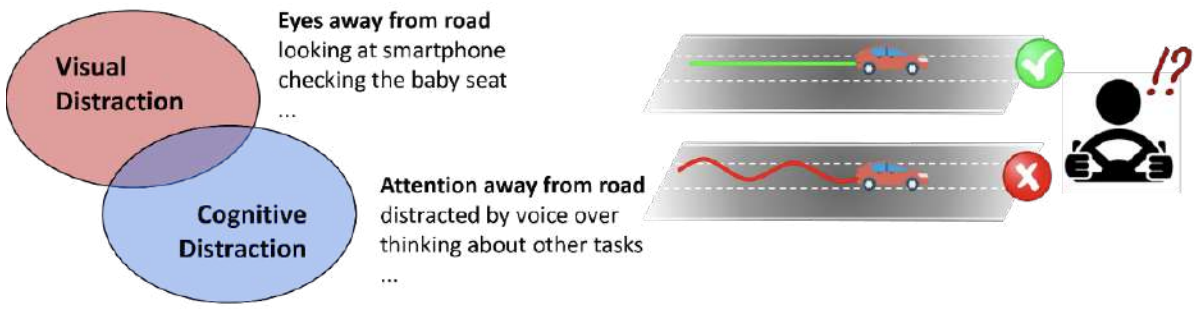


Figure 1.3: Cognitive and Visual distraction idea

In order to achieve the objectives of this thesis in Chapter 2, we first of all, a summary of the methods of preparation, extraction and selection of data and machine learning techniques was carried out with a focus in time series prediction/classification problems.

Subsequently, in Chapter 3 a review of the Driver Automation System and the need for them to monitor the driver's status is reported. Here the various types of distractions and ways used by recent research to detect them are illustrated.

Follows Chapter 4 which describes the dataset used (starting from the collection of data) and the analyzes and pre-processing that have been carried out on it.

Subsequently, in Chapter 5, features extraction and selection are discussed based on recent research and on correlation of them.

After identifying the features, Chapter 6 illustrates the training and evaluation of the tested models. Validation focus on standard assessment metrics: AUC measurements, precision, recall and the resulting f1 score. Furthermore, in this chapter an a posteriori analysis is also carried out on the output returned by the DDC through inferential analysis on the features that contribute most to the classification (global features importance) with eXplainableAI techniques.

In Chapter 7 is explained how our classifier has been implemented in the embedded system able to communicate with the simulator and the remaining components of the Next Perception project.

Finally, in Chapter 8 reports the conclusions, our contribution provided by this research thesis and the limitations encountered, followed by future work to be carried out

A generic architecture that summarizes the work done in this thesis is shown in figure

1.4.

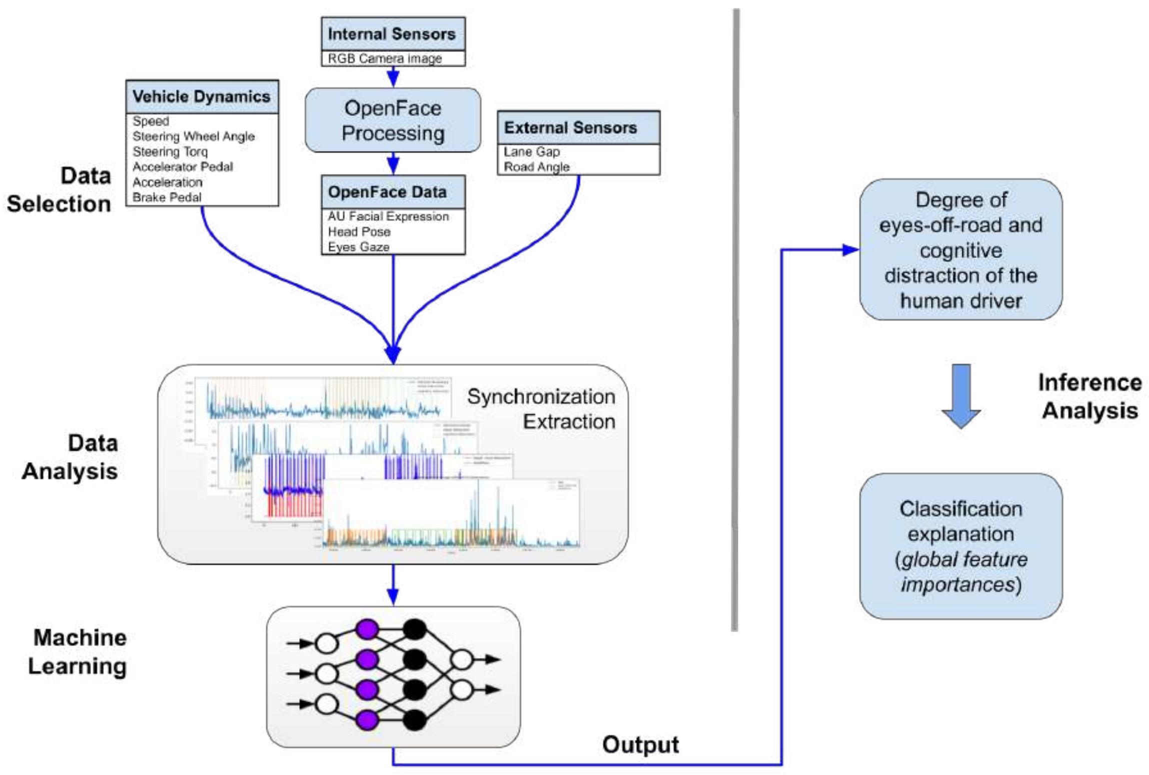


Figure 1.4: Architecture of this thesis work

Chapter 2

Machine Learning Techniques

Machine Learning (ML) is a term used for computer programs that can learn from a data set and be used for prediction and the classification of a data set in hand. Machine learning is a sub-domain of artificial intelligence (AI).

The goal of machine learning is usually to understand the structure of the data and to match that data to models that can be understood and used by humans. Machine learning is a field that continues to evolve. For this reason, there is something else to keep in mind as you work with machine learning methods, or analyze the impact of machine learning processes.

Two widely accepted methods of machine learning are supervised learning trains algorithms based on input by human and output model data, and unsupervised learning algorithm provides a non-labelled data to allow it to detect input and produce output.

2.1 Supervised Learning

The majority of applications related to Machine Learning falls into the Supervised Learning problem that is indeed the most common sub branch of ML today.

In Supervised Learning, the model uses the labelled data for learning. After the thorough analysis of the data, the algorithm determines which label should be assigned to new data based on pattern and associates the patterns to the unlabelled new data.

Supervised Learning algorithms are designed to learn by example and indeed the name "supervised" learning itself comes from the idea that training these algorithms is like having an entity that behaves like a teacher that tells if an output is correct or not. Hence a labelled dataset is provided to the algorithm in which a "right answer" (ground truth) is given at priori and then the algorithm learns from this training data.

Then once the algorithm is trained, it provides a “right” answer based on new data, where the output is unknown, finding a relationship between the input and the output.

In Supervised Learning, the model uses the labelled data for learning. After the thorough analysis of the data, the algorithm determines which label should be assigned to new data based on pattern and associates the patterns to the unlabelled new data.

Moreover, supervised learning problems are categorized into "regression" and "classification" problems.

In a regression problem the aim is to predict results within a continuous output, this means mapping input variables to some continuous function (e.g. predict the age of a person based on their face).

Instead in a classification problem, the aim is to predict results in a discrete output, and this means to classify data into one of two or more discrete numbers (classes or categories) mapping input variables into discrete categories (e.g. predict whether a driver is distracted or not, or predict if he uses a phone, interacts with radio, talks with co-passengers, etc.).

To give a more formal explanation, let’s define some terms that will be used from here on.

Let’s consider with $x^{(i)}$ the input variables while for the predicted value is used $y^{(i)}$ in order to indicate the target value or the “output”. The training example is the pairs ($x^{(i)}$,$y^{(i)}$) with i from 1 to m, where m is the dimension of the training set.

Furthermore, X indicates the input values space and Y the output values space. The aim of supervised learning is to learn a function $\boldsymbol{h} : \boldsymbol{X} \to \boldsymbol{Y}$given a training set $(x(i), y(i))$ so that the learned function $h(x)$ is a “good” predictor for the corresponding value of y.

Some popular examples of supervised ML algorithms are Linear Regression, Logistic Regression, Random Forest, Support Vector Machine and Neural Networks.

It is also of primary importance to highlight that all these Machine Learning algorithms completely depend on optimization techniques, that are the processes by which the optimal set of parameters are found. Here Gradient Descent will be discussed, that is the simplest and used optimization technique.

Other two important concepts are model complexity, and the bias-variance trade-off that are related quantities.

Model complexity refers to the complexity of the function that the algorithm is attempting to learn (similar to the degree of a polynomial). The proper level of model complexity is generally determined by the nature of the training data. If only a small amount of data is provided, or if the data are not uniformly spread throughout different possible scenarios, then the best choice is to opt for a low-complexity model. This is because a high complexity model will overfit if used on a small number of data points.

Overfitting refers to learning a function that fits training data very well, but does not generalize to new and unseen data points. Basically in this case the algorithm is learning to produce the training data without learning the real underlying structure in the data that leads to this output.

Therefore bias-variance trade-off relates to model generalization and any model must be well balanced between bias, which is the constant error term, and variance, which is the amount by which the error may vary between different training sets.

Note that bias and variance typically move in opposite directions from each other; increasing bias will usually lead to lower variance, and vice versa, so it is necessary to find the best trade-off suitable for the specific cases.

2.1.1 Linear Regression

In linear regression a linear relationship is assumed between the input variables x and the single output variable y and can be represented by the equation:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

where y is the predicted value, θ_0 is the bias term, $x_1 ... x_n$ are the features values where n is the number of features, and finally $\theta_1 ... \theta_n$ are the model parameters. The primary goal of ML algorithms is always to build a model, which is basically a hypothesis function which can be used to find an estimation for y based on x the input features finding the right θ values: $y = \theta_{Tx}$.

In order to measure the goodness and the accuracy of this hypothesis function a cost function is used. The cost function is defined as "a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.". The cost function is defined as the average difference between all the results of the hypothesis and the actual predicted output y.

For example the Mean Square Error (called MSE or quadratic loss) is basically the sum of squared distances between the target variable y_i (ground truth) and predicted values $\hat{\boldsymbol{y}}_i$ and this loss function is a way of measuring how well the model fits into the data. The difference between the predicted values and ground truth measures the error difference.

$$e_i = \hat{y}_i - y_i$$

Other loss functions can be used: the Mean Absolute Error (MAE or L1 loss), Huber loss (Smooth Mean Absolute Error) and the Cross-entropy loss (or log-loss) and all of these provide a different measure of how good a prediction model does in terms of being able to predict the expected outcome.

There is not a single loss function that works for all kinds of dataset but each one can perform at their best in different cases. Then, once a proper loss function has been

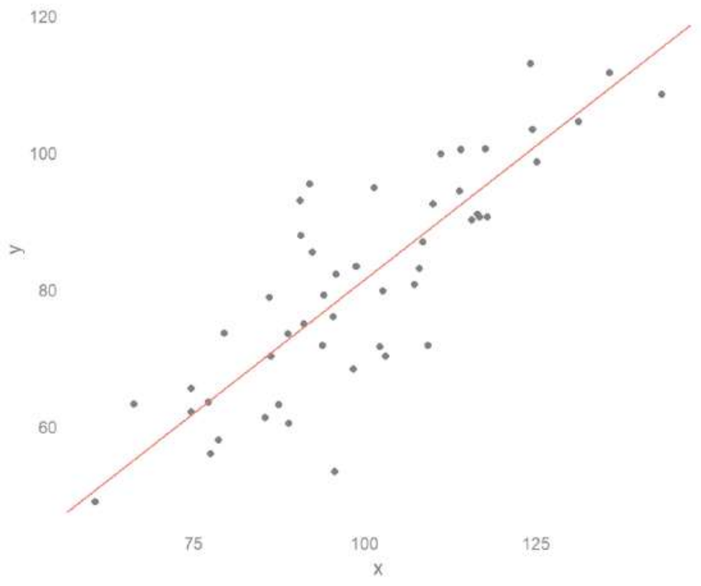


Figure 2.1: An example of a linear model, that is the line that best approximates all the points on the plane.

selected a common optimization method to find the minimum $\min_\theta J\theta$ is the Gradient Descent.

The best fit line is considered to be the line for which the error between the predicted values and the observed values is minimum. It is also called the regression line and the errors are also known as residuals. Residuals can be visualized by the vertical lines from the observed data value to the regression line. To give an example, refer to Figure 2.1 where the straight line that best approximates all the points is easily identified.

It is almost always practically impossible that a system with a number of equations considerably higher than the number of unknowns is solvable, however this system of equations can be formalized in matrix notation, more compact and homogeneous:

$$\mathbf{Xw} = \mathbf{y}$$

The X matrix shows the coefficients of the features and bias; w represents the parameters that best approximate, i.e. those values that multiplied by the coefficients must give as a result the vector y.

2.1.2 Support Vector Machine

When there is a classification problem, as in the example shown in Figure 2.2, the goal is to find the decision boundary, which correctly divides the two classes of examples; two linearly separable classes can be classified by infinite hyperplanes and the goal of the SVM is to find the hyperplane that divides the best way the two classes. This

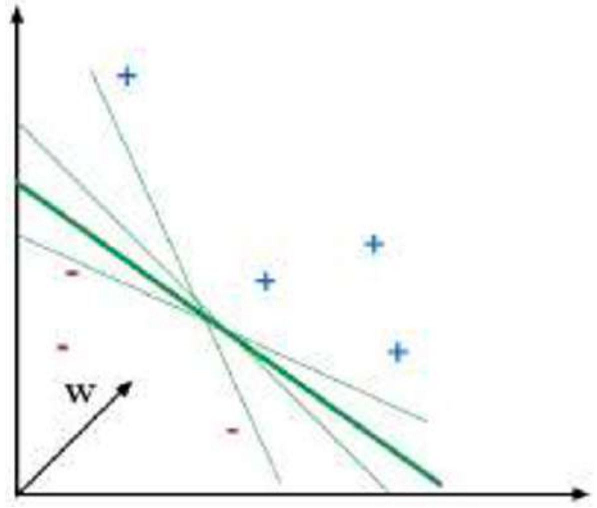


Figure 2.2: Search for a linear classifier among the different (and infinite) possible ones linear classifiers for the positive and negative examples proposed.

kind of classifier models are highly accurate and are able to compute and process the high-dimensional data and they are very flexible in modelling diverse types of data. Moreover, SVMs belong to the general category of kernel methods and thanks to the kernel trick a non-linear classification boundary can be easily generated using a method designed for linear classifiers (Linear SVM).

Support vector learning is the problem of finding a separating hyperplane that separates the positive examples (labeled +1) from the negative examples (labeled -1) and the margin of the hyperplane is defined as the shortest distance between the positive and negative instances (training data sample) that are closest to the hyperplane.

The intuition behind searching for the hyperplane with a large margin is that it is more resistant to noise than a hyperplane with a smaller margin. The maximal margin classifier is the hyperplane for which the margin is the largest. The training examples nearest to the decision boundary are called support vectors: as we shall see, the decision boundary of a support vector machine (SVM) is defined as a linear combination of the support vectors.

An explanation of what has just been said can be seen in the figure 2.3 Formally we suppose that all data satisfy the constraints:

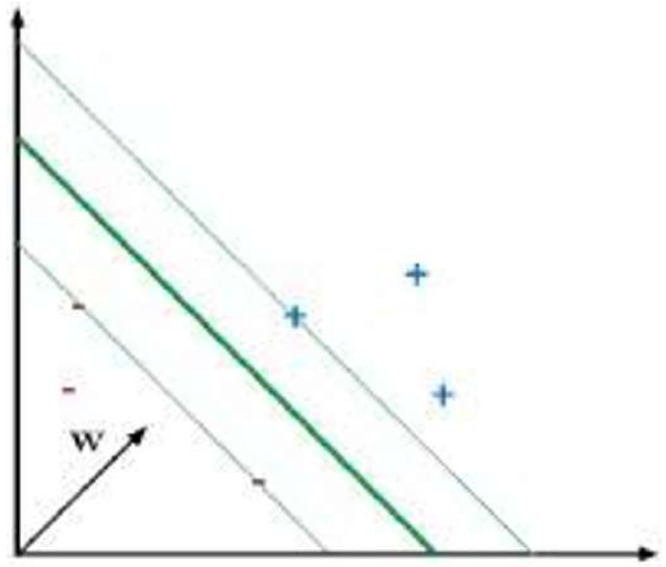


Figure 2.3: The ideal linear classifier, equidistant from the closest examples of both classes.

$$\begin{array}{ll} \omega \cdot x_i + b \geq +1 & y_i = +1 \\ \omega \cdot x_i + b \leq +1 & y_i = -1 \end{array}$$

So we can reformulate the problem as:

$$y_i \left(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + b \right) \geq 1$$

It can be shown that the maximum margin separating hyperplane can be constructed by solving the Primal optimization problem:

$$\min_{\omega \in H} \tfrac{1}{2} \|\boldsymbol{\omega}\|^2 \quad \text{subject to} \quad y_i \left(\boldsymbol{\omega} \cdot \boldsymbol{x}_i + b \right) \geq 1 \quad \forall i$$

As cost function we can use the Relu function so that:

$$\boldsymbol{\theta}^T \boldsymbol{x} > 1 \text{ for } y = 1 \text{ and } \boldsymbol{\theta}^T \boldsymbol{x} < -1 \text{ for } y = 0$$

Now the model must learn parameters θ, or w, such that $\theta Tx > 1$ for $y = 1$ and$\theta Tx < -1$ for $y = 0$. This is related to the large-margin intuition.

The overall objective for a SVM looks like:

$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} \operatorname{Cost}_1 \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \right) - \left(1 - y^{(i)} \right) \operatorname{Cost}_0 \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \right) \right] + \tfrac{1}{2} \sum_{i=1}^{n} \boldsymbol{\theta}_j^2$$

Notice how λ (the regularization parameter which governs the trade-off between overfitting and the complexity of the model) has been replaced by another constant that plays the same role $C = 1\lambda$. Looking at the formula it is clear how a Support Vector Machine chooses a decision boundary which maximizes the margins from all the classes.

For the purposes of this discussion, it is assumed that the SVM constant C is set to a very large value, $C \to \infty$. A large value of C acts like a small value of λ, this is like having SVM with no regularization, and this means that the model will be subject to overfitting. As C goes up, the penalty associated to the term with $Cost1$ and $Cost0$ goes up, so the SVM is highly incentivized to make both the costs equal to 0 (i.e. make $\|\theta Tx\| > 1$).

Moreover, SVM is a very powerful tool thanks to the kernel trick that makes possible to classify non linearly separable datasets. What the kernel trick does is to use some mapping function $K(x)$, called kernels, that projects the data onto a higher-dimensional space where it is possible to find a linear decision boundary.

Indeed, the basic idea with nonlinear SVMs is to map training data into a higher dimensional feature space via the kernel function $K(x)$ and construct a separating hyperplane with maximum margin in the input space.

It can be shown that a linear decision function in the feature space corresponds to a non-linear decision boundary in the original input space.

2.1.3 Artificial Neural Network And Deep Learning

Artificial Neural Networks (ANN) are a powerful and universal tool, inspired by the biological nervous system and the human brain, used mainly for pattern recognition, computer vision and function approximation. Indeed, when dealing with a complex, partially unknown and non-linear system ANN are the “perfect” tool for fulfilling the task of a non-linear function approximator.

Non-linear decision boundary can be done mainly in two different ways: Adding non-linear features or using ANN modeling as a non-linear decision boundary by internalizing the nonlinearity.

In our nervous system, Neurons are computational units that take inputs (dendrites) as electrical inputs (called "spikes") that are connected to outputs (axons). Dendrites are like the input features $(x_0, x_1, ..., x_m)$ and the output is the result of the hypothesis function $\varphi(\bullet)$, i.e the activation function. In neural networks a logistic or sigmoid activation function in often used.

The simplest structure for a neural network is the Perceptron that is a simple binary classification algorithm proposed by Frank Rosenblatt in 1957. Its basic unit component is the so-called artificial neuron.

The perceptron structure is formed by an input layer, that stored the components of the input vector x, the channels of weights w that connect the input layer to the neuron and finally the body of the neuron that contains the sum function and the activation function. In figure 2.4 you can see what has just been described.

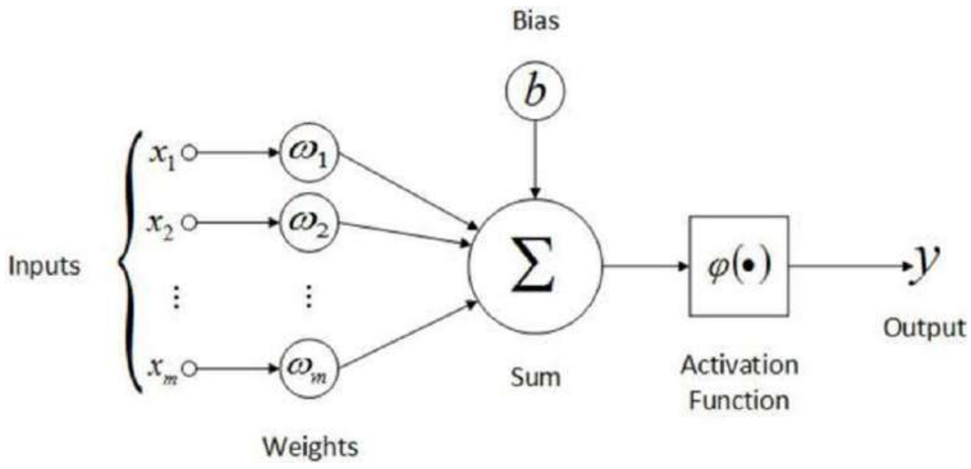


Figure 2.4: A simple Neural Network Architecture: Perceptron

In the figure it can be seen that in addition to the weights, another factor is also considered, namely bias. Weights are the values that are computed over the time of training the model. Initially we start the value of weights with some initial value and these values get updated for each training basis of the error function.

Bias is like the intercept added in a linear equation. It is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron.It helps in training the model faster and with better quality.

The net sum is the sum of the multiplication of input and weight. Activation function is used to make the neural network non-linear. It limits the output between 0 and 1.

The Perceptron learning process can be described as in the following steps:

1. Perceptron takes the input, multiplies them by the weights and computes their sum. The weights allow the perceptron to evaluate the relative importance of each input.
2. The bias factor is added. This procedure is needed in order to move the activation function and makes possible to fine-tune the numeric output of the perceptron.
3. The neuron feeds the previously computed sum through the activation function (in this case a logistic or sigmoid function) that maps the input values to the required output values. Activation function help the learning process especially in case of a multi-layer perceptron (MLP). The non-linear properties of the activation function make possible to train complex neural networks.

4. The perceptron output is a classification decision. In a feedforward neural network the information always moves in one direction, from the input layer to the output. In case of an MLP the output of one layer perceptron is the input of the next layer. The final layer is the output layer and its output is the final prediction.

As anticipated before, a multilayer perceptron (MLP) is a structure made by many perceptrons, stacked in several layers where each layer takes input from the previous and then signals to the next layer. Figure 2.5 shows a Multi-Layer Perceptron with three layers. Each perceptron in the first layer on the left (the input layer), sends outputs to all the perceptrons in the second layer (the hidden layer), and all perceptrons in the second layer send outputs to the final layer on the right (the output layer). Neural

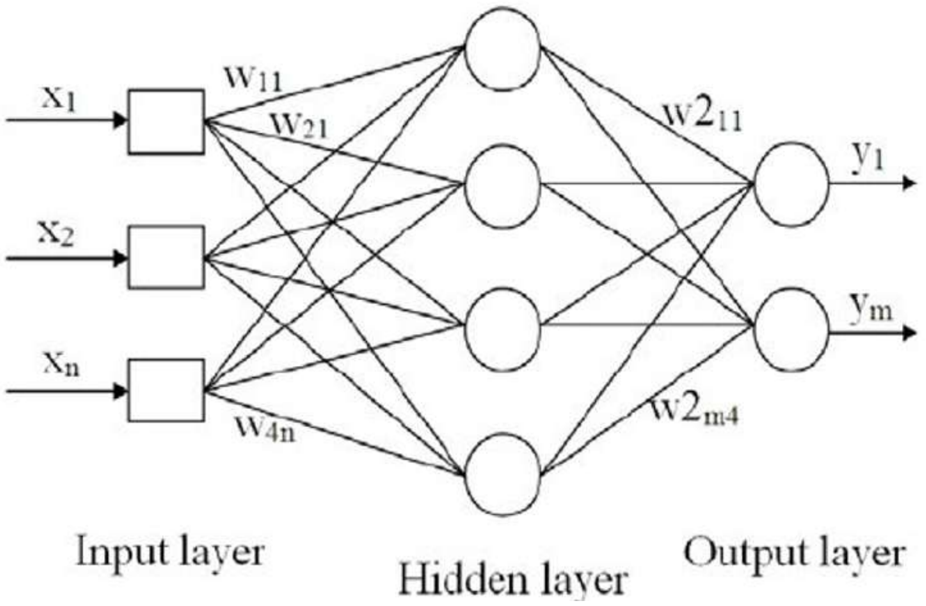


Figure 2.5: A Simple Multi Layer Perceptron with one hidden layer

Networks algorithms learn by discovering better weights at every step that result in a more accurate prediction. There are several algorithms for the fine tuning of the weights and the most common is the backpropagation that is a basic concept in modern neural network training and it belongs to supervised learning.

The Backpropagation training method for Feed Forward Neural Network is based on the gradient descent approach that iteratively adjusts the network weights and gradually decreases the error on the training set in an iterative procedure.

Backpropagation is used in order to update the parameters Ω (matrix of weights w) and this will bring, after a certain number of back and forward iteration, to the optimal set of parameters. In this backpropagation phase the term $\delta j(I)$ is computed for each node and in each layer, and it represents the error of node j in layer I that is to say that captures the error in the activation and is computed starting from the last layer. Therefore, by iteratively executing the forward and backward propagation phases the set of optimal parameters Ω minimizing the cost function is reached.

Once the Artificial Neural Network is trained, it provides the “correct” predictions. It is also important to note that one of the most important hyperparameters for ANN is the learning rate that specifies how fast the learning process is performed. The learning rate parameter value lies between 0 and 1 and its correct setting is crucial to the success of the learning process and finding the optimal values.

Deep learning is a branch of machine learning, more specifically of Artificial Neural Network, that has great flexibility for learning complex patterns in real-world data. In deep learning the more abstract representation calculates in terms of the less abstract ones. Deep neural networks learn categories incrementally using their hidden layers and have to be fed by a huge amount of data.

Using deep networks reduces the need for domain expertise and dedicated feature selection as the model learns complex patterns on its own in an incremental approach. In theory, a simple neural network with a hidden layer can estimate almost any function. But in practice, shallow neural networks can be inefficient for learning complex patterns, nonlinear correlations and large data sets.

Neural networks generally have different input features but can also have more target features. The inputs feed into layers of hidden units, which can be considered characteristics that are never directly observed but useful for prediction.

Each layer independently from the others evaluates the input and classifies it through the available examples. The sum of all these outputs also moderated by the weights for each result of the different "neurons" will be the final output result.

The fundamental part of neural networks is the hidden layer that processes the output of another neuron to produce a more specific response by adding features up to the final output without being seen from the outside. The formula from one layer to the next is this equation:

$$o_j = f\left(\sum_i w_{i,j} a_i + b_i\right)$$

The layer with the nodes a serves as the input for the layer with the nodes o. To calculate the values for each output node, we need to multiply each input node by a weight w and add a bias b.

All these must then be added together and passed to a function f. This function is considered to be the activation function and there are various functions that can be used depending on the level or problem. It is generally common to use an adjusted linear unit (ReLU) for hidden levels, a sigmoid function for the output level in a binary classification problem, or a softmax function for the output layer of multi-class classification problems.

Each layer is implemented as a separate module that implements the prediction forward and the BP (Back Propagation) backwards, i.e. it passes the importance of the weights, updated gradually, even to the previous layers: The algorithm first initializes the weights with random values that are modified during training through BP.

This is done using optimization methods (also called optimizers) such as gradient descent, specifically used to reduce the error between the calculated and desired output (also called the target output). The error is determined by a loss function whose entity we want to reduce with the optimizer such as the entropy loss function.

In training each neuron there are two variants of the backpropagation algorithm:

- **By pattern**: updates the weights backward starting from the output level, using the generalized delta rule
- **By epochs**: called batch mode, the weights are updated only after all the examples have been processed. The learning process continues from age to age until the stop condition

There are mainly two stopping criteria:

1. Total Mean Square Error change: a situation of convergence is considered when the rate of change of the mean square error per epoch is sufficiently small (or does not change at all), usually in the range of change of [0.1, 0.01].
2. Generalization based: called batch mode, the weights are updated only after all the examples have been processed. The learning process continues from age to age until the stop condition

The fascination of neural networks and in deep learning, lies in the fact that the hidden layers do not have to be defined by the designer, in fact what the hidden layers represent is a learning result. Some examples of deep neural networks are LSTM and CNN.

CNN networks or "convolutional neural networks" (CNN) are neural networks that find wide application in the field of Computer Vision, in the 3d version, but which have found positive results in certain applications also in the 1D format applied to time series or in NLP (Natural Language Processing) field. They are so called because they are based on the convolution operation. That is, it is a mechanism that allows you to slide a filter (or kernel) on the input, and to calculate at each iteration a value representing the inverse correlation. In fact, the convolution operation is equivalent to the correlation operation only that the filter is first rotated and therefore the result of the convolution will be opposite to that of the correlation.

These networks consist of three specific types of layers (see figure 2.6) :

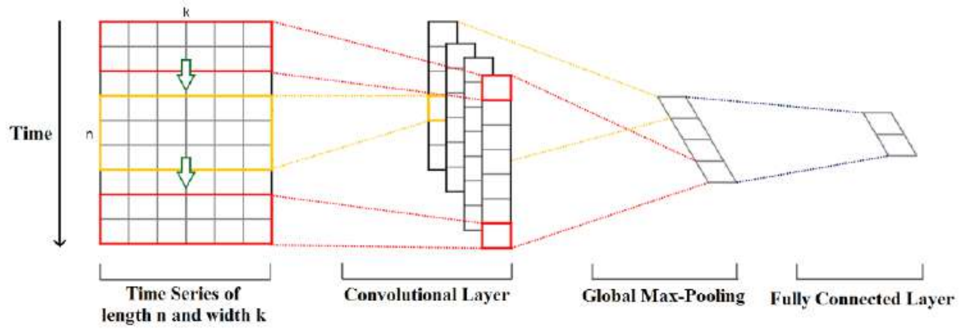


Figure 2.6: A Convolutional Neural Network on a time series

1. **Convolutional layer**: characteristic of this type of neural network, it is inspired by biological processes for visual analysis in living organisms. The layer of neurons that deals with convolution divides the data into various overlapping fragments, which are then analyzed to identify the particularities that characterize it, transferring the information to the following layer in the form of a "feature map" containing the relationships between neurons and particularity. To do this, at each positioning on the input, the filter weights are multiplied with the content of the input and a sum is calculated. The result of this convolution operation is a new vector / matrix (feature map), smaller than the input, which contains a representation of the same but higher level. It is possible to set the convolution width to indicate how narrow it should be compared to the original.

2. **Pooling layer**: it is a fairly common process in neural networks, which consists in reducing the size of the data by generalizing, in order to speed up the analysis without losing too much precision. In the case of an image, the process is very similar to a reduction in pixel quality. This layer therefore allows to simplify the information extracted from a convolution layer. The degree of simplification depends on the value of the offset for moving the window (“stride”). A frequently used pooling algorithm is max pooling, which fetches the highest value on the window.

3. **Normalization layer**: allows to avoid anomalies due to previous layers. The most used function, since it is the fastest, is called ReLu or Rectified Linear Unit

There is also a fourth layer called the full connection layer or the total connection layer. It is the last hidden layer of the neural network, in which all the inputs of the various neurons are put together, allowing the particularities to be identified. It is needed at the end of the process to classify the represented input with a higher level of abstraction. In this layer all the nodes of consecutive layers are connected to each other.

Convolutional networks have some advantages in terms of performance: they are more parallelizable than recurrent networks, since the state of the network, at each step, depends solely on the local context (through the convolution operator) rather than on the set of past states, as for the RNN. But to its detriment, the convolutional network is unable to recall examples seen previously.

The RNN networks or "recurrent neural network" instead, are a particular type of network that has memory of what it has processed in the short term: in fact, they take as input both the current example and the one they previously perceived in a cyclic way, in such a way to be able to consider more than one example and have more context and precision in the decision (see figure 2.7).

It is different from other Artificial Neural Networks in it's structure. While other networks “travel” in a linear direction during the feed-forward process or the back-propagation process, the Recurrent Network follows a recurrence relation instead of a feed-forward pass and uses Back-Propagation through time to learn.

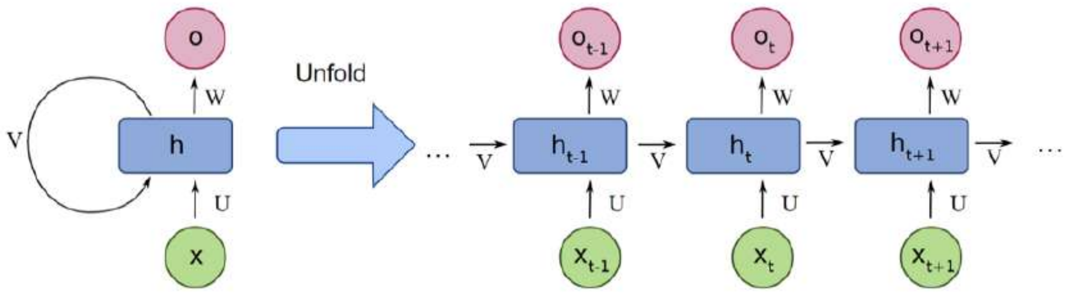


Figure 2.7: A RNN unit and its unrolling

This network is based on two different functions:

1. h_t: contains information about the previous state. In fact, it is computed using the previous vector $h_(t-1)$ and the current word vector x_t. We also apply a nonlinear activation function f (usually tanh or sigmoid) to the final summation. It is acceptable to assume that h_0 is a vector of zeros.

$$h_t = f\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$$

2. y_t **or** o_t: calculates the expected state vector in a given time interval t using for example the softmax function to produce a vector (V, 1) with all elements adding up to 1. This probability distribution gives us the index of the next state that most likely comes from the distribution of the previous and memorized examples. It uses the cross-entropy loss function in each phase of time t to calculate the error between the predicted and the actual word.

$$y_t = \text{softmax}\left(W^{(S)}h_t\right)$$

W weights are matrices initialized with random elements, adjusted using the error from the loss function. We make this adjustment using the back-propagation algorithm.

Once you have the right weights for your training data, you can make a new prediction.

This information is then composed together in order to obtain some kind of correlation between events close to each other. These correlations are called "long-term dependencies" because each event is a function of the events that preceded it.

The "pure" RNNs present a big problem; they cannot remember facts seen a long time ago but always tend to remember the recent ones, in fact they were quickly replaced by LSTM (Long Short Term Memory) which differ in that the updates of the hidden layer are replaced by specially created memory cells, called gates, which allows you to remember long-term dependencies.

An LSTM (long-short term memory network) is a type of recurrent neural network that allows for example the accounting of sequential dependencies in a time series.

Given that correlations exist between observations in a given time series (a phenomenon known as autocorrelation), a standard neural network would treat all observations as independent, which is erroneous and would generate misleading results.

The main key to the operation of an LSTM is the state of the cell, and as you can see in figure 2.8, it is the horizontal line that crosses the top of the diagram.

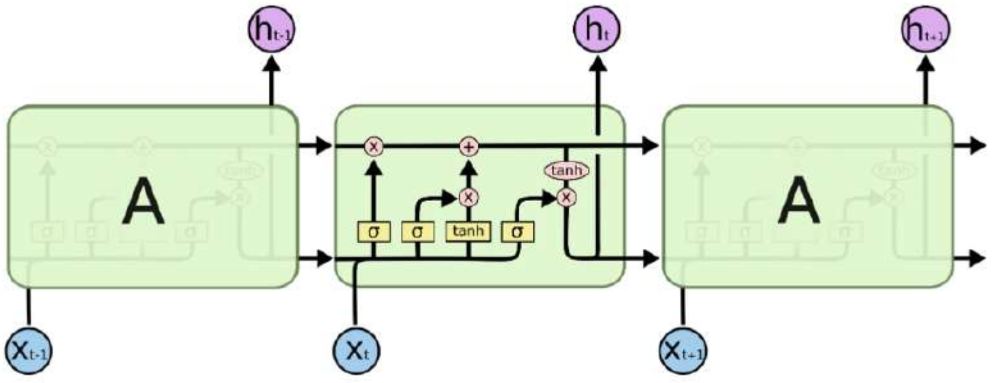


Figure 2.8: An unfolded Long Short Term Memory

LSTM and CNN layers are often combined when working with a time series. This allows for the LSTM layer to account for sequential dependencies in the time series, while the CNN layer further informs this process through the use of dilated convolutions.

With that being said, standalone CNNs are increasingly being used for time series forecasting, and the combination of several Conv1D layers can actually produce quite impressive results, rivalling that of a model which uses both CNN and LSTM layers.

2.2 Unsupervised Learning

The second major problem type of Machine Learning is Unsupervised Learning (UL) that allows to find and infer the underlying structure of the data by clustering the data based on relationships among the variables in the dataset. In unsupervised learning unlabeled data are provided and the aim is to find the underlying causes and identify the intrinsic structure and hidden pattern behind the data.

Since in unsupervised learning there is no prior knowledge on what the "clusters" look like, in contrast with supervised learning, there is no feedback based on the prediction results since labelled data are not provided.

So, the training dataset is made up as a set of input examples where there aren't target variables (labels). Therefore, unsupervised learning technique turns out to be very useful in all those cases where there is no clue of what the desired target output should be, such as, for example, the determination of a reference market for a new product that has to be put on the market by a company.

There are many applications of Unsupervised Machine Learning techniques some examples of that can be:

1. Distance-based methods: There are many type of distance-based methods unsupervised such as clustering with k-means or DB-Scan. In clustering automatically split the dataset into groups according to similarity. However, it can happen that cluster analysis overestimates the similarity between groups and does not treat data points as individuals. For this reason, cluster analysis is a poor choice for applications like customer segmentation and targeting.
2. Anomaly Detection: automatically discover unusual data points in the dataset. This is useful in pinpointing fraudulent transactions, discovering faulty pieces of hardware, or identifying an outlier caused by a human error during data entry.
3. Latent variable models: are commonly used for data preprocessing, such as reducing the number of features in a dataset (dimensionality reduction and PCA-Principal Component Analysis).

2.2.1 K-means Clustering

The fundamental idea behind Clustering is to group and categorize a set of objects into many subsets, called clusters, in such a way that the all the items inside one subset are most "similar" to each other.

The way to distinguish between "similar" and "dissimilar" items is done through the use of some metrics to calculate the similarity. Several metrics are present and the research about which among the different measurements is the best is one of the challenges.

The most used metric is the Euclidean distance between points x_1 and x_2, that basically is the length of the line segment connecting them. In Cartesian coordinates, for two points, $x_1 = \left(x_1^{(1)}, x_1^{(2)}, \ldots, x_2^{(n)}\right)$ and $x_2 = \left(x_2^{(1)}, x_2^{(2)}, \ldots, x_2^{(n)}\right)$, their Euclidean distance is given by:

$$d\left(\boldsymbol{x}_1, \boldsymbol{x}_2\right) = \sqrt{\sum_{i=1}^{n} |x_1^i - x_2^i|}$$

where n is the dimension of a data point that is the number of features that characterize each data.

K-means is one of the simplest unsupervised learning algorithms that solve the clustering problem. The main idea of K-means is to define "K" number of centroids that represents the number of clusters.

Each element in the data set is assigned to a cluster center (centroid) to which it is closest, for example assigning a given sample of the dataset to the centroid with respect to which it has the minimum Euclidean distance.
Provided that there is a suitable distance definition, then the algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids
2. Assign each object to the group that has the closest centroid
3. When all objects have been assigned, recalculate the positions of the K centroids
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

These steps are iterated until the algorithm converges to one optimum value (most likely a local optimum). The choice of the K value has to be taken in accordance to the number of classes in which the data must be clustered. Figure 2.9 shown resulting clusters of K-Means application obtained with a K equals to three.

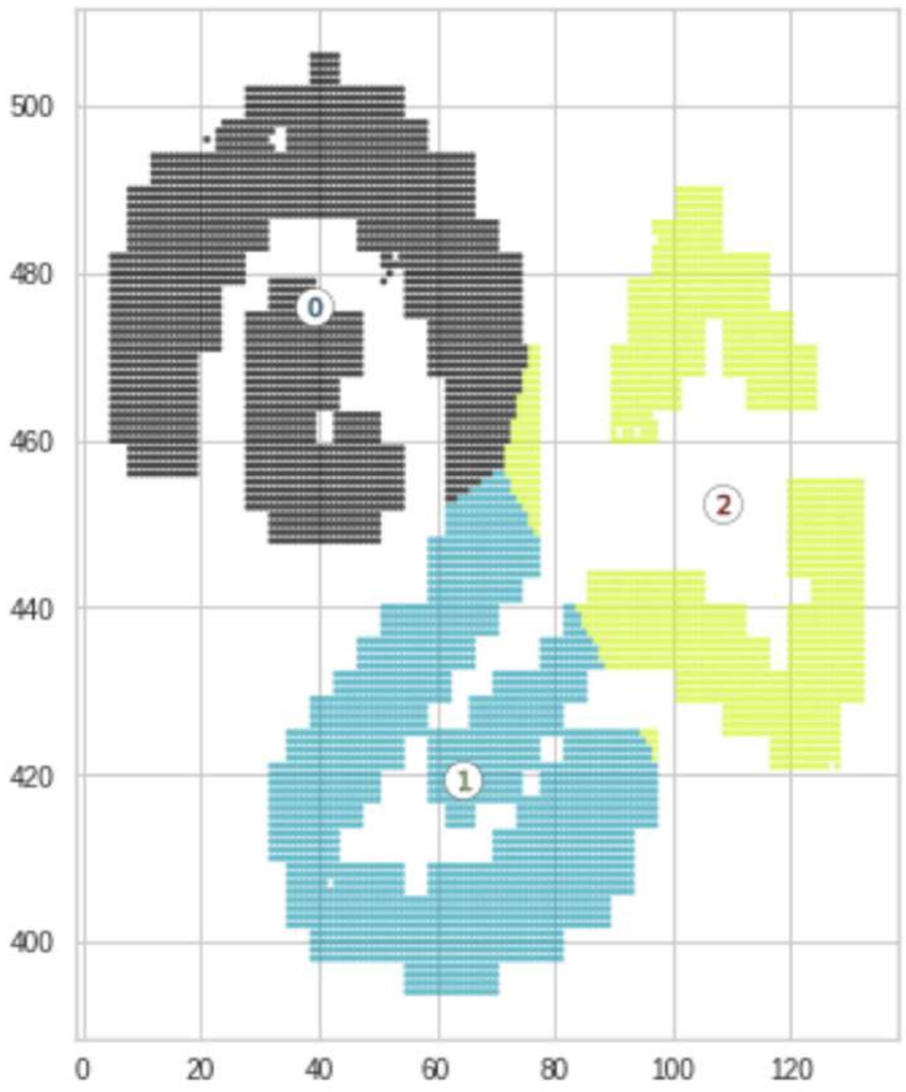


Figure 2.9: K-means clustering example with 3 centroids

2.3 Time Series Classification

Time series classification attempts to categorize time series into distinct categories, and it is used for a wide range of applications. Some applications include the recognition of signals, biometrics, sequences, sound, trajectories, and more. The challenge of using time series is that time series are structural patterns that are dependent on element order.

Traditionally, time series classification was tackled using distance-based methods [2]. However, recently, artificial neural networks have had many successes in time series classification [3].

For example, in recent research there are many uses of neural networks as recurring networks, such as RNN or LSTM [4], in order to capture long-term dependencies in sequences. Also, recent work has shown that feedforward networks such as Multi-Layer Perceptrons (MLP) and temporal Convolutional Neural Networks (CNN) [5] can also achieve competitive and sometimes better results for time series recognition.

Part of the recent successes of neural networks is due to the recent availability of data. However, acquiring large amounts of data can be a problem for many time series recognition tasks. One solution to acquiring more data is to generate synthetic patterns, i.e., data augmentation, data extraction.

Notably, data extraction/augmentation is a universal model-independent data side solution. Features Extraction attempts to increase the generalization ability of trained models by reducing overfitting and expanding the decision boundary of the model [6].

2.3.1 Time Series Processing

With the raise of smart sensors and of the Internet of Things paradigm, there is an increasing demand for performing Data Mining tasks (classification, clustering, outlier detection, etc.) on data stream produced by these interconnected devices.

In particular, Data Mining for time series has gained a relevant importance in the last decade in automotive systems. Time series generally refers to a sequence of data that are generated in successive order over a period of time and time-based attributes play an important role in a time series dataset.

For these temporal data, data processing can be performed using various algorithms and decomposition techniques for time series analysis. In addition, features can also be obtained by sequence comparison techniques, such as dynamic time warping or other measures of similarity.

Data preprocessing is a technique to transform the raw data into some meaningful and understandable format.

Time series usually contain some distortions, which could be consequences of bad measurements or just a property of the underlying process that generated the time series. The presence of distortions can seriously deteriorate the performance of the models. Pre-processing transformations can greatly improve the performance of time series applications.

The task of the pre-processing transformations is to remove different kinds of distortions/noise. Some of the most common pre-processing tasks are: offset translation, amplitude scaling, removing linear trend, removing noise, etc.

Another advantage that we can achieve thanks to pre-processing transformations lies in the fact that if we used a single datapoint of a time series to perform the classification we would not consider datapoints seen previously. It is possible to perform processing operation to extract new information from the past by statistical operation, in such a way that the new created data have information about the past within them so as to enhance and increase the data and the decision boundary recognition capacity.

There are many ways to processing a time series in order to make predictions [7]. The most used are:

1. Sliding Window: In a sliding window, tuples are packed within a window that moves across the stream of data according to a fixed interval. A time-based sliding window with a length of x seconds and a sliding interval of y seconds contains tuples that arrive within an x-second window. The tuples within the window are evaluated every y seconds. Sliding windows can contain overlapping data and the same event can belong to more than one sliding window.
   Moving average model is probably the most sliding window approach to time series modeling. The current moment is the average of the moments seen in the past. The moving average can be used to identify interesting trends in the data. Depending on the value we assign to y, the splicing can be with or without overlap So, defining a window to apply the sliding model allow to to smooth the time series, and highlight different trends. In fact, thanks to this windows we can calculate different statistical values that indicate the progress of the datapoints on the time. Some of them are the calculation of the maximum or minimum value within the window, the average (i.e. moving average), the standard deviation and other statistical values. So, sliding window is the way to restructure a time series dataset in such a way that the machine learning models have a wider visibility considering also previous data rather than considering a single datapoint.

2. Exponential smoothing: Exponential smoothing uses a similar logic to moving average, but this time, a different decreasing weight is assigned to each observations. In other words, less importance is given to observations as we move further from the present.

3. Tumbling Window: In a tumbling window, tuples are grouped in a single window based on time or count. A tuple belongs to only one window. This method can be applied, for example, for the computation of the average of a price of a stock over the last five minutes, repeated every five minutes.

As already mentioned, another way to improve time series is to use data augmentation techniques.

A taxonomy of time series data augmentation techniques was defined in [8] (Figure 2.10). They start the discussion from the simple transformations in time domain. And then they discuss more transformations on time series in the transformed frequency and time-frequency domains.

Besides the transformations in different domains for time series, they also summarize more advanced methods, including decomposition-based methods, model-based methods, and learning-based methods. For learning-based methods, they further divide them into embedding space, deep generative models (DGMs), and automated data augmentation methods (Figure 2.10).

Data augmentation techniques will be briefly explained below:

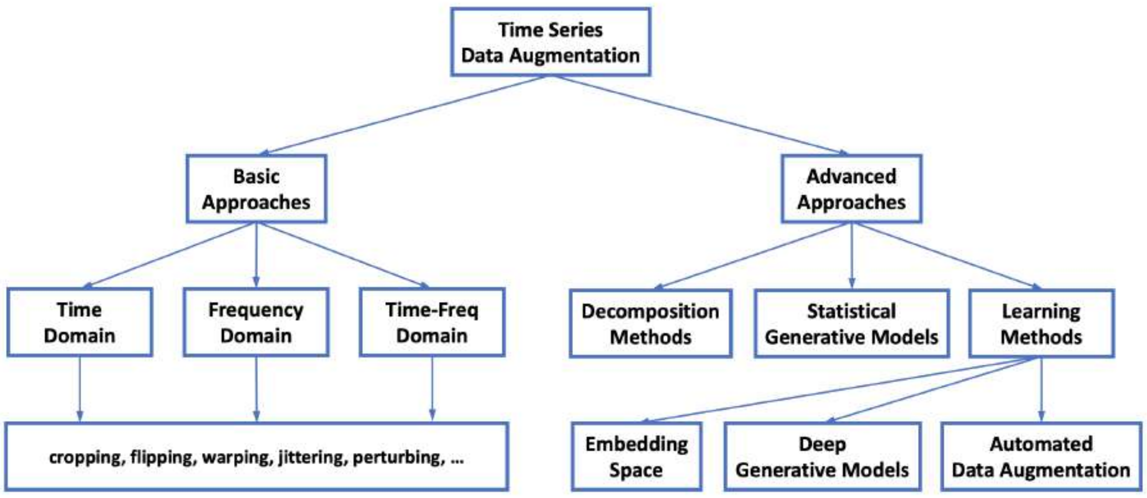


Figure 2.10: A taxonomy of time series data augmentation [8]

1. Time Domain: the elements of the time series are displaced to different time steps than the original sequence. To do this we manipulate the original input time series directly. Some example of time domains transformation are
   - Slicing: The general concept behind slicing is that the data is augmented by slicing time steps off the ends of the pattern

     $$\mathbf{x}' = x_{\varphi}, \ldots, x_t, \ldots, x_{W+\varphi}$$

     where W is the size of a window and φ is a random integer such that $1 \leq \varphi \leq T - W$. Slicing in this way is also sometimes referred to as Window Slicing (WS) due to the use of a window of size W and has the advantage of preserve time dependencies.
   - Permutation: refers to a method of rearranging segments of a time series in order to produce a new pattern. It should be noted that permutation does not preserve time dependencies.
   - Time Wrapping: Time warping is the act of perturbing a pattern in the temporal dimension. This can be performed using a smooth warping path or through a randomly located fixed window. So, data wrapping re-samples the input signal by a randomly selected factor and not preserve time dependencies.

2. Frequency Domain: Frequency domain transformations are transformations that are specific to periodic signals, such as acoustic data. The following are common methods of frequency domain transformations for data augmentation.

   - Frequency warping: In audio and speech recognition, frequency warping is a popular method of data augmentation and it is the act of perturbing a pattern in the frequency domains, as in time domain.
   - Fourier transform-based methods: augment by manipulating the data under a Fourier transform. Gao et al. [9] proposed utilizing amplitude and phase perturbations in order to augment in the frequency domain. This is done by adding Gaussian noise to the amplitude and phase spectra found by a discrete Fourier transform.
   - Spectrogram augmentation:frequency warping data augmentation is performed before conversion into a spectrogram. However, recently, a method called SpecAugment [10] was proposed that augments the spectrogram data itself. In order to augment the data, SpecAugment performs three key operations on the spectrogram: time warping, frequency masking, and time masking. In this way, SpecAugment is both a time domain and frequency domain augmentation method.

3. Time-Frequency Domain: Time-frequency analysis is a widely applied technique for time series analysis, which can be utilized as an appropriate input features in deep neural networks.

4. Decomposition-based Methods: as a part of advanced data augmentation methods [8] Decomposition-based time series augmentation has also been adopted and shown success in many time series related tasks, such as forecasting and anomaly detection. The idea is to decompose a time series in order to generate new time series.

5. Statistical Generative Models: Time series augmentation approaches based on statistical generative models typically involve modelling the dynamics of the time series with statistical models

As demonstrated in [8], performing data augmentation operations allows for improvements during the learning phase and therefore to generate models that are able to generalize and classify better.

Is important to note that some data augmentation methods improved the accuracy and some methods were detrimental. As demonstrated in [11] for multivariate time series classification for deep learning approach, the best way to augment data and extract more information one single data-point is using time-domain transformation, such as Slicing, Window Warping, in fact, as verified by them, this transformations tended to have the most positive effects for each model while other, i.e. Frequency Wrapping had significantly degraded accuracies.

Chapter 3

State of the art

3.1 Driving Automation System

The continuing evolution of automotive technology aims to deliver even greater safety benefits and automated driving systems (ADS) that, one day, can handle the whole task of driving when we don't want to or can't do it ourselves.

Fully automated cars and trucks that drive us, instead of us driving them, will become a reality.

The automotive industry worldwide invests enormous sums in the automation of vehicles to make driving safer and more efficient, but also more comfortable.

According to the German Association of the Automotive Industry (VDA), the German automotive sector will invest almost 68 billion Euros in electric cars, digitalization, connected, and automated driving over the next three years [12].

Intelligent infrastructure with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications supported by advanced electronic driver assistance systems are capable to achieve substantial reduction in accidents and death-rates thus making Europe the by far safest traffic region in the world and the arrival of autonomous driving will revolutionize the way people travel.

Autonomous mobility can become a $1.4 trillion market by 2040 in terms of car sales. When considering the additional revenue from autonomous mobility services, the total market value for autonomous cars and robotaxis can reach $2.5 trillion per year by 2040 [13]. In figure 3.1 it is possible to graphically see what has just been said.

The shift towards higher driving automation may suggest that inattention will not be an issue for much longer, but unfortunately this will not be the case in the expected future as fully automated driving is still a long way off.

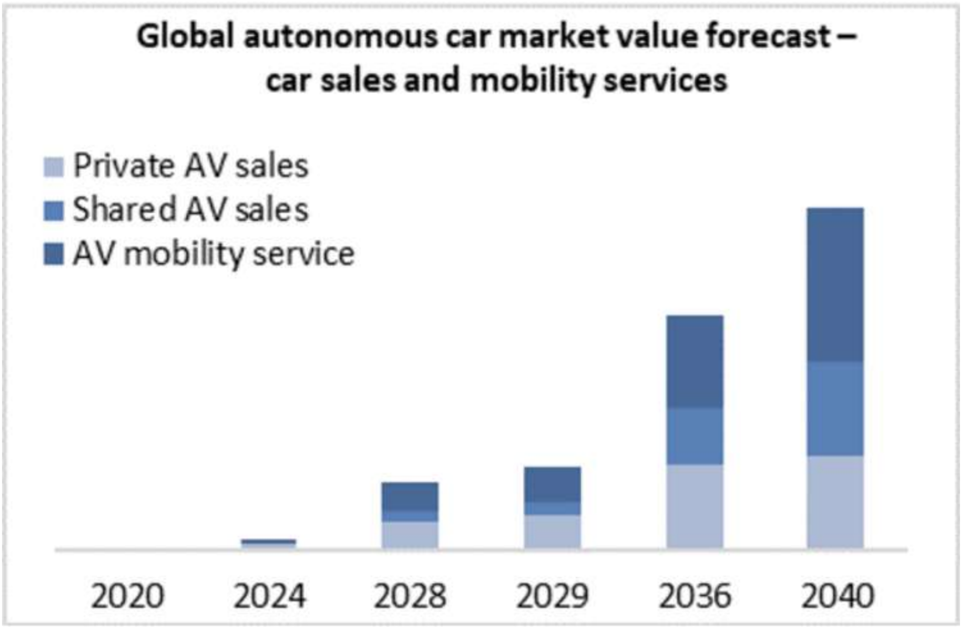


Figure 3.1: Market forecast of global autonomous cars and mobility services

Automated or “self-driving” vehicles are a future technology rather than one that you’ll find in a dealership tomorrow or in the next few years. Researchers expect that humans will continue to play a role in automated systems such as vehicles, even at higher levels of vehicle automation.

Among the most important DAS that nowadays can be found on any car, there are the Antilock Braking System (ABS), the Electronic Stability Control (ESC) and Traction Control System (TCS).

DAS are strictly linked to the use of proprioceptive sensors, which are able to measure most of the vehicle quantities, such as velocity, acceleration, wheel rotational speed and so on.

Thereafter, in order to enhance even more the level of assistance given to the human driver, exterioperceptive sensors, i.e. sensors that acquire measurements from the external environment (road, other vehicles, traffic etc.), were introduced in a new kind of systems, named Advanced Driver Assistant Systems (ADAS).

ADAS represent an evolution of DAS since they can help the human driver in performing either some critical tasks, such as an emergency braking, or the most common tasks, for instance maintaining a constant speed, parking and so on.

Indeed, the most known ADAS are Adaptive Cruise Control (ACC), Lane keeping Control (LKC), Park Assist (PA), Autonomous Emergency Braking (AEB).

ADAS constitute the first step towards the autonomous driving

3.1.1 A Taxonomy of Automation

The Society of Automotive Engineers (SAE) devised a classification for the cars based on the level of automation in turn based on the functionality of the driving automation system features [14].

The two lower levels of driving automation (1 and 2) refer to cases in which the human driver continues to perform part of the DDT (Dynamic Driving Task) while the driving automation system is engaged. These are therefore referred to as "driver support" features.

The upper three levels of driving automation (3 to 5) refer to cases in which the automated driving system (ADS) performs the entire DDT on a sustained basis while it is engaged. These are therefore referred to as "automated driving" features (or the vehicles equipped with them).

- 0. No Driving Automation: the human driver performs all driving tasks.
- 1. Driver Assistance: driver is assisted while performing some tasks. Although this, the driver is fully responsible of accomplishing every other tasks and monitoring the environment.
- 2. Partial Driving Automation: the system handles itself the acceleration, deceleration and steering. Still, the human driver will monitor the environment and act if needed.
- 3. Conditional Driving Automation: the car is able to drive itself and monitors the environment, but the driver must be ready to take control if required.
- 4. High Driving Automation: the car drives itself and occasionally asks for help to the driver, but it is able to handle the situation even if the driver's response is inadequate.
- 5. Full Driving Automation: the car is fully able to drive itself. Each scenario is correctly handled, so the role of the driver disappears.

These level definitions can be used describe the full range of driving automation features equipped on motor vehicles in a functionally consistent and coherent manner.

The generic term "driving automation system" refers to any Level 1 to 5 system or features that performs part or all of the DDT on a sustained basis.

While, we can say that a car has an automated driving system (ADS) if it belongs to classes 3, 4 or 5. So, the term ADASes refer more to class 1 and 2 and do not pretend to be substitutes of the driver, but assist the driver (indeed ADAS stands for Advanced Driver Assistance System).

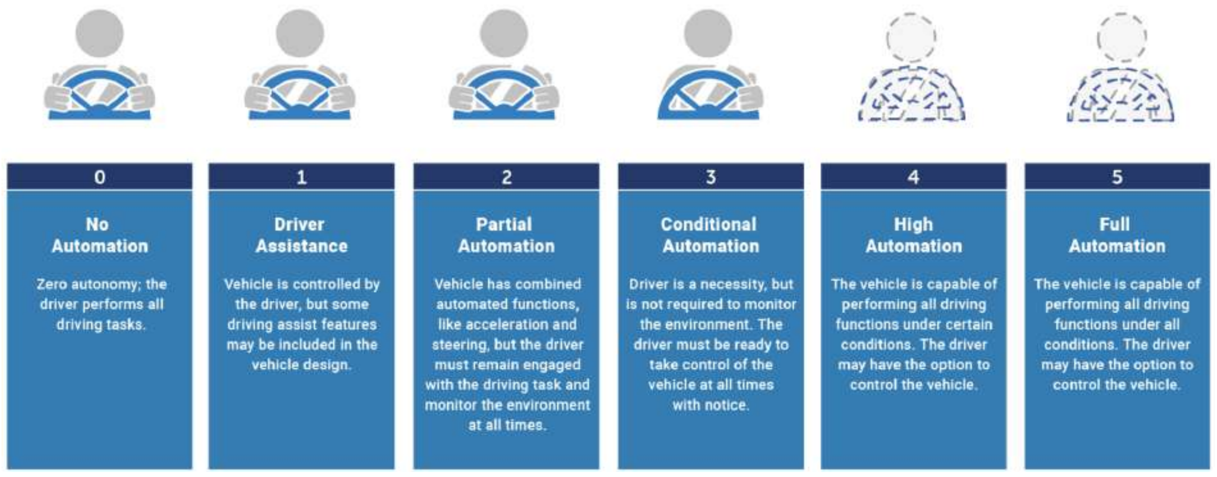

0
No Automation
Zero autonomy; the driver performs all driving tasks.
1
Driver Assistance
Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.
2
Partial Automation
Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.
3
Conditional Automation
Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.
4
High Automation
The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.
5
Full Automation
The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

Figure 3.2: SAE: Automation Levels

ADAS are systems to help the driver in the driving process. ADAS help the driver by signaling dangers and helping in performing maneuvers, without taking the direct control of the car (unlike Autonomous Driving Systems).

An ADAS faces those situations by drawing the attention of a distracted user (especially when the situation requires full attention of the driver), assisting the user while maneuvering, warning the user if it is trying to perform a risky or illegal action and provide a reasonable alternative.

3.1.2 Three Way of Distraction

In order to create a ADASes system, one of the most important aspect, to make the system understand, is when the driver is distracted. This is possible by developing specific classifiers able to identify distraction when occurs and so mitigate its effective by modifying accordingly the strategies of the system or even by providing a direct feedback to the driver.

A framework for discussing the sources of driver distraction is defined in [15] which tells us that there are several types of distractions including cognitive, visual and manual distractions, which may be distinguished from each other based upon the resources required to perform the task. The sources are:

1. **Visual**: such as when the driver takes his/her 'eyes-off-road' for some other task. So, visual distraction occurs when the driver is looking away from the road scene, even for a split second. EOR is one of the most common distractions for a driver. Examples of activities causing EOR are adjusting devices in the vehicle (HMI), looking towards other seats or outside. All generally result in the driver not looking straight ahead, which is what he/she needs to be doing for safe driving.

2. **Manual**: also called biomechanical distraction, in which the driver is doing something else in the car cabin with his/her hands not being on the steering wheel, for example, answer a call or send a text message, grab food and eat, or grab a beverage and drink, all while driving.

3. **Cognitive**: in which the driver is not processing the information required for safe driving. More specifically, refers to the "look but not see" situations when the drivers' eyes are focused on the forward roadway, but his/her mind is not. Cognitive distraction occurs when a driver is thinking about something that is not related to the driving task. In the driving context, while visual distraction can be summarized by EOR, cognitive distraction can similarly be viewed as "mind-off-road" (MOR). Typically, cognitive distractions can result from fatigue, conversation with a co-passenger, listening to the radio, or other similarly loading secondary tasks that do not necessarily take a driver's eyes off the roadway. This makes it one of the hardest distractions to detect as there are no visible clues whether the driver is distracted. Cognitive distraction is probably the most difficult type of distraction to assess because it is difficult to observe what a driver's brain (as opposed to hands or eyes) is doing. While also verbal distractions have been reported in the scientific literature, they were not significantly associated with accident responsibility [16]

The three way of distractions can occur independently or can co-occur while performing an activity. Therefore, the crash risk is higher for multitasking activities.

Earlier driver distraction was defined to be associated with any secondary task (task not related to driving), but this definition has changed over time with the development of complex In-Vehicle Information System (IVIS) and handheld devices.

So, how to correctly identify whether the driver is in a distracted driving state and to provide the necessary warnings for the driver to avoid potential safety risks, has become one of the most concerning issues, also because it represents a starting point for a good driving assistance system.

3.1.3 Ways to detect Driver Distraction

Driver distraction can be measured through many modalities due to the changes it causes in driver behaviours.

Driver behavior detection using driver status data, driving data, psychological data such as heart rate, pupil movement, vehicle acceleration or combination of them is another proposed solution for estimating driver distraction or the level of the driver's cognitive load.

These data can be collected from car data buses or external sensors such as cameras and eye trackers. Deep learning and other machine learning methods use these data to learn driving patterns and driver status.

There are various studies that have proposed different parameters to be used to detect distraction and we can divide the various categories of measurements for driver distraction detection and their descriptions in the following categorization:

1. **Driver Biological and Physiological Measures**: Measuring and inferencing from biological and physiological signals from the driver's body constitute a fundamental material that underlies the driver's cognitive distraction status. Physiological measures such as heart rate information, skin conductance, Electrocardiogram (ECG), electroencephalogram (EEG), galvanic skin response (GSR), Electrooculogram (EOG), respiration, electromyogram (EMG) and Heart Rate Variability and others, have also been used to detect driving distraction. Although, as evidenced and reported by S Pradeep Kumar et al [18] there is potential correlation to the EEG collected data and driver distraction. They, to record the EEG signals, use a 40 channel polysomnography (PSG) device from allengers provided with 21 EEG channels. More specifically, the latter study focused on 21 electrodes of the EEG because of its sensitivity to cognitive workload. In this work the alpha and theta sub bands were considered for predicting the normal and fatigue state of the drivers. So, they proved that occipital region and temporal lobe are the most important factor to identify fatigue and distraction. However, studies testing skin conductance and heart rate information showed only a weak relationship between these measures and distraction. Indeed no significant relationship was found between skin conductance or heart rate [17] and driver distraction.

2. **Vehicle Dynamics Data**: Significant effects of driver distraction are observed on driver's vehicle control in recent research, such as drivers adapting to drive at a slower speed to increase available response time when distracted, Longitudinal Control Parameter and Lateral Control Parameter are significant under distracted driving [19]. Chang Wang et al addressed the impacts of speed and naturalistic visual distraction on driving performance on a real road [20].
   Driving Performance as decision variables for driving distraction are observed by making drivers perform additional secondary tasks such as a cell phone conversation, co-passengers conversation, navigation control, and playing a radio with varying workload driving in a simulator and real environments [21] [22].
   As described by Kentaro Lio et al in [23] common measures of driving performance for detecting distraction are speed, lateral control, and reaction time. More specifically, in the context of visual distraction, drivers generally slow down when distracted by a visual stimulus, in fact lower driving speeds and posted speed limits were associated with phone handling [23] . This can be explained as a compensatory mechanism for the perceived risk, which can be decreased through reduced speed. Generally, visual distraction impairs lateral control because the driver needs to compensate for errors made when taking the eyes off the road, which leads to larger deviations in lane positioning and furthermore

steering control is also reported to be less smooth than in normal driving.
Reaction time is evaluated by several measures such as Stopping distance, Peripheral Detection Time (PDT), and Driver Response Time (DRT), Foot Transfer Time [24].
Sebastian Zepf et al [25] use vehicle dynamics data to calculate higher level features, such as number of crashes and number of infractions. For number of infractions they considered exceeding the maximum allowed speed, moving too slow (<50% of the allowed speed), and driving out of the lane as infraction cases.
Another important determining factor coming from driving performance is that drivers tend to increase the distance to the leading vehicle in the car-following scenario when they engage in cognitively demanding secondary tasks [26]. They showed a significant decrease in situational awareness due to cell phone use compared to routine driving, which was in line with reports of drivers missing signs and losing track of their location while engaged in a secondary task.
This suggests that drivers usually compensate for distraction induced by a secondary task with driving parameters.
A recent research by Johan Engström et al. [27] suggest us that driving under cognitive distraction increased time and increased steering angle variance in the time course from onset of steering control to lane change and increased steering angle variance in the time course after lane change.
Accordingly to this research, we can say that both cognitive and visual loads affect driving performance.

3. **Driver Behavioral Measures**: mostly identified by image and audio processing techniques such as eye data, hands detection, head rotation, head nodding, facial features expression, voice activity and so on.The driver behavioral measures are non-intrusive, real-time and fast at detecting the onset of abnormal driver states. Lisheng Jin et. al [19] use pupil diameter, horizontal and vertical fixation angles and Blink frequency as driver physical measures.
Trivially, the use of features such as head-pose and head-orientation could be a crucial factor in identifying visual distraction. However, they reported that this method has a high potential for false positives. This is because, even if a driver's head is tilted to the side, his eyes could still be looking on the road. Therefore it becomes necessary to add data of the eyes to those of the head to get more information. Eye data, for example, has a number of features like fixation duration, blink percentage, gaze deviation, Percentage of Eyelid Closure, and others. Hence, observing one or all of these features will depend on the purpose and requirement of the system. Imen Jegham et al [28] implemented driver distraction detection and classification using colour and depth map data from the Kinect sensor. Eye behaviour that is gaze detection and blinking, arm position, head orientation, and facial expressions are merged together and fed into a classifier for accurate sorting. For example, gaze tracking is the technique that allows estimating the direction where a person is looking at and from it we can identify the cognitive distraction [29]. As described by Agapito Ledezma et al [30] this problem, generally, can be faced through two different methods: Appearance-based methods rely on eye appearance where gaze estimation consists of extracting fea-

tures from the eye images and learns a mapping function that relates to the eye image features and gaze's positions; Model-based methods rely on a geometric eye model representing the structure and function of the human visual system. All of them refers to computer vision techniques. In many recent researches already pre-trained tools are used for the extraction of features such as OpenFace [32], , a pre-trained computer Vision tool which in turn is based on OpenCV an opensource library for computer vision. Papakostas et al. [31], for example, use OpenFace to extract from video the head-pose, eye gaze information, facial landmark coordinates and action unit (AU) presence as well as intensity values.

3.2 Distracted Driver Machine Learning review

In recent years, ML algorithms have been used for various purposes such as biodiesel research [33], Measuring and modeling the motor system [34], and Coronavirus disease (COVID-19) cases analysis [35]. Furthermore various approaches have been used to classify driver distraction in recent years.

A substantial amount of research has been conducted on supervised machine learning algorithms for distraction detection and all of them are characterized by their input data, their ground truth definition of driver distraction, and machine learning approach.

In general, input sources include driving behavior, head and eye tracking, and/or driver physiological measures. Prior algorithms have used one of two types of ground truth: binary (e.g., distracted cases and normal driving cases) and multiclass (e.g., cognitively distracted cases, texting cases, and normal driving cases).

Algorithms that leverage multiple sources of input and multiple classes of ground truth have the most power for inference because supervised machine learning algorithms can only learn from the data and labels provided in their training datasets.

There are several machine learning approaches including Support Vector Machines [36], Bayesian Networks [37], Decision Trees [38], Random Forests [39], and deep neural networks.

Hina et al. [46] presented the functionalities of an alternative ADAS. In this research, they modeled the driving context using ontological approach. The values obtained from a simulation are taken and passed onto the ontological template in order to produce a real-time driving situation in which the actual values are those obtained from the simulator.

After, machine learning is invoked to identify the sampled driving event. In more specific terms, they built a neural network with one hidden layer having 1,000 hidden neurons and a learning rate of 0.001, obtaining an accuracy of 75%.

McDonald et al. [47] analyze a set of driver performance (e.g., brake force, lane offset, speed, and steering angle) and physiological data (e.g., breathing rate, heart rate, and perinasal perspiration) using advanced machine learning approaches. In this research, they testing different machine learning algorithms including Random Forest, Decision Tree, Naive Bayes, SVM with linear kernel function and with radial kernel function, and some Neural Network architecture trained through data collected from 48 participants using a driving simulator.

Yanli Ma et al. [22] selected forty participants to conduct the driver distraction experiment when operating IVIS, and the data of driving performance indicators such as eye movement, speed, were collected. They create a real-time detection of distraction system based by using support vector machine, and three kernel functions of the model were conducted comparative analysis and validation. The results show that SVM models can effectively evaluate the degree of drivers' distraction. At the same time, when the Radial Basis Function is used as a kernel function, the accuracy for recognizing driver distraction is 89.9%

Thanks to the availability of computational power and the emergence of big data, deep learning methods have recently received lots of attention. Referring to the transportation field, deep learning has applied to several areas including macroscopic traffic conflict prediction [40] [41], transportation planning [42], traffic surveillance and congestion detection [43], behavior prediction [44] and signal control [45].

In the context of driver distraction there are many recent studies using deep neural networks and various feature sets as inputs.

Kouchak et al. [53] conducted an experiment in eight different driving scenarios and the same road to produce an artificial intelligence-friendly dataset and used this to train a traditional neural network and a multilayer neural network in order to classify driving data to eight groups. Based on our observation and collected data in this experiment, driving context has an effect on the severity of driver distraction. It means, doing a distractive task in adverse conditions results in a higher level of driver distraction and has a more negative effect on the driver's performance. As a result, we believe that detecting the context of driving and driver distraction status together provides more valuable information about the cognitive load and distraction level of the driver compared to methods that only detect if the driver is distracted or not. They reached 93% test accuracy using deep neural networks.

Tamas et al. [54] propose a new three-part framework for driver distraction detection. A dataset of images with a variety of driver's ethnicity, gender, and camera positions was used in this framework. The first part is a convolutional deep neural network that is used for high abstracted feature extraction. The second part is a max-pooling layer that decreases the dimension of features, and the last part includes six fully connected layers and a SoftMax layer. The best test accuracy was 95% and the average accuracy was 80% per class. This system uses a multi-class classification process to map the input observation to a driver state. This approach needs some cameras to collect driving data. In fact, our approach detects driver states, distracted and not distracted, using Can-bus data and a front camera.

Li et al. [49] discuss a lane detection method using deep neural networks. Two deep neural networks were used in the model. A multitask deep convolutional network has been used to detect the target considering the region of interest. A recurrent neural network, which is adapted to be used as structured visual detection, is another deep network that's used in this model. The first network models lane structures, and the recurrent neural network detects the lane boundaries without any prior knowledge. Both proposed networks outperform conventional detectors in the practical traffic scene. Images of the driving environment were used as inputs of these two deep networks, so they can be used in cars that are equipped with cameras such as autonomous cars. This model completely relies on the collected data by external devices.

Botta et al. [50] proposed a non-intrusive approach to detect visual and manual distraction based on a Single Layer Feedforward Neural Network (SFLN). They considered as features vehicle dynamics data and environmental data, without using eye-tracker information. First of all they carried out a phase of extraction of the characteristics with a moving average that detects average, std, min, max, etc. Finally, they propose an original approach which benefits from matrix sparsity, showing lower computational times with respect to standard implementations with a single layer feedforward neural network trained through pseudo-inversion. Moreover, theirs genetic based technique outperforms the usual one for the majority of the considered subjects.

The detailed data from the NDS SHRP-2 Wildlife Driving Study was analyzed to obtain information on disability and distracted driving by Arvin et al. [51] . The risks associated with carrying out activities other than driving are quantified and discussed in terms of critical safety events. By monitoring driver biometrics (in terms of distraction), vehicle movements and driving volatility after a phase of extraction of the characteristics to obtain volatility indices at the event and temporal level (i.e. velocity and variations in longitudinal and lateral acceleration). Finally, the raw data and the extracted functionalities are fed into the deep learning phase. Deep Neural Net (NN), Convolutional Neural Net (1D-CNN), Long Short-Term Memory Recurrent Neural Net (LSTM RNN) and 1DCNN-LSTM models are developed to classify events as basic or critical events and evaluate performance models. They reported that 1DCNN-LSTM outperforms other deep learning methods presented and can detect abnormalities in driving, which can lead to crashes and near misses.

As reported in a very recent research by Kashevnik et al. [52], who have done a literature review and a framework to address the problem of driver distraction, at the moment there are no non-intrusive way to measure the driver mental condition. Using the intrusive methods are not applicable for the drivers since every time the driver start his/her trip it is uncomfortable to wear some devices. This is the starting point and the overall objective of our research in this thesis.

Chapter 4

Dataset Description and Analysis

4.1 Driving Simulator and Data Collection

Within Use Case 2 of the Next Perception Project (Section 1.2), the Demonstrator UC2D1 will be involved in Pilot 1 as a Driving Simulator by RE:Lab. The pilot will focus on the evaluation of the demonstrator's performance in monitoring and measuring the Driver Complex State (DCS), in particular looking at the technical feasibility, efficiency, accuracy, and user acceptance.

In the first cycle, the pilot will focus on collecting data needed for the development of the monitoring solutions and each of the modules to be integrated into the Driving Simulation Engine; its evaluation phase will focus on technical feasibility and will take place in a simulation environment.

The potential challenges identified in the context of the pilot, at a technology level, are:

1. The combination of a set of different monitoring components and smart sensors (e.g., camera-based distraction and emotion monitoring, vehicle data-based cognitive and behavioral distraction monitoring, etc.) to develop an unobtrusive Driver Monitoring System that will provide a reliable and accurate "fitness to drive" condition. The goal will be to infer a combined cognitive/emotional state of the driver, and also the activities and positions of occupants inside the vehicle cockpit (including the driver).
2. The coverage of more scenarios, in particular two, situations and conditions, with respect to the current state of the art (e.g., the combined impact of cognitive states and emotions at driving).
3. The use of heterogeneous data sources.

4. The improvement of monitoring systems performances against the systems currently on the market.
5. The design and development of appropriate multimodal and interactive HMI solutions for Automated/Autonomous Driving Functions (ADFs), which take into account the overall state of the driver, including emotions (emotional design).

The pilot will take place in Reggio Emilia, Italy, at RE:Lab's facility where the driving simulator (based on SCANeR Studio 1.7 platform, including features for sensors simulation, ADAS, and automated driving functionalities) is settled, with real users and will follow two predefined scenarios of simulation.

To highlight the decisive role of the DMS and to transparently communicate the main operating logic of this system, a dashboard has been prototyped that provides at a glance the indication of the output of each component of the Driver's Complex State (emotional, cognitive, and visual distraction, excitement) in a given instant, that is, it allows to identify the trends of the outputs of the different classifiers.

Together with these trends, the dashboard displays the Fitness to Drive index instant by instant: since this index is linked to the various components of the Driver Complex Status, through the dashboard it is possible to understand why the driver is judged unsuitable by the system or how suitable for driving, i.e. it is possible to see the link that exists between the Fitness to Drive index and the various classification outputs relating to the driver's status.

We emphasize here that the dashboard is not intended to be used by the driver, but is a demonstration tool, particularly useful for developers for demonstration purposes and to verify the operation of the various software modules and their integration.

The Next Perception Driver Monitoring System, in fact, integrates several classification modules. For demonstration purposes, it is useful to see the output of this integrated system by varying the input data.

As part of a hypothetical demo, it is imagined that starting from test data appropriately collected, it can be demonstrated through the dashboard how the Driver Monitoring System returns the expected results in terms of the Fitness to Drive index.

Pilot 1 involves 26 participants and is led by RE:Lab. Besides leading the pilot, RE:Lab will be responsible for the test setup and the integration of the modules into the simulator. Each scenario was approximately 35 mins long.

A CSV file was written from the simulator to store all data and an mp4 file was recorded to generate a front RGB video for each participant. The frequency of vehichel dynamics data collection was 20 Hz (1 data-point each 0.05s), which is the output rate of the vehicle CAN bus while the frequency of RGB camera data collection was 30 Hz (1 data-point each 0.33s).

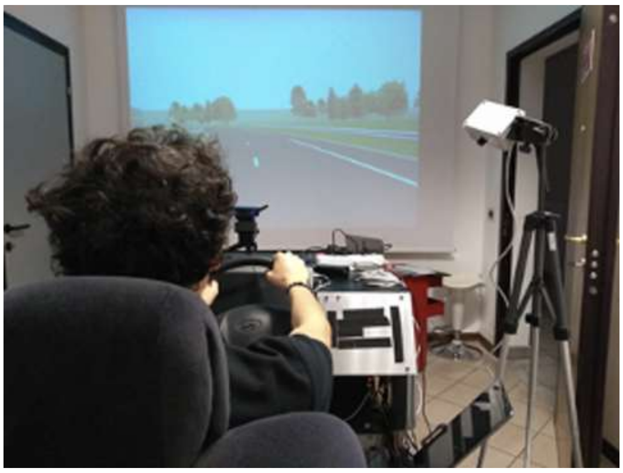


Figure 4.1: RE:Lab's Driving Simulator

The test focuses on eyes-off-road condition and cognitive distraction, and in each phase in which there is a distraction, while the user drives carefully, at random time intervals, 8 different periods of distraction are performed.

The test is divided into 4 parts (each of about 8 minutes):

1. **Baseline B**: safe driving, the only task is to drive and there is no kind of distraction;
2. **Visual Distraction V**: Participants were asked to select on the HMI the red figure out of a total of 4 different colored figures; the user wait for an acoustic signal and when it occurs, turn around with their head to interact with the HMI (causing visual distraction)
3. **Cognitive Distraction C**: participants were presented with a sequence of numbers, and were asked to indicate when the current number matched the one from n steps earlier in the sequence. For our experiments we set N=2. This is the so-called N-Back test. This distractor aimed to challenge exclusively the Cognitive capabilities of the subjects while driving. N-Back is a cognitive task extensively applied in psychology and cognitive neuroscience, designed to measure working memory.
4. **Cognitive and Visual Distraction D**: users must listen to a sequence of 7 numbers dictated by the system, and at the end, they must turn around and interact with the HMI by selecting between 4 possible digits shown, the one heard.

The order in which the different phases were followed is different. Specifically, the simulator divides the test into 4 different orders (BCVD, DBCV, VCBD, and DVCB) and randomly chooses the execution order for each participant.

The driving circuit is shown in figure 4.2.

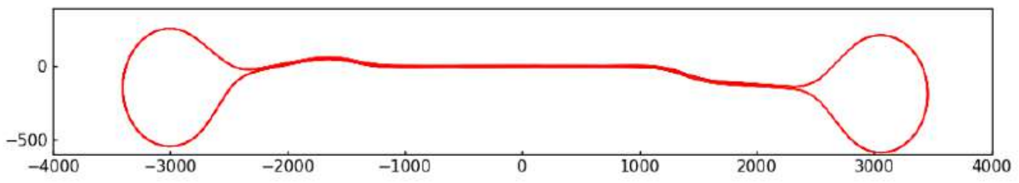


Figure 4.2: Driving Circuit

All the time instants in which there is a visual distraction (eyes-off-road) and in which there is cognitive distraction are manually annotated in different variables, where 0 indicates that in that frame there isn't a distraction while 1 indicates that there is a distraction. So that they can be used as labels for the training phase.

At first, the tests of some users for which the data are damaged (either for the video or for the vehicle dynamics data) were eliminated (for example users 1 3, and 12 have a missing part of video or 9 and 17 that have vehicle dynamics recorded data damaged)

A serious problem was encountered with acquired data. During the Baseline driving phase, in which the only task should be driving, some users, due to boredom, spoke to other people in the room that was not well isolated. In order to reduce the bias induced by these 'incorrect drivers' some of these have also been eliminated.

4.1.1 Vehicle Dynamics Data

The vehicle dynamics data are retrieved through Controller Area Network (CAN)-Bus in cars while simulators have inbuilt systems for the same. They are focused on in-vehicle processing of information about the driver and the vehicle and were sampled at 20 Hz so a value is recorded for each variable every 50 ms.

Firstly we have eliminated all unreal features such as the absolute position (x,y, and z) in the simulator environment and all the irrelevant features such as the GearBox Mode, Engine Status, and Clutch Pedal which respectively were useless that in the simulator environment GearBox Mode was always automated (so no clutch pedal too) and engine status was always on.

The remaining variables sampled in this study from Vehicle Dynamics Data are given in Table 4.1 with a short description and unit of measurement. As you can see, Vehicle motion (acceleration and speed) is generally described in terms of the velocities: forward, lateral, vertical, roll, pitch and yaw in the vehicle-fixed coordinate system as referenced to an earth-fixed (inertial) reference frame, respectively u (longitudinal velocity), v (side velocity), w (normal velocity), p (roll velocity), q (pitch velocity) and r (yaw velocity).

VARIABLE NAME	DESCRIPTION	UNIT OF MEASUREMENT
Accel	The acceleration of the vehicle's CoG frame. Specified in x, y, z, heading, pitch, roll.	m/s2
accelPedal	accelerator pedal position.	Float from 0 to 1, 0 is depressed
brakePedal	brake force	Float from 0 to 1, 0 is depressed
engineSpeed	engine speed	rad/s
Speed	The speed of the vehicle's CoG frame in the chassis referential. Specified in this order heading, pitch, roll.	m/s
steeringTorq	Steering wheel torq, positive to the left.	N/m
steeringWheelAngle	Steering wheel angle, positive to the left.	rad

Table 4.1: Vehicle Dynamics

A vehicle is modelled in two main parts: the unsprung mass that takes into account for the mass of the wheels, part of the suspension and other component directly connected to them, and the sprung mass that is basically the vehicle total mass supported by the suspension.

The Society of Automotive Engineers (SAE) has introduced standard coordinates and a notation to describe vehicle dynamics that are widely used (figure 4.3). Here the earth-fixed reference frame's axes are referred as X, Y and Z and the vehicle reference frame's axes as x, y and z. The angles are roll, pitch and yaw.

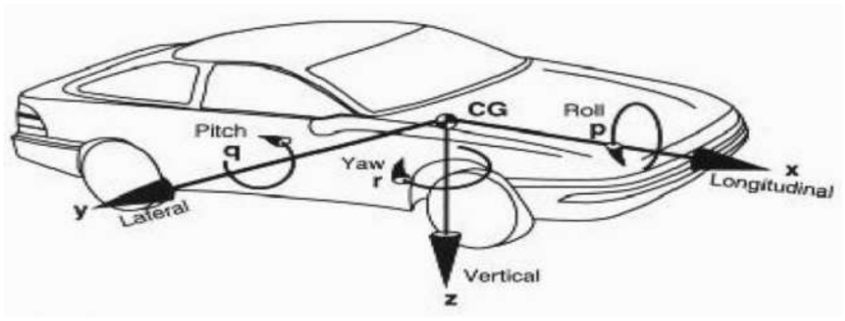


Figure 4.3: Vehicle Axis System (SAE)

4.1.2 Road Information

External monitoring sensors include external sensors that monitor the road and obstacles outside the vehicle. As for Vehicle Dynamics, they are collected using the CAN bus in the driver simulator (and from the external sensors in a real environment such as a camera).

These sensors elaborate the raw data in the respective hardware modules, and communicate the processed data directly to the ECU, in order to collect extra vehicle data according to the road network and status for each wheel of the vehicle.

The processed data allows us to reconstruct an abstract view of the external world.

Firstly we eliminated unreal variables (that do not exist in a real environment) such as ground Index, laneType under the wheel, the id of the object under the wheels, the IDs of the Roads, the IDs of the lanes, the Road Gap that indicates the lateral distance of the vehicle regarding the middle of the road or the absolute position of each wheel in the vehicle frame.

Also, unnecessary variables were deleted such as grip for each wheel. The remaining variables given by Road Information Data are reported in Table 4.2 with a short description and unit of measurement

VARIABLE NAME	**DESCRIPTION**	**UNIT OF MEASUREMENT**
rotation	Rotation of each wheel around the X, Y, and Z-axis.	rad
speed	The angular speed of each wheel.	rad/s
angle	The absolute angle of this wheel.	rad
vhlSx	LSR (Longitudinal Slip Ratio) of the wheel.	rad
vhlDelta	Wheel Slip Angle (angle in radians between actual speed vector of the wheel and their orientation).	rad
laneGap	The lateral distance of the vehicle regarding to the middle of the recognized lane.	m
roadAbscissa	The curve distance of the vehicle along the current road.	m
roadAngle	The angle between the axis of the road, and the heading of the vehicle.	rad

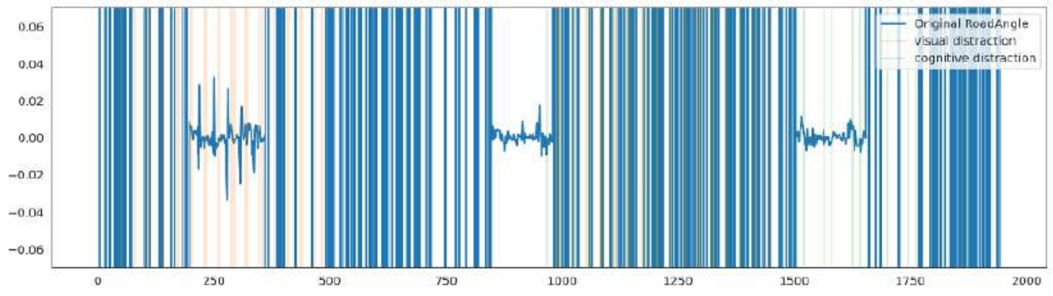
Table 4.2: Road Information

It is important to note at this point how preprocessing operations were carried out to make the features usable and meaningful.

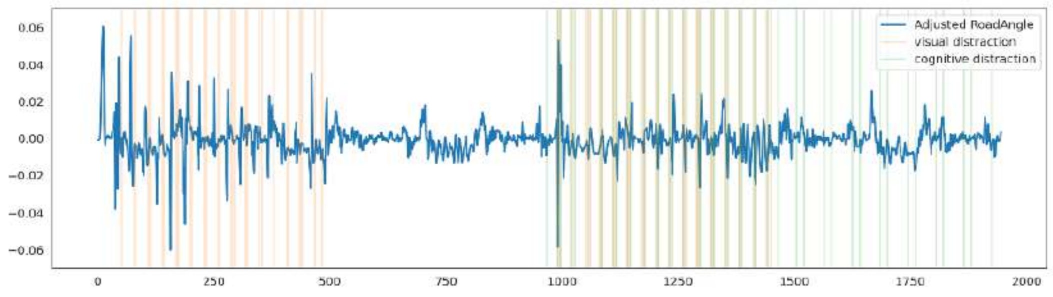
For example for the roadAngle variable. It represents the direction that the vehicle is taking with respect to the road; the angle is measured precisely with respect to the direction that the vehicle is taking and therefore is composed of oscillations between $-\pi$ and $+\pi$ ($-180°$, $+180°$). Each change of direction indicates a jump between - and +, as you can see in figure 4.4a.

So the goal is to eliminate this oscillation in such a way as to obtain the oscillation of the vehicle with respect to the road insensitive to the fact that the vector of the road and the vehicle can be rotated of 180°.

To do this, we have subtracted π in all those points where the value is greater than $\pi/2$



(a)



(b)

Figure 4.4: Two roadAngle plot of the same driver (a) Data are not very significant and do not reflect the induced phases of distraction (b) After processing, the same feature has interesting correlations with visual distraction, as they correspond to points where one has moved (with respect to the center of the road) without carrying out a maneuver due to the circuit.

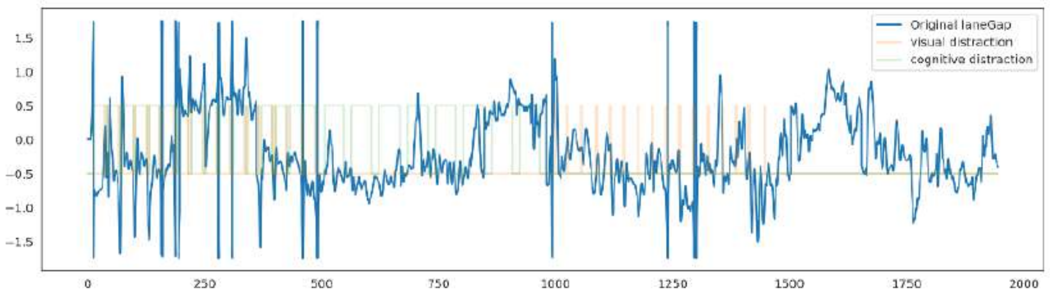
and instead added π in all those points where the value is less than than $-\pi/2$. The obtained feature is significant with visual distraction, as you can see in figure 4.4b.

Another feature that caught our attention was laneGap. This represents the lateral distance of the vehicle regarding the middle of the recognized lane.

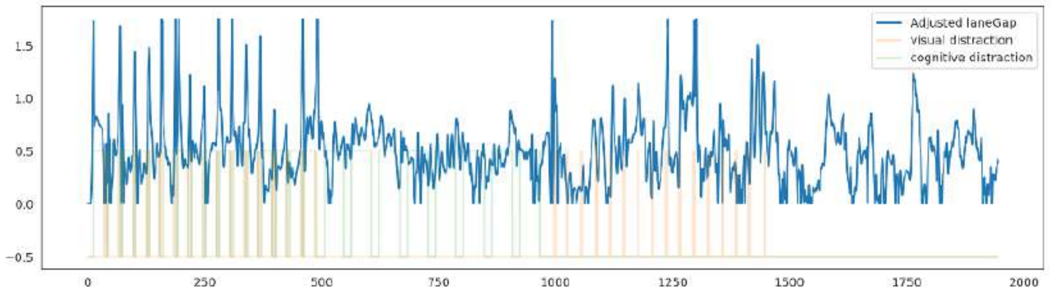
This feature is significant with the visual distraction detection because whenever there is a visual distraction we will most likely get closer to the recognized line or even pass it, then the distance increases until the next line is recognized (if passed the old one) and in this case, there is a jump.

Therefore, these jumps help to understand when the driver is not looking forward. However, it cannot be used as the only information as lane changes deriving from exits or roundabouts (present in the motorway context of the simulator) appear to have the same meaning in the data, see figure 4.5a

It is therefore necessary to discriminate through the set of all the features available when the change of line is due to the road circuit or to a distraction. In order to make it more meaningful, it has been transformed so that the value is only positive and carries the peaks to the points where there is a lane change, see figure 4.5b.



(a)



(b)

Figure 4.5: (a) Data is very confusing and misleading (b) After processing, the same feature has interesting correlations with visual distraction, as they correspond to points where drivers exceed the lane and invade the adjacent lane

4.1.3 Internal Camera

In order to detect features from the video, was used a trained facial recognition system. A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source.

The idea of facial recognition technology firstly appeared in 2014, when Facebook announced its DeepFace.
Now, in 2021, most facial recognition algorithms exceed the most accurate algorithm from late 2013.

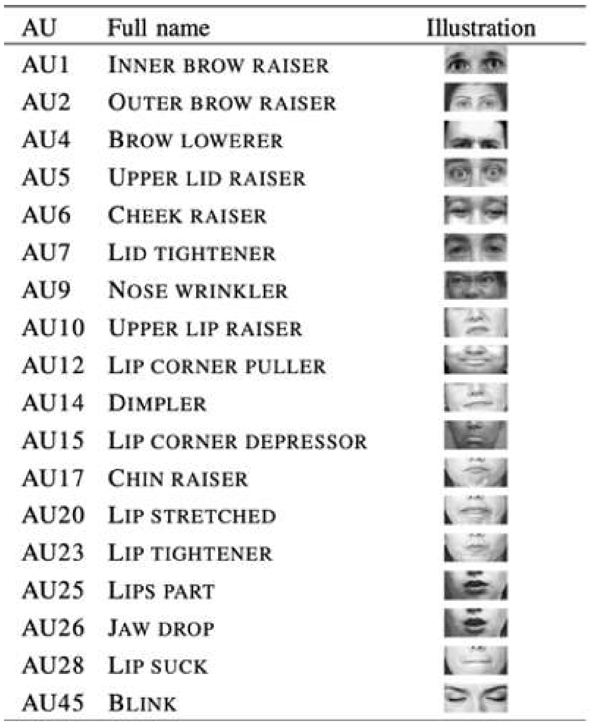
There are several face recognition algorithms, some of them are:

1. MediaPipe [55]: is one of the most widely shared and re-usable libraries for media processing within Google based on machine learning. It is a framework for building multimodal(e.g video, audio or any time series data),cross-platform (i.e Android, IOS, web, edge devices) applied ML pipelines. Some of notable applications of MediaPipe are Face Detection, Face Mesh, Multi-hand tracking, Hair Segmentation, 3D Object Detection and Tracking

2. MMPose [56]: open-source toolbox for pose estimation based on PyTorch. MMPose implements multiple state-of-the-art (SOTA) deep learning models, including both top-down & bottom-up approaches

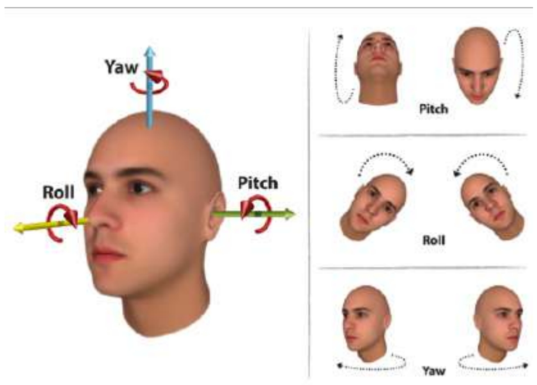
3. Deepgaze [57] : is a library for human-computer interaction, people detection and tracking which uses Convolutional Neural Networks (CNNs) for face detection, head pose estimation and classification. Deepgaze is based on OpenCV and Tensorflow, some of the best libraries in computer vision and machine learning.

4. OpenFace 2.0 [58]: a tool intended for computer vision and machine learning researchers which is capable of very accurate facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. The algorithm behind OpenFace was based on OpenCV and uses Convolutional Neural Networks.



AU	Full name	Illustration
AU1	Inner brow raiser	
AU2	Outer brow raiser	
AU4	Brow lowerer	
AU5	Upper lid raiser	
AU6	Cheek raiser	
AU7	Lid tightener	
AU9	Nose wrinkler	
AU10	Upper lip raiser	
AU12	Lip corner puller	
AU14	Dimpler	
AU15	Lip corner depressor	
AU17	Chin raiser	
AU20	Lip stretched	
AU23	Lip tightener	
AU25	Lips part	
AU26	Jaw drop	
AU28	Lip suck	
AU45	Blink	

(a)



(b)

Figure 4.6: (a) AUs monitored by OpenFace 2.0 (b) Yaw,Pitch and roll of head-pose

The tool that we have used is OpenFace 2.0 as it is very fast, and also returns, unlike the others, in addition to the head pose and eyes gaze, the Action Units (facial muscle movement).

The algorithm behind OpenFace, explained in [58], is briefly described below. The face is detected by the Multi-Task Convolutional Neural Network (MTCNN) face detector, which was trained with WIDER FACE and CelebA.

In contrast to other facial detectors, which use HoG-SVM, Haar–Cascade, etc., MTCNN can detect frontal, profile, and highly-occluded faces by using a single detector. The bounding box of the detected face is used to detect facial landmarks and facial behavior recognition.

The Convolutional Experts Constrained Local Models (CE-CLM) detect 2D Cartesian coordinates of 68 facial landmarks. The CE-CLM internally has data on the 3D coordinates of facial landmarks, and head direction (yaw, pitch, and roll) is estimated by the perspective-n-point algorithm. Using facial landmarks detected by CE-CLM and HoG features extracted from the aligned detected face image of 112×112, the intensity and presence of facial Action Units (AU) are estimated by support vector regression.

Output features provided by Openface 2.0 include yaw, pitch, roll, (head Rotation, figure 4.6b , Eyes gaze direction (averaged for both eyes and converted into more easy-to-use format), and Facial Action Unit (AU) presence as well as intensity values.

We experiment with both individual and combinations of those features and we conclude that AU intensity values were the ones encapsulating the richest amount of information for our scope. To describe AUs we first need to introduce the Facial Action Coding System (FACS).

FACS is a framework designed to group facial movements based on their appearance on the human face. This grouping depends on slight instant changes in facial appearance caused by individual facial muscles. AUs are the individual units used by FACS to code complex facial expressions.

Thus, AUs can be seen as a mid-level representation of facial expressions, providing a higher level of information than just a group of facial landmarks but being much more descriptive than an affect-based classification or regression model.

Openface provides AU intensity in the form of a continuous variable for 17 different AUs. Intensity values may range from zero (AU is not present) to five (maximum intensity). The AUs monitored by Openface can be seen in Figure 4.6a. For example for AU26 and AU45, corresponding to mouth opening and eye blink was estimated to be higher than a value between two and three.

To make the head pose even more precise, and given the visual distancing context, in which the head turns to perform a secondary task, it was decided to create a single variable starting from Yaw, pitch and roll.

In this case we have calculated the sum of the squares of the radians of the Yaw and the Pitch (excluding the roll that hardly happens in the car environment). In this way a more precise value is obtained which indicates a degree of rotation of the head.

In addition, cleaning operations were also carried out for the AU. In this case the intensity value of each AU was set to 0 if the general confidence of OpenFace did not exceed 0.8.

Furthermore, the intensity value is multiplied by its specific confidence value so as to increase or decrease it based on the associated confidence.

4.2 Merging: Time Lagged Cross-Correlation

One problem encountered was about synchronizing of video which did not start at the same time of vehicle dynamics data recorded. As described [59], there are many techniques to measure synchronous between time series data: Pearson correlation, time-lagged cross-correlations, dynamic time warping, and instantaneous phase synchronize.

In this thesis, a time-lagged cross-correlation method was used between the annotated visual distraction columns and the Head Pose extracted from OpenFace.

Time lagged cross-correlation (TLCC) can identify directionality between two signals such as a leader-follower relationship in which the leader initiates a response that is repeated by the follower.

There are a couple of ways to investigate such a relationship including Granger causality, used in Economics, but note that these still do not necessarily reflect true causality.

To allow that shifting, we compare the labeled data on visual distraction with the degree of rotation of the head found in OpenFace, and through a 1D Gaussian filter, the TLCC gets the offset value between the video and the vehicle dynamics data.

In our case, as shown in figure 4.7, TLCC is measured by incrementally shifting the HeadPose time series vector (blue) and repeatedly calculating the correlation between it and target Visual Distraction (red) and after selecting the offset at the peak of the correlation.

If the peak correlation is at the center (offset=0), this indicates the two-time series is most synchronized at that time. However, the peak correlation may be at a different offset if one signal leads another. We have implemented a cross-correlation function using pandas functionality.

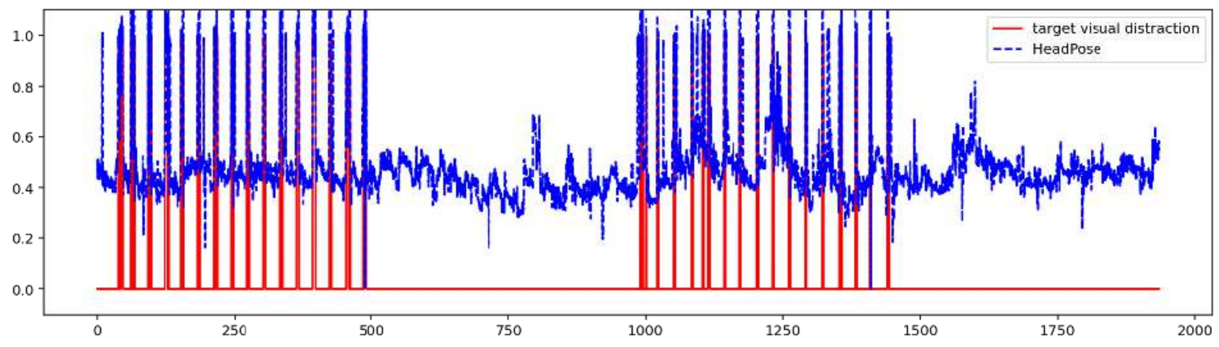


Figure 4.7: Time Lagged Cross-Correlation: alignment of target of visual distraction and the head-pose.

Subsequently, the various sources of information were merged, after downsampling of video data, in order to use it as a single dataset composed by 151 features.

Chapter 5

Features Selection and Extraction

5.1 Features Selection

Feature selection is a process where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.

One of the key factors for feature selection is to eliminate features that are not informative, which involves selecting features that have less correlation to each other and high correlation with class labels. It helps to remove redundant, misleading, or useless features which do not contribute to the model performance.

Furthermore, perform feature selection on the data before using them for training machine learning models allows to reduces overfitting, in that less redundant data means less opportunity to make decisions based on noise, but also improves accuracy, in that less misleading data means modeling accuracy improves and also reduces training time, in that less data means that algorithms train faster.

There are two main types of feature selection techniques: wrapper and filter.

1. Wrapper: This methodologies use a learning algorithm in a selection process to detect features that are useful in the context of other features: a feature subset of all features is searched which enhances the performance by the learning algorithm.
2. Filter: This methodologies evaluate the goodness of each feature according to some criterion and select features ranked with high scores. Filter-based feature selection methods use statistical measures to score the correlation or dependence between input variables that can be filtered to choose the most relevant features.

Finally, there are some techniques called Intrinsic that are when there are Algorithms that perform automatic feature selection during training. For example with a decision tree.

The wrapper technique has an issue when features are strongly correlated with each other; as a result, when one feature is garbled, the other correlated feature is able to provide the necessary information to the model. This results in lower importance being given to the correlated features.

This problem has been avoided by clustering correlated features and choosing one feature from each cluster for the correlation task as it is able to provide similar information to the other features in its cluster.

In this thesis, hierarchical clustering was performed on the features using Spearman rank-order correlation and a threshold was chosen to pick a single feature from each cluster.

When features are collinear, permuting one feature will have little effect on the model's performance because it can get the same information from a correlated feature.

One way to handle multicollinear features is by performing hierarchical clustering on the Spearman rank-order correlations, picking a threshold, and keeping a single feature from each cluster.

First, we plot a heatmap of the correlated features (figure 5.1) and after, hand-select an appropriate cut-off on the dendrogram ward in order to collapse repetitive features.

As shows figure 5.1 there are 3 main cluster, and below are shown manual selected features for each sub-cluster based on recent research criteria highlighted by recent research reported in Section 3.1.3.

The orange one shows a very strong relationship between the variables it contains, this is because for example steering torq is a feature derived from other information which in this specific case has a very strong correlation between them. For this cluster, only the speed vector y and the steering torq.

Going to the end we see that also the last cluster shown (the red one in the figure) contains strong correlations since we are correlating values referring to the front wheels with more refined features that calculate the steering angle. In fact, the wheel angle, the steering wheel angle, and the acceleration vector y were chosen from the second cluster.

For the remaining features, it was not possible to further specify other strong correlations because there are values that are not strictly correlated. In fact, they are collapsed into a single large cluster, represented with the color green.

To make a selection of some features, it was necessary to choose a cut point, and for each sub-cluster created, a variable is chosen.

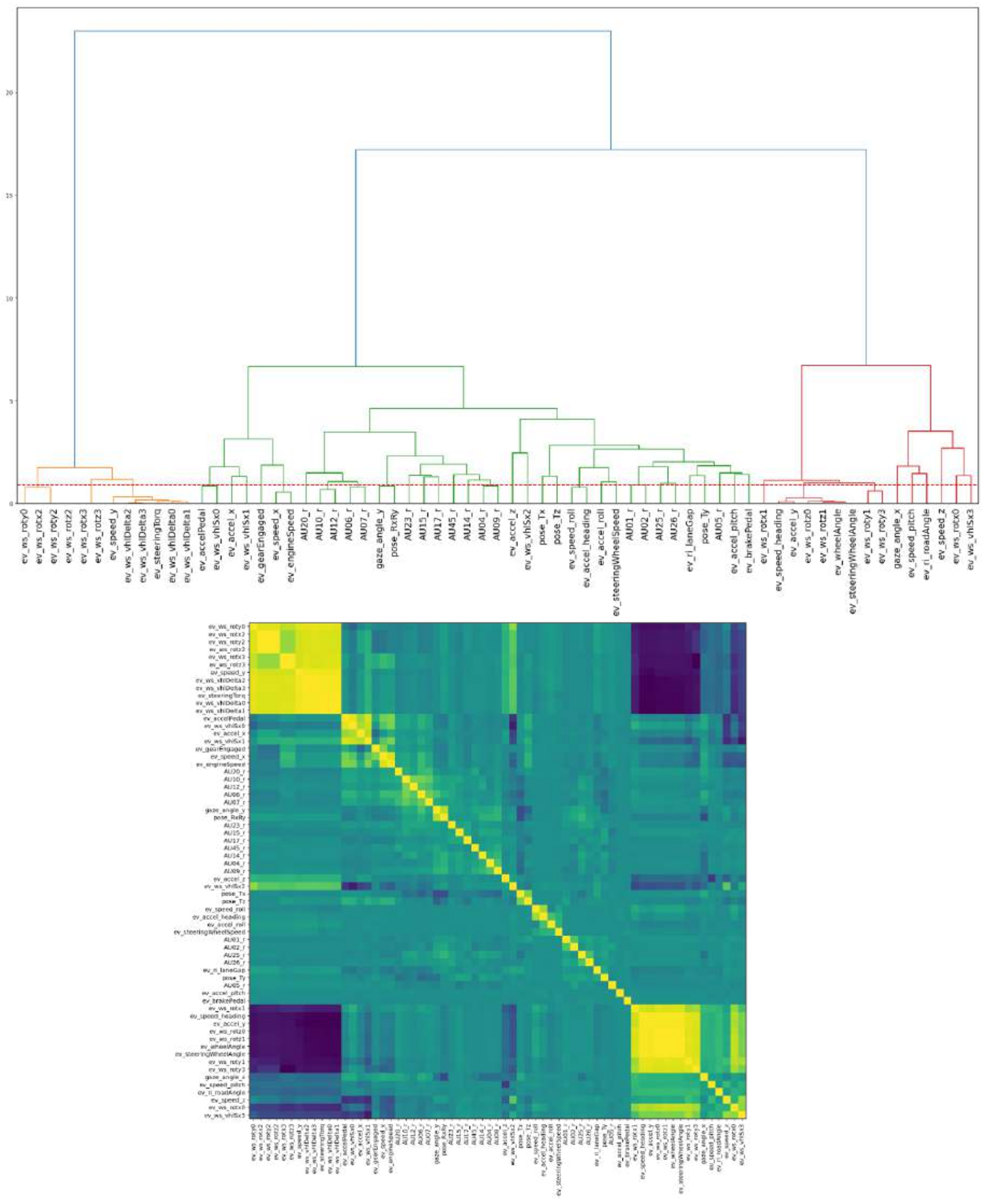


Figure 5.1: Hierarchical clustering and heatmap using Spearman rank-order correlation on all features

This large cluster with the cut at a certain level we can further subdivide it into sub-clusters.

Starting from the first sub-cluster we obtain, and going to see the heatmap as well, we realize how there is a strong correlation between the acceleration pedal, the acceleration and speed vectors x, and the Longitudinal Slip Ratio of the wheel 0 and 1 (i.e. front wheels) but since the latter is the slipping behaviour of the wheel of an automobile it can be replaced by the acceleration and speed vector x and the accelerator pedal.

Furthermore, it is still possible to note in this sub-cluster how the features of the acceleration vector x and engine speed are strongly correlated. This is trivially true as the acceleration vector x depends on the engine speed and incorporates it. In fact, even the latter was discarded.

Seeing again the next sub-cluster in the dendrogram and heatmap we see a much weaker correlation between the features except for the road Abscissa and the speed pitch. Between the two it was decided to discard the road Abscissa as the speed pitch represents a higher-level feature.

The vector z of the speed (i.e. the lateral one) is correlated, albeit slightly, with the direction of the eyes and the road angle. However, being weak correlations, it was decided to keep them all.

Continuing we see a sub-cluster made up of 5 features that refer to the Action Units recovered from the front camera. Let's see how for example AU10 is somehow correlated with AU12 (in fact in the heatmap the correlation takes on a light green color).

This is trivially given by the fact that the two action units refer to muscle movements of the lips, respectively upper lip raiser and lip corner puller, very similar to each other and therefore tend to occur together. In fact, noting within the heatmap we can see how even AU25, which always refers to the lips, although it has not been clustered together with the others, takes on a greenish color in correspondence with AU10 and AU12.

Furthermore, since the Action Units have continuous values between 0 and 5 which indicate the intensity of the reference action, and since it is possible that more than once they are worth 0, the correlation between them is not very consistent since using a simple correlation of spearman two vectors containing many zeros and a few different values would be very correlated. We have this problem with all the features that tend to remain 0 for a long time and only take effect at times (such as the brake pedal).

In recent research, as reported in Section 3.1.3, the correlations between the Action Units and distraction have been analyzed. Some of these appear to be more significant than the others but in general each can contribute to the identification of the distraction.

Due to this and also to the fact that the AUs indicate different facial muscle movements, it was decided to keep all the Action Units extracted from the video.

Continuing with the features in the large red cluster, thus excluding the AUs, we still see the correlation between, for example, the head pose and the gaze of the eyes considered absolutely relative to the camera (clearly obvious).

Although there is this correlation, we have chosen to keep both variables as in reality they express two completely different and partly independent features (for example I can look away without turning my head or vice versa I can keep my gaze fixed by turning my head).

Two further features that have attracted attention are the acceleration vector z and the LSR of the rear wheels. They seem to belong to the first cluster (i.e. the orange one) since the correlation corresponding to its features is higher (in fact they tend to be green). For this reason, we have chosen to keep only the acceleration vector z.

Other features that we encounter, all have a low correlation between them and therefore we have chosen to keep them all. In addition, many of these are being used in recent research to identify distraction. Some of the latter are, in fact, speed and acceleration roll, steering wheel speed, lane gap, and brake pedal.

Furthermore we have applied an automatic feature selection through a function called leaders. It works by going, for each cluster (cut at the threshold level), to returns the root nodes in a hierarchical clustering corresponding to a cut defined by a flat cluster assignment vector, and for each flat cluster of clusters, this function finds the lowest cluster node in the linkage tree such that:

1. Leaf descendants belong only to flat cluster
2. There does not exist a leaf that is not a descendant and that also belongs to a cluster.

The table 5.1 shows in the first column the set of all features ordered by linkage function and in the second and third columns there is the name of the features selected in correspondence to the first column, respectively for the leaders and for those selected manually

All Features Order By Clustering	leadersSelection	manualSelection
ev_ws_roty0	ev_ws_roty0	
ev_ws_rotx2		
ev_ws_roty2		
ev_ws_rotz2		
ev_ws_rotx3		
ev_ws_rotz3		
ev_speed_y		ev_speed_y

All Features Order By Clustering	**leadersSelection**	**manualSelection**
ev_ws_vhlDelta2		
ev_ws_vhl Delta3	ev_ws_vhl_Delta3	
ev_steeringTorq		ev_steeringTorq
ev_ws_vhlDelta0		
ev_ws_vhlDelta1		
ev_accelPedal		ev_accelPedal
ev_ws_vhlSx0		
ev_accel_x	ev_accel_x	ev_accel_x
ev_ws_vhlSx1	ev_ws_vhlSx1	
ev_gearEngaged	ev_gearEngaged	
ev_speed_x		ev_speed_x
ev_engineSpeed		
AU20_r	AU20_r	AU20_r
AU10_r		AU10_r
AU12_r		AU12_r
AU06_r		AU06_r
AU07_r		AU07_r
gaze_angle_y		gaze_angle_y
pose_RxRy		pose_RxRy
AU23_r	AU23_r	AU23_r
AU15_r	AU15_r	AU15_r
AU17_r	AU17_r	AU17_r
AU45_r	AU45_r	AU45_r
AU14_r	AU14_r	AU14_r
AU04_r		AU04_r
AU09_r		AU09_r
ev_accel_z	ev_accel_z	ev_accel_z
ev_ws_vhlSx2	ev_ws_vhlSx2	
pose_Tx	pose_Tx	
pose_Tz	pose_Tz	
ev_speed_roll		ev_speed_roll
ev_accel_heading		
ev_accel_roll	ev_accel_roll	ev_accel_roll
ev_steeringWheelSpeed	ev_steeringWheelSpeed	ev_steeringWheelSpeed
AU01_r	AU01_r	AU01_r
AU02_r	AU02_r	AU02_r
AU25_r	AU25_r	AU25_r
AU26_r	AU26_r	AU26_r
ev_ri_laneGap	ev_ri_laneGap	ev_ri_laneGap
pose_Ty	pose_Ty	
AU05_r	AU05_r	AU05_r
ev_accel_pitch	ev_accel_pitch	ev_accel_pitch
ev_brakePedal	ev_brakePedal	ev_brakePedal

All Features Order By Clustering	**leadersSelection**	**manualSelection**
ev_ws_rotx1	ev_ws_rotx1	
ev_speed_heading		ev_speed_heading
ev_accel_y		ev_accel_y
ev_ws_rotz0		
ev_ws_rotz1		
ev_wheelAngle		ev_wheelAngle
ev_steeringWheelAngle		ev_steeringWheelAngle
ev_ws_roty1		
ev_ws_roty3		
gaze_angle_x	gaze_angle_x	gaze_angle_x
ev_speed_pitch	ev_speed_pitch	ev_speed_pitch
ev_ri_roadAngle	ev_ri_roadAngle	ev_ri_roadAngle
ev_speed_z	ev_speed_z	ev_speed_z
ev_ws_rotx0	ev_ws_rotx0	
ev_ws_vhlSx3	ev_ws_vhlSx3	

Table 5.1: Features manual selected vs leaders selected

Furthermore, there was a correlation between the manually selected features and those returned by the leaders' function provided by the dendrogram.

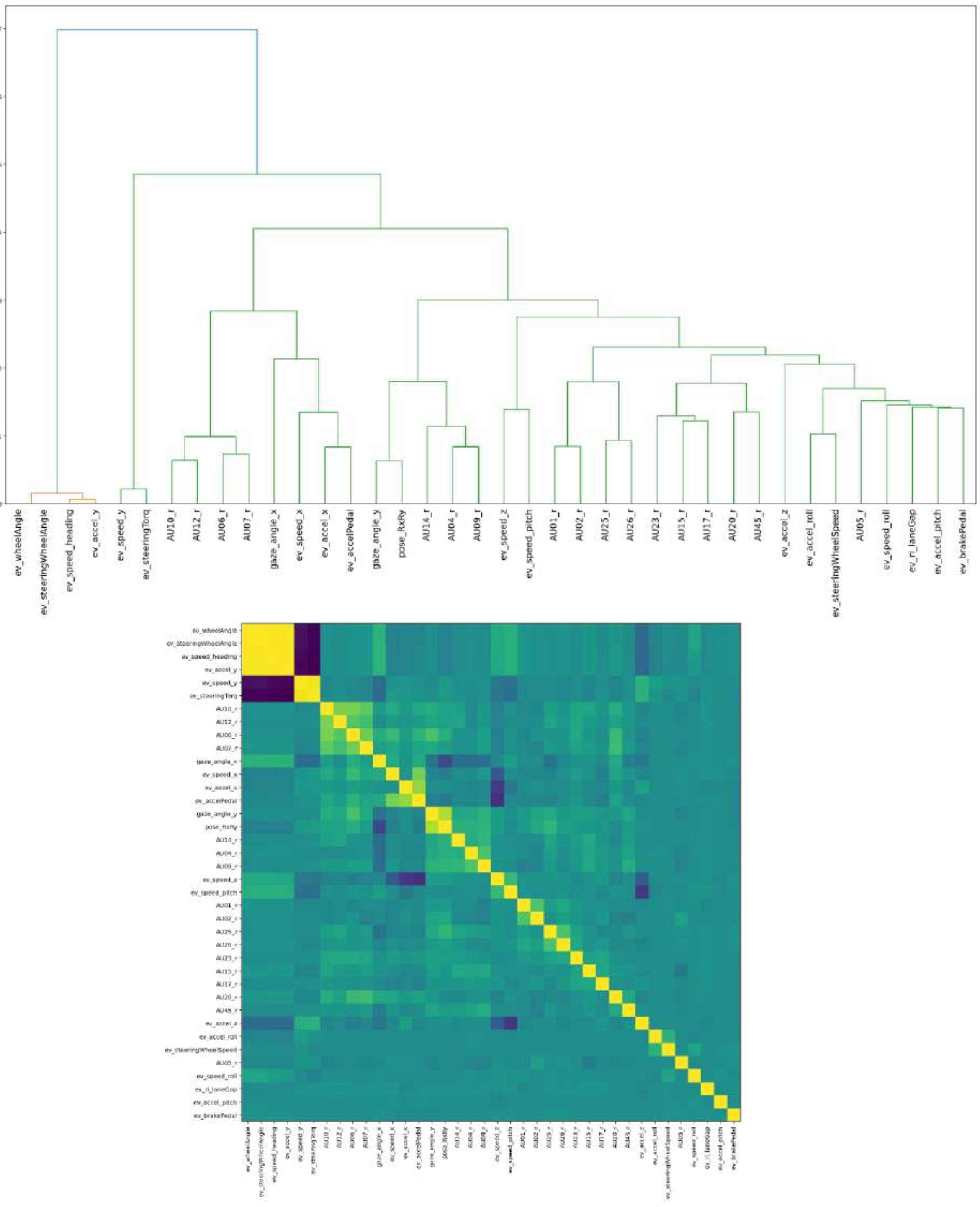


Figure 5.2: Hierarchical clustering and heatmap using Spearman rank-order correlation on manual selected features.

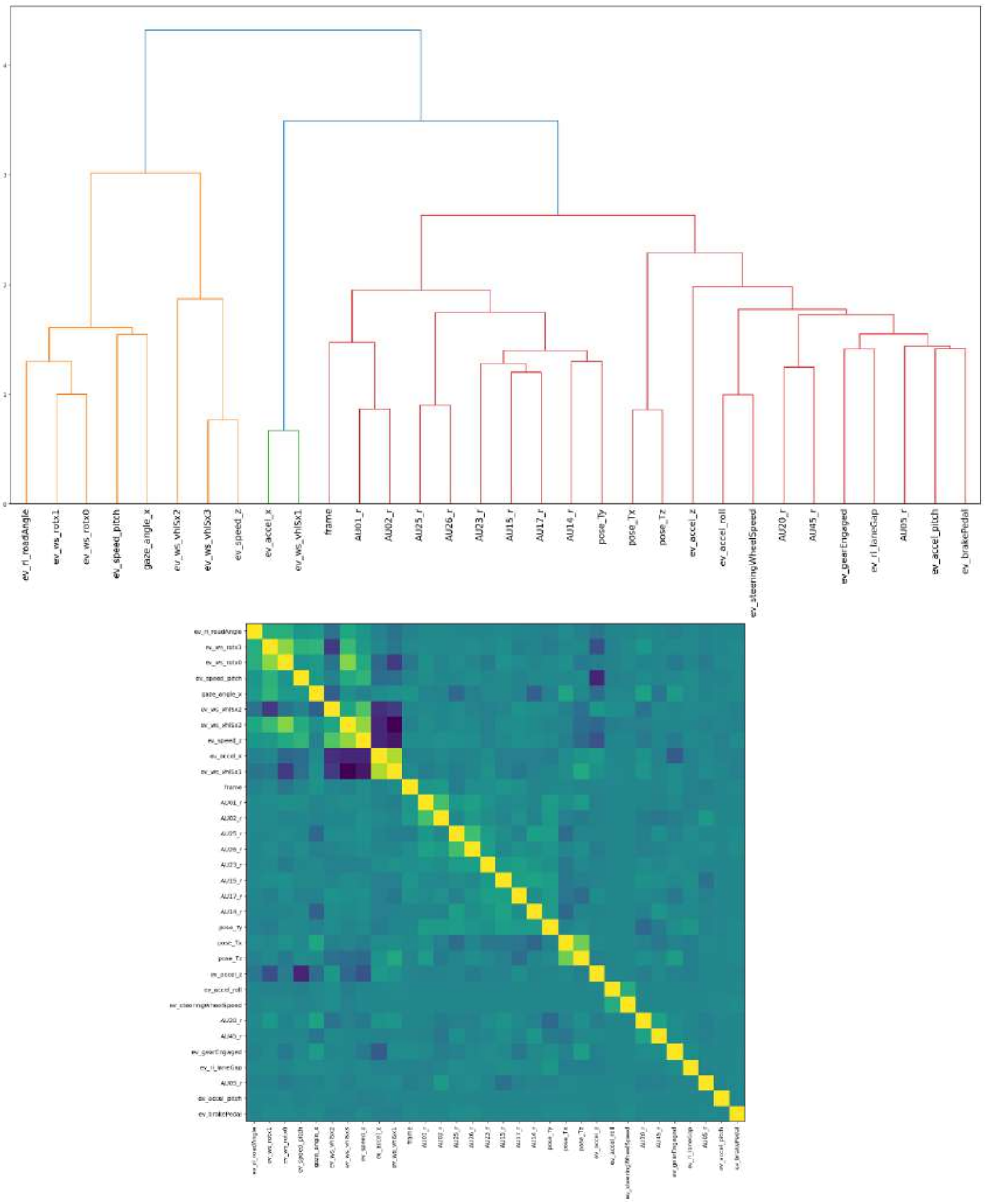


Figure 5.3: Hierarchical clustering and heatmap using Spearman rank-order correlation on leaders selected features.

In figure 5.2 we can see the dendrogram and the heatmap of the hand-selected features, and in figure 5.3 we can see the dendrogram and the heatmap of the leaders selected features.

We can see how the selection based on correlation creates much less cohesive clusters, in fact the maximum correlation between two different features is not less than 1 (cut value used in the previous phase). Despite this, however, we note that many features that were important in the state of the art are not considered and in their place we find lower level features.

A preventive analysis on the selected features based on the correlations and considering the state of the art seen (Chapter 3.1.3), it was decided to select 39 different features, shown in the table 5.2

Selected Features
AU01_r
AU02_r
AU04_r
AU05_r
AU06_r
AU07_r
AU09_r
AU10_r
AU12_r
AU14_r
AU15_r
AU17_r
AU20_r
AU23_r
AU25_r
AU26_r
AU45_r
accelPedal
accel_pitch
accel_roll
accel_x
accel_y
accel_z
brakePedal
ri_laneGap
ri_roadAngle
speed_heading
speed_pitch
speed_roll
speed_x
speed_y
speed_z
steeringTorq
steeringWheelAngle
steeringWheelSpeed
wheelAngle

Selected Features
gaze_angle_x
gaze_angle_y
pose_RxRy

Table 5.2: Features Selected

5.2 Feature extraction

As already reported in Section 2.1, a supervised learning problem consists of input patterns (X) and output patterns (y), such that an algorithm can learn how to predict the output patterns from the input patterns.

The dataset we used was already predisposed to a supervised learning problem, but each moment represented a series of point values.

In order to obtain more information from every moment of time, it was decided to process the dataset through a window capable of extracting and maintaining information of multiple point values.

In this way we are able to extract new features starting from the original ones. Then we make a features selection to keep only the most significant features.

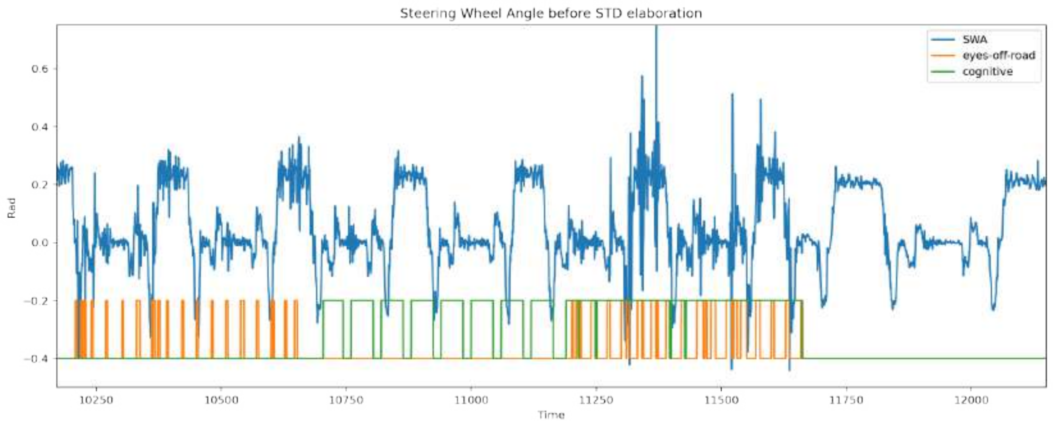
Sliding window is the way to restructure a time series dataset in such a way that the machine learning models have a wider visibility considering also previous data rather than considering a single datapoint.

One of the more popular sliding window statistics is the moving average. It is commonly used with time series to smooth random short-term variations and to highlight other components (trend, season, or cycle) present in your data.

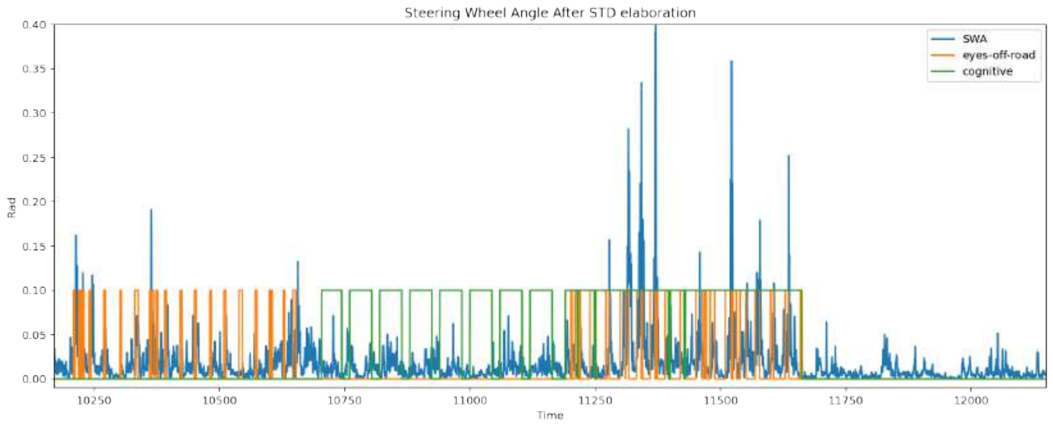
The moving average is also known as the rolling mean and is calculated by averaging data of the time series within k periods of time. The rolling method by pandas provides rolling windows over the data, allowing us to easily obtain the simple moving average.

In our case, for each of the continuous input variables above we consider a window of 1.45 seconds (corresponding to 35 frames) sliced of 25 milliseconds (corresponding to 5 frames). This produces a two-second window with an overlap of 25 milliseconds.

Furthermore, Pandas allows us to calculate not only the average but also other statistical measures. In fact, for each of the continuous input variables above, in a sliding window, six input channels have been generated, namely the last value, average, minimum, maximum, standard deviation, and trend.



(a) Data are not very significant in the different condition of distraction.



(b) After STD processing, the same feature has interesting correlations with eyes-off-road distraction, and there is also a difference between only cognitive distraction and baseline.

Figure 5.4: Two Steering Wheel Angle plot of the same driver before and after window processing.

For the average, minimum, maximum, standard deviation, Pandas was used, which with the windowing function also allows you to calculate these parameters within the window.

An example of the additional information that we can retrieve with this type of processing can be seen in figure 5.4. Figure 5.4a shows the steering wheel angle before performing the windowing and as we can see, there are no visible clues that allow us to determine differences between distraction and normal driving. In fact, they follow a constant trajectory that represents more than anything else the path.

For example, with the calculation of the STD, we immediately notice, in b, the features acquires a greater significance and allows us to certainly distinguish some moments of eyes-off-road. On the other hand, we notice small differences for the phase of only cognitive distraction compared to the normal driving phase.

For the last value instead, we proceeded by selecting the last value of each variable in the window. In this way, a punctual value is obtained (and not a derived value) which identifies the only real value contained in that window.

For the trend there is no ad-hoc function in pandas so, for each window, we proceeded to calculate the average for each window and to calculate the differential between the previous and the current average value.

In this way, for each variable, we add information relating to the previous windows and we are able to realize if the average value in the windows rises, falls, or remains stable with respect to the previous average value.

A further feature was created starting from the laneGap.
As reported in Section 4 the dataset description, laneGap is a discriminatory feature as a distance jump represents a change of sensor lane. But this change of lane can occur both in case of distraction and when changing lanes or roads.

Furthermore, in the window, through the calculation of the average, for example, we lose the information relating to the lane change.

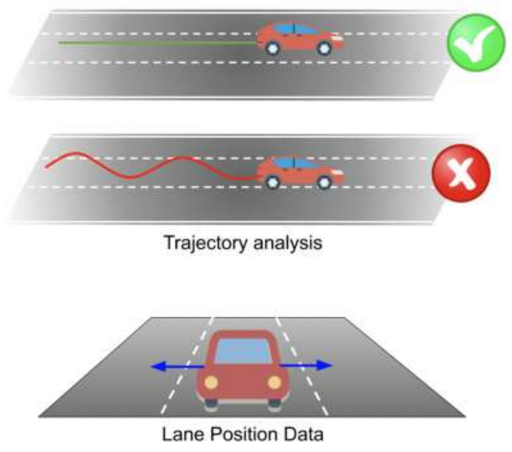
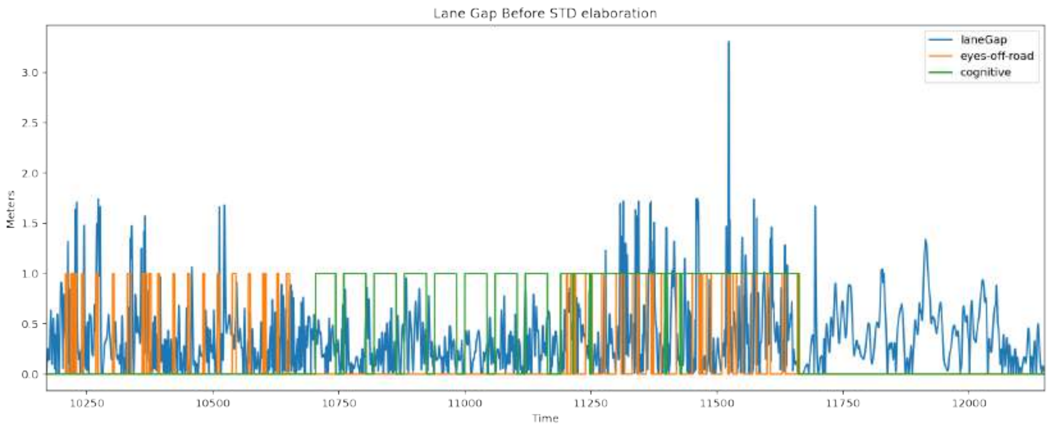


Figure 5.5: The idea of lane Position Data to detect the right and wrong trajectory.

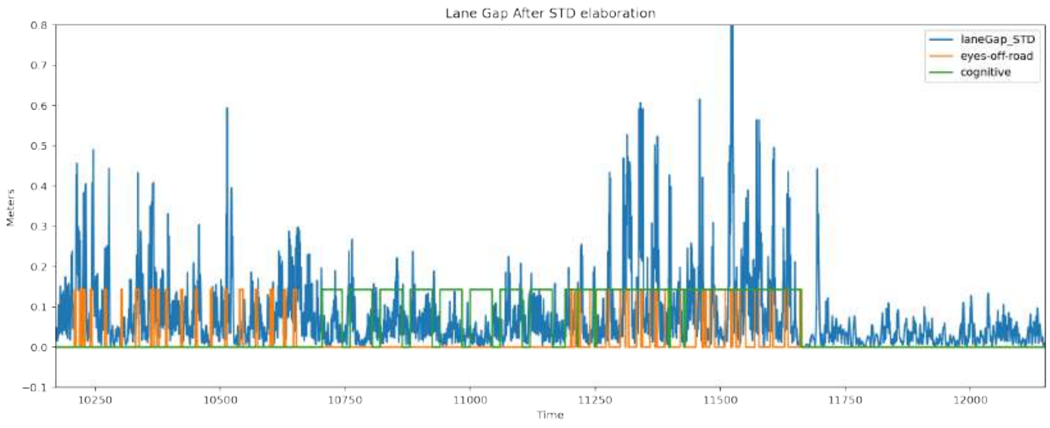
The idea is to recognize one from the other simply by evaluating, after a change of line, if the car quickly returns to the previous reference, or if this lane change is an isolated event (which therefore indicates a safe change of lane) (figure 5.5).

Initially, therefore, it was decided to work inside the window, counting lane changes inside it, but being the window of 1.45 seconds most of the counts that were highlighted were all single.

For this reason, it was decided to calculate the differential between the average values determined in the glazing. This differential will be low at the change points (gradual



(a)



(b)

Figure 5.6: Two laneGap plot of the same driver (a) Data are not very significant in the different condition of distraction (b) After STD processing, the same feature has interesting correlations with eyes-off-road distraction, and there is also a difference between only cognitive distraction and baseline.

descent or rise) but instead will have a higher value (both positive and negative) when the change occurs abruptly. So, for each step, it computed the corresponding change of a function.

The graph figure 5.6b identifies what has just been said (blue line), in fact, thanks to this processing it is possible to more easily identify the lane change points in which the driver was visually and cognitive distracted respect to 5.6a, unlike the simulation circuit curves whose values are attenuated.

Another important processing that we have dealt with concerns window labeling.
The label, for either distraction, was 0 or 1. When we calculate a window values, also the value of the distraction would be changed.

The goal is to leave an integer value for the eyes-off-road (0 or 1), determined by calculating the average within each window of all values referring to the eyes-off-road and to bring the distraction value to 0 if the average distraction value in the window does not exceed 0 and instead bring it to 1 if the value exceeds 0.

In this way we can teach the model to notice in advance and immediately from the first signs of distraction.

Since the cognitive distraction is not instantaneous, as is the eyes-off-road condition, the labeling was not carried out by setting 1 as soon as a value greater than 0 appeared in the window.

In this case, it was decided to set the distraction to 1 only when the cognitive average value was actually present in at least half of the data points contained in a window. To do this, it was decided to set the distraction to 1 only when the average value in the window exceeded 0.5, and vice versa to set it to 0.

Chapter 6

Models Training and Evaluation

6.1 Model Training

After carrying out features extraction and selection, we proceeded to create models to carry out the prediction. As seen in the state of the art, there are several machine learning approaches and the newest refer to neural networks.

In this study, many machine learning techniques have been tested, but the best ones obtained and reported are an MLP and a 1D-CNN, that outperforms all simple machine learning models and other neural networks tried, such as K-NN, random forest, SVM, LSTM.

An important thing that it is essential when we carry out the learning, is to divide the data set into a training, test, and validation set. In this way, we can monitor the performance of the network on examples never seen before at each epoch (Test Set) and after test if the accuracy and generalization are also good for validation data (Validation Set).

6.1.1 Models Summary

The generic network architecture is shared for all types of neural networks and was implemented through the Keras library in Python. The functions of Keras, therefore, allow you to manually create a single model with two different outputs.

Overall, we have the input layer at the beginning and instead at the end two different Dense layers with a single neuron, where each one makes its prediction, one for visual distraction and the other for cognitive distraction, both with sigmoid activation.

Since there are two output neurons, two different measures of loss will be required during compilation. Furthermore, the two tasks are the same; we want to predict if, or not, a driver put his eyes-off-road or if is cognitively distracted. For this reason, as loss measures, it was decided to implement binary cross-entropy. Keras also calculates the total loss weighed.

In fact, because gradient descent requires you to minimize a scalar, you must combine these losses into a single value in order to train the model. The resulting loss values are summed into a global loss, which is minimized during training.

Note that very imbalanced loss contributions will cause the model representations to be optimized preferentially for the task with the largest individual loss, at the expense of the other tasks. To remedy this, you can assign different levels of importance to the loss values in their contribution to the final loss. In such a situation, to balance the contribution of the different losses.

The simplest way to combine different losses is to sum them all and multiplying the value by a value between 0 and 1 which indicates its importance. In our specific case, for example, we have that the visual distraction task turns out to be a very trivial task as the head pose feature is also included among the features.

This means that the classifier will easily be able to determine a moment of visual distraction at the expense of cognitive distraction which, on the other hand, does not have such specific features.

In this way, the loss of visual distraction will be low and will always go down, while the loss of cognitive distraction will be more fluctuating. For this reason, it was decided to simply assign a much greater weight to the loss of cognitive distraction (0.9) and instead of a small weight to visual distraction (0.1).

Further specification concerns the optimizer used. It was decided to use the one named Stochastic Gradient Descent optimizer with an adaptive learning rate. Specifically, the adaptive learning rate technique used, named Step decay, allows you to schedule drops of the learning rate by a factor every few epochs. The training set was partitioned in a sequential sequence of batch size.

This promise to stabilize gradient computation. In fact, network parameters are updated using gradient descent and regularization with the following formula:

$$\theta^{\text{new}} \leftarrow \theta - \eta_t \left[\tfrac{1}{B_k} \textstyle\sum_{i \in I} \left[\tfrac{\partial}{\partial \boldsymbol{\theta}} L\left(f\left(\mathbf{x}_i; \boldsymbol{\theta}\right), \mathbf{y}_i\right)\right] + \tfrac{B_k}{N} \lambda \tfrac{\partial}{\partial \boldsymbol{\theta}} R(\boldsymbol{\theta})\right]$$

That computing the derivate of the loss for each example x and target values y in the specific batch and sum out together.
This process is controlled by the learning rate $\theta - \eta_t$ which indicates the amount of displacement required to achieve a good change in weights.

In fact, the summed and normalized value of a single batch will be multiplied by the learning rate (which will resize it) before adding it to the values of the weights prior to the iteration and then making the change.

Another trick that has been included in the stochastic gradient descent to avoid blindly trusting the value we compute on the batch is the momentum α, which is a memory mechanism (of inertia) that remembers how we changed the parameters in the previous step and with a factor between 0 and 1 (aka inertia) we keep moving in that direction. The following formula explains what we just said:

$$\Delta\boldsymbol{\theta} = -\eta\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta}) + \alpha\Delta\boldsymbol{\theta}^{\text{old}} \text{ , where } \alpha \in [0,1]$$

Furthermore, the batch size influences the stability and speed of learning.

During the training, when all the examples end after several batches, then an epoch is said to be over. At this point, it is not certain that after having dealt with all the examples there is nothing more to change so you can proceed for several eras and try to better refine the value of the weights.

However, there must be a limit to learning, in fact, trusting only the data and the loss value, we must determine when to stop and therefore how to reach convergence.

In fact, the loss value will tend to decrease as the epochs advance, but we must be careful not to fall into overfitting or to simply store the training data as otherwise, the network will not be able to generalize on new examples. To do this, a model callback

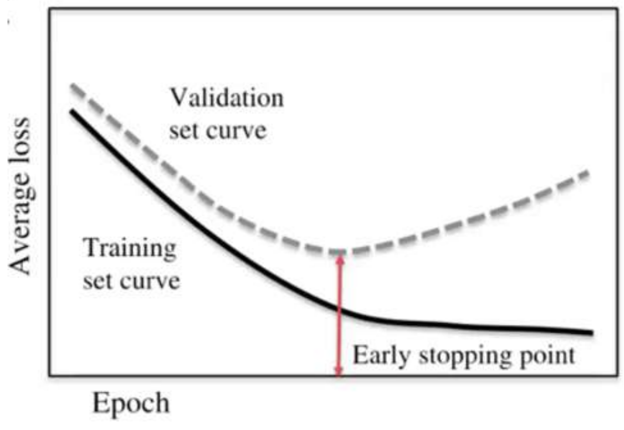


Figure 6.1: Early stopping point in a model being learning

has been added and customized in our models' training phase. Early Stopping is one of the most widely used regularization techniques to combat the overfitting issue used as callback during the training.

It monitors the performance of the model for every epoch on a held-out validation set during the training and terminates the training conditional on the validation performance. The way it does is to stop training as soon as the validation error reaches a minimum.

As the epochs go by, the algorithm learns and its error on the training set naturally goes down, and so does its error on the validation set. However, after a while, the validation error stops decreasing and actually starts to go back up.

This indicates that the model has started to overfit the training data. With Early Stopping, you just stop training as soon as the validation error reaches the minimum. Figure 6.1 shows what has just been described.

But in the case of Stochastic and Mini-batch Gradient Descent, as in our case, the curves are not so smooth, and it may be hard to know whether you have reached the minimum or not.

Therefore, the algorithm could stop the training too quickly and may end up with the model trapped in a local minimum. By disabling it we requested the training to be completely up to the last epoch, obtaining an overall better performance on the validation.

A good trade-off, in the cases for which the full training is too time-consuming, maybe to set the "patience" threshold to a higher value.

In our case, we set the "patience" threshold at 50 epochs with a min delta of 0.005.

This method is great for looking for a global minimum and not getting trapped in a global minimum, but it increases learning time and does not guarantee that you will get the best pattern.

This is because when the learning ends, then it means that n epochs have passed (in our case 50) and the loss has not gone down. Returning the model obtained in the last epoch would not return the model for which the loss was minimal.

To solve this problem, we use the extension of the callback in learning by also adding a Model Checkpoint. Model Checkpoint, like Early Stop, monitors a parameter (in our case the loss) and saves the model whenever it improves compared to the previous one.

In this way, the Early Stop tells us when we can stop because we have hopefully reached a global minimum, while the Model Checkpoint saves the model at the point of the global minimum.

Following the ordinary procedure for supervised learning, we use 10-fold cross-validation on 90% of the instances, while 10% of each dataset has been kept as a separate test set, on which the learned models have been tested.

We used a “leave-one-out” approach: one model has been trained on the data from 9 out of 10 subjects, and tested on the data of the left-out subject (in turn, on every subject) and results averaged.

The best two models generated and the hyperparameters fine-tuning will be shown below.

MLP

For the Multi-Layer Perceptron model, in addition to the generic model just described, a simple hidden layer (Dense) with Relu as activation function was added with a tuned number of neurons. For this model the hyperparameters tried are:

1. Learning rate - 0.1, 0.01,0.001,0.005,0.0001
2. Batch size - 1, 32, 64, 128, 256, 512, 1024, 2048
3. Number of hidden neurons - 15, 30, 50, 100, 150, 300

To find the best combination we have monitored the f1 score of each model and finally the best combination turned out is 0.01 for learning rate, batch size of 128, and 15 hidden neurons. In figure 6.2 is shown the MLP model summary

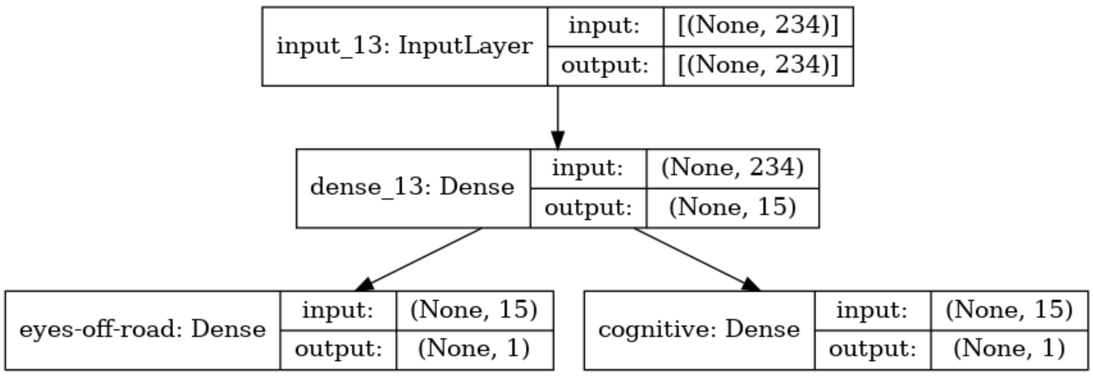


Figure 6.2: MLP model summary

1D-CNN

For the 1D Convolutional Neural Network model, in addition to the generic model, were added two 1D-CNN layers, each followed by a Max Pooling layer. the second has half as many filters as the first.

These two layers should extract details from the features and make them more meaningful for prediction. Finally, after flattening operation on the output of the 1D-CNN, a dense layer was added to make the prediction on the CNN features. For this model the hyperparameters tried are:

1. Learning rate - 0.1, 0.01,0.001,0.005,0.0001
2. Batch size - 1, 32, 64, 128, 256, 512, 1024, 2048
3. Number of filters - 5, 10, 30, 60, 90, 150
4. Kernel size - fixed to 234 to apply the convolution in only one dimension
5. Strides - 1, 3, 5, 7, 10, 15
6. Number of hidden neurons of the last dense layer - 15, 50, 150, 300, 500

To find the best combination we have monitored the f1 score of each model and finally the best combination turned out is 0.0001 for learning rate, batch size of 128, 60 filters, 1 stride and 15 for hidden neurons of the dense layer. In figure 6.3 is shown the CNN model summary.

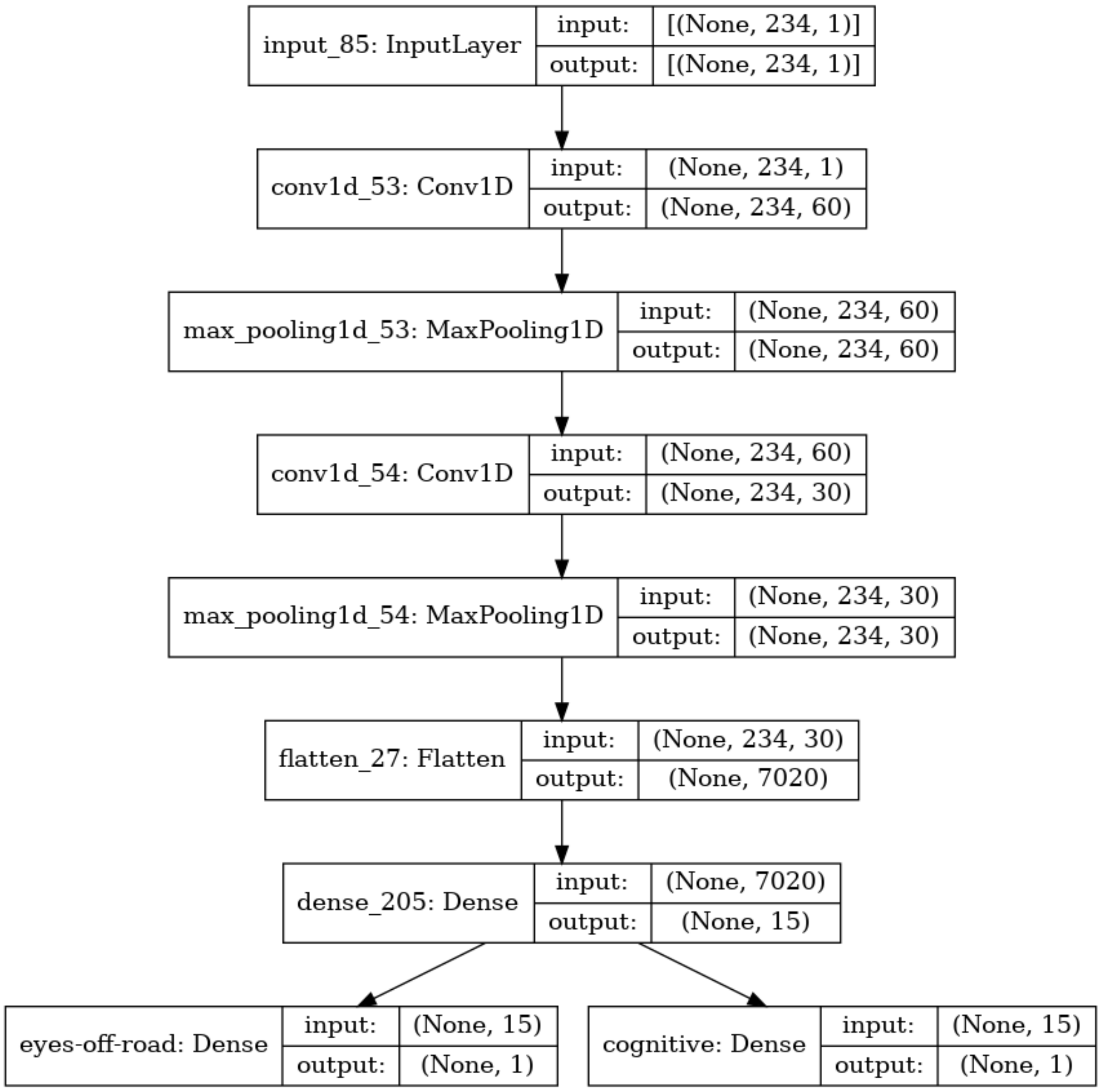


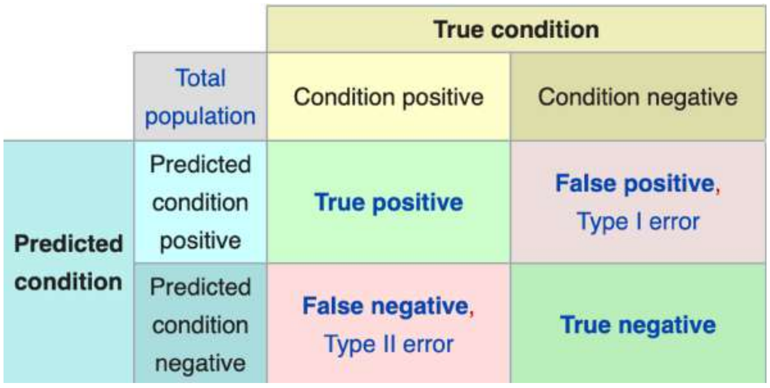
Figure 6.3: CNN model summary

6.2 Model Evaluation

To score the models' performance, some common metrics were used: accuracy, precision, recall, and F-measure (F_1), confusion matrix and ROC curve with AUC score which are defined as following

1. **Confusion Matrix**: which shows the ways in which your classification model is confused when it makes predictions. The idea is that the number of correct and incorrect predictions are summarized with count values and broken down by each class. This is a summary of prediction results on a classification problem.

It represents a specific table showing the performance of a supervised learning algorithm. Each line of this table represents instances of predicted values, while the columns represent instances of actual values. It turns out to be a special contingency table with two simple "real" and "predicted" dimensions and the same class instances in both dimensions. Specifically for binary problems, it is composed of 2 rows and two columns that report the number of false positives (FP), false negatives (FN), true positives (TP), true negatives (TN). As can be seen from the figure 6.4, there are two types of error: the first indicates false positive, ie when the condition is actually negative and the prediction is positive; the second indicates the false negative, which, unlike the previous one, indicates a situation in which in reality the condition is positive but the system predicts negatively. From these values, the main metrics such as precision, recall and accuracy are calculated in both dimensions.



		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive	False positive, Type I error
	Predicted condition negative	False negative, Type II error	True negative

Figure 6.4: Confusion Matrix Metrics

2. **Accurancy**: which measures the overall performance of the model by quantifying the proportion of correct predictions over all predictions:

$$\text{Accuracy } = \frac{N_c}{N}$$

where N_c is total number of correct observations, and N is total number of observations.

3. **Precision**: which measures the number of true positives over total predicted positives. The precision of class c can be obtained by:

$$\text{Precision }_c = \frac{TP_c}{TP_c + FP_c}$$

where TP_c and FP_c are the number of true-positive and false-positive of class c (i.e. baseline and eyes-off-road events), respectively.

4. **Recall**: which measures the number of corrected classified observations over the total observations in the class c which measures the number of corrected classified observations over the total observations in the class c:

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

where FN_c is the number of false-negative of class c.

5. **F1-Score**:which applies weighted harmonic average to the precision and recall:

$$F_{1_c} = 2 * \frac{\text{Precision }_c * \text{ Recall }_c}{\text{Precision }_c + \text{Recall}_c}$$

The advantage of F_1 is its suitability for imbalanced data where the proportion of one of the classes is lower compared to others. This measure combines precision and recall metrics and weighting the classes based on the proportion.

6. **Area under the curve (AUC)**: is one of the well-known measures that measure the area under the ROC curve to quantify the performance of a binary classifier where ROC can be obtained by plotting the true positive rate (recall) vs false positive rate. the figure shows an example of the ROC curve and AUC. A very performing classifier will approach the upper left corner of the chart. Random guessing, as if it were the toss of a coin, is identified by the diagonal identified in the figure by the dashed black line.

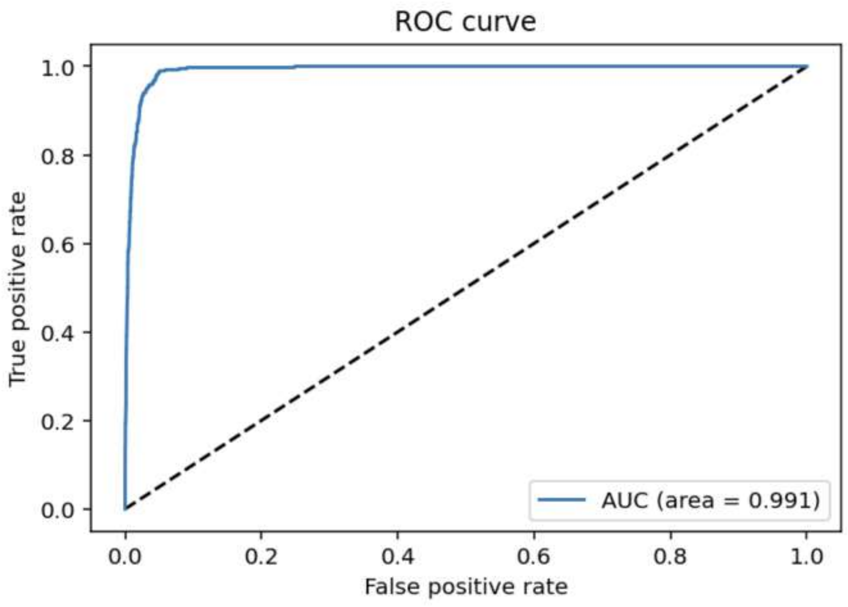


Figure 6.5: ROC curve example

6.2.1 Result

Below you can see the model's result with the metrics just explained. For each classifier, two figures are reported (one for each class to be predicted). In each image, there is the confusion matrix, ROC curve, and a classification report that reports precision, recall and f1-score for each class (0-baseline, 1-distraction) and accuracy.

F1-SCORE											
MODELS	1	2	3	4	5	6	7	8	9	10	**AVERAGE**
MLP	80%	83%	87%	88%	**92%**	82%	81%	84%	86%	88%	**85%**
1DCNN	85%	80%	87%	88%	**91%**	81%	79%	84%	85%	83%	84%

Table 6.1: F1-score values of the visual distraction class calculated through cross-validation with 10 different participants (leave-one-out).

ACCURACY											
MODELS	1	2	3	4	5	6	7	8	9	10	**AVERAGE**
MLP	52%	**60%**	57%	57%	57%	58%	55%	54%	57%	59%	**57%**
1DCNN	45%	**61%**	60%	57%	59%	57%	51%	51%	53%	50%	54%

Table 6.2: Accuracy values of the cognitive distraction class calculated through cross-validation with 10 different participants (leave-one-out).

Immediately below are two graphs showing the prediction compared to the actual labelling (y-axis), in order of time (x-axis) and another graph containing the Cumulative Distributional Function (CDF) of actual and predicted label which indicates what is the cumulative frequency of each value between 0 and 1.

As already mentioned, the validation was carried out by performing a cross-validation by alternating 10 different participants and the results reported represent the average of every single model created. In fact, when we talk about the maximum score, we mean the best model obtained during the cross-validation.

The models return a value between 0 and 1. To calculate the above metrics, you need binary output.We have obtained this values by cutting the continuous value, between 0 and 1, returned by the model with a threshold. If the value exceeds the threshold then eyes-off-road will be 1, otherwise 0.

After checks, the thresholds were determined. For the eyes-off-road the determined threshold is 0.8 while for the cognitive distraction it is 0.5.

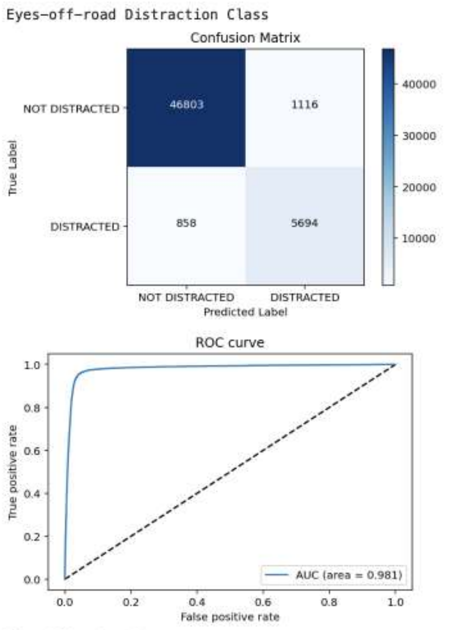
In table 6.1 we can see the results obtained by the two models for the eyes-off-road condition, calculated through cross-validation. In table 6.2, on the other hand, we see the values obtained for cognitive distraction.

In the figure 6.6 we can see the results obtained by the MLP model cross-validation results in classifying the eyes-off-road, while in the figure 6.7 we can see the results for cognitive distraction.

In the figure 6.8 we can see the results obtained by the 1D-CNN model in classifying the eyes-off-road, while in the figure 6.9 we can see the results for cognitive distraction.

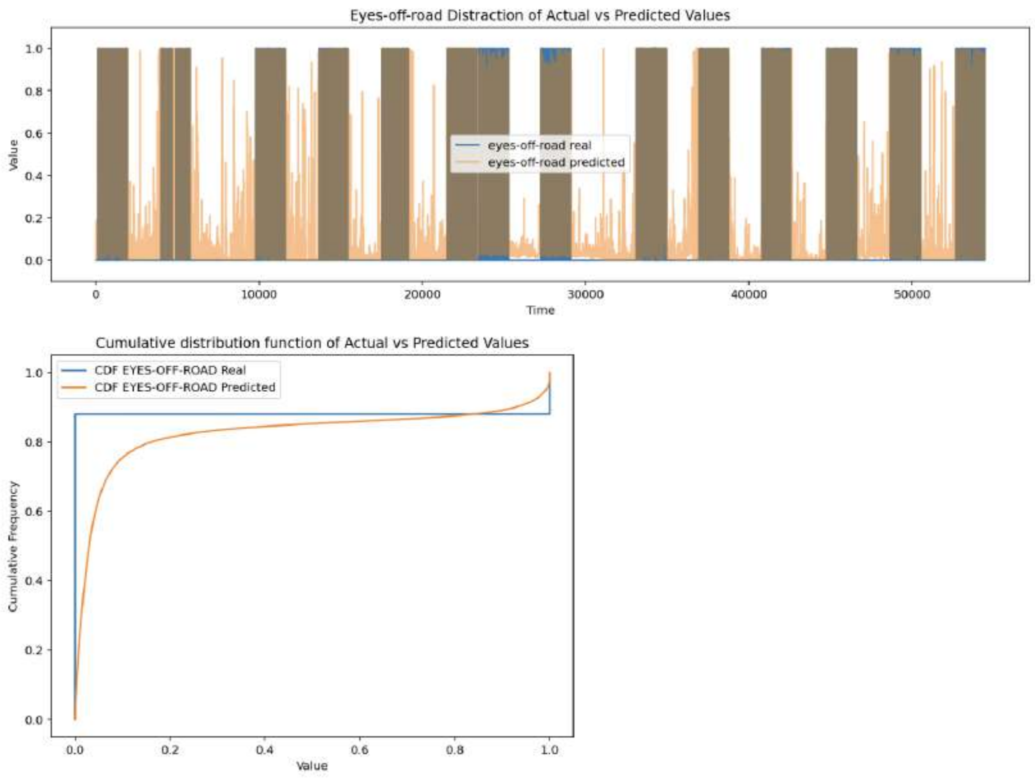
As we can see, the average f1-score obtained for eyes-off-road condition is 85% for MLP model and 84% for 1D-CNN Unfortunately, the results of the cognitive distraction condition were not really satisfactory, since the best-obtained performance was around 61% of accuracy and 57% of average accuracy (this is a very poor result).

However, these two models, as already mentioned, outperform traditional machine learning methods tried.



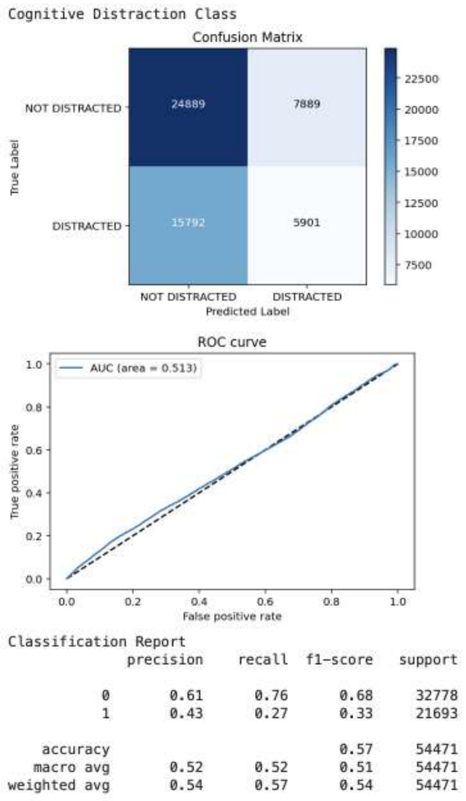
```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     47919
           1       0.84      0.87      0.85      6552

    accuracy                           0.96     54471
   macro avg       0.91      0.92      0.92     54471
weighted avg       0.96      0.96      0.96     54471
```

(a)



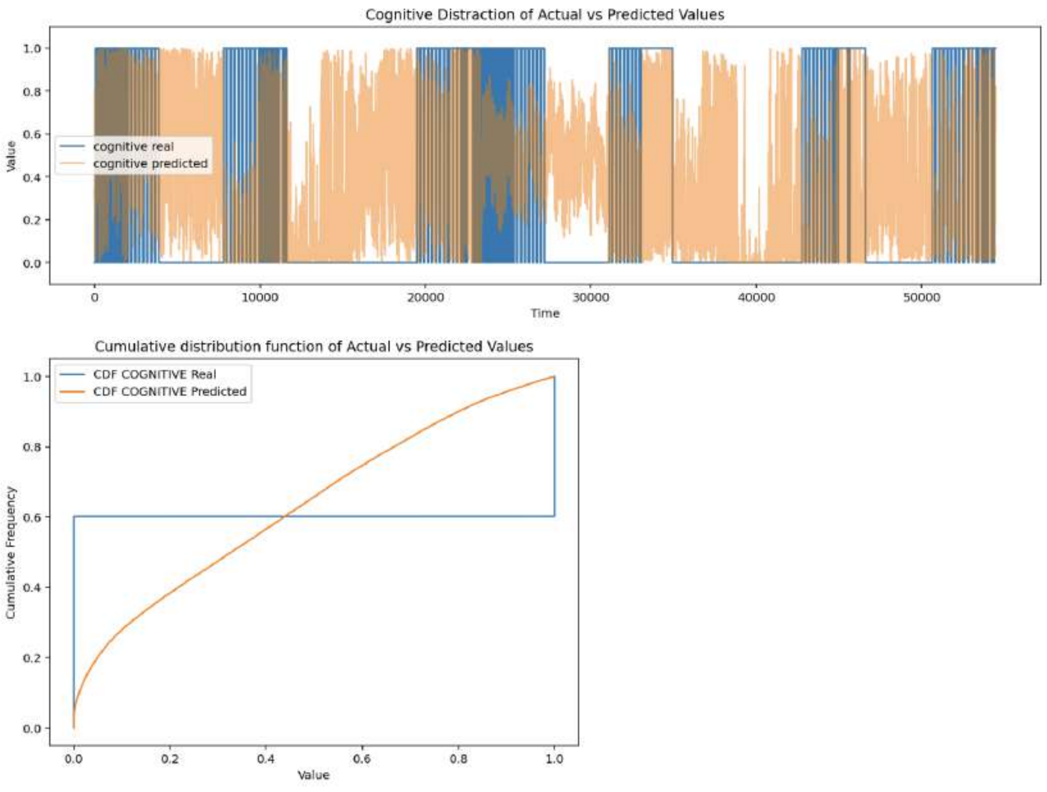
(b)

Figure 6.6: MLP Eyes-off-road result (a) Confusion Matrix, ROC curve ad classification Report (b) Actual vs Predicted Plot and Cumulative Distribution Function



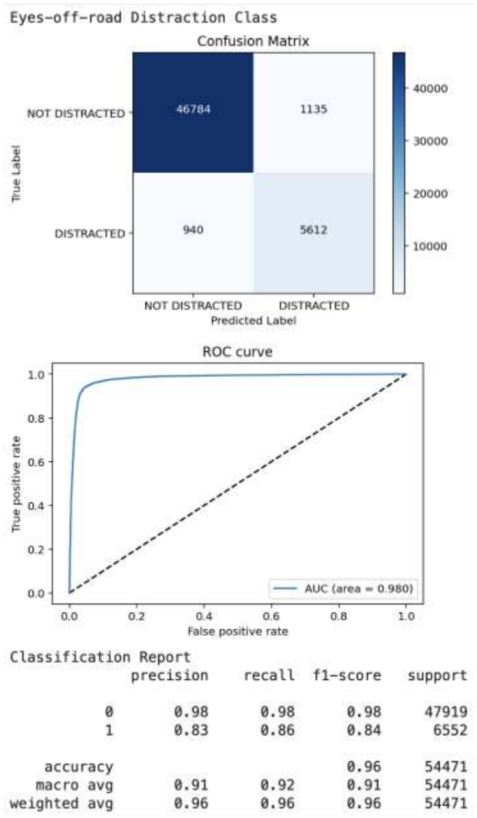
```
Classification Report
              precision    recall  f1-score   support

           0       0.61      0.76      0.68     32778
           1       0.43      0.27      0.33     21693

    accuracy                           0.57     54471
   macro avg       0.52      0.52      0.51     54471
weighted avg       0.54      0.57      0.54     54471
```

(a)



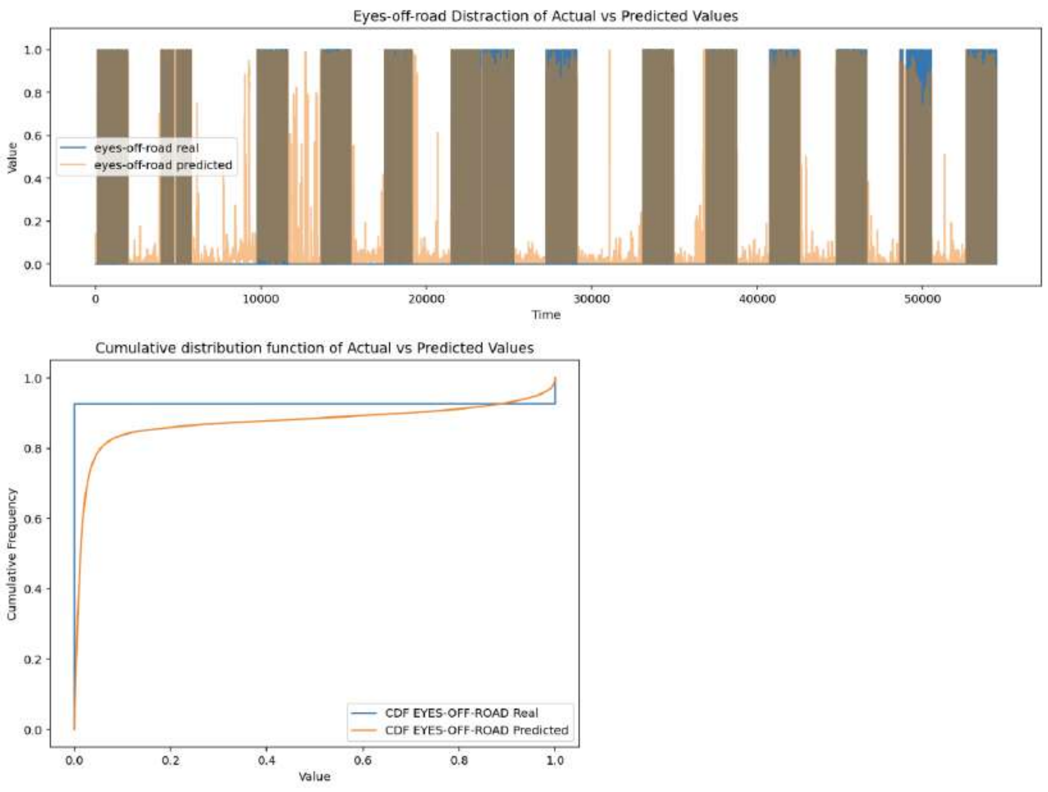
(b)

Figure 6.7: MLP Cognitive result (a) Confusion Matrix, ROC curve ad classification Report (b) Actual vs Predicted Plot and Cumulative Distribution Function



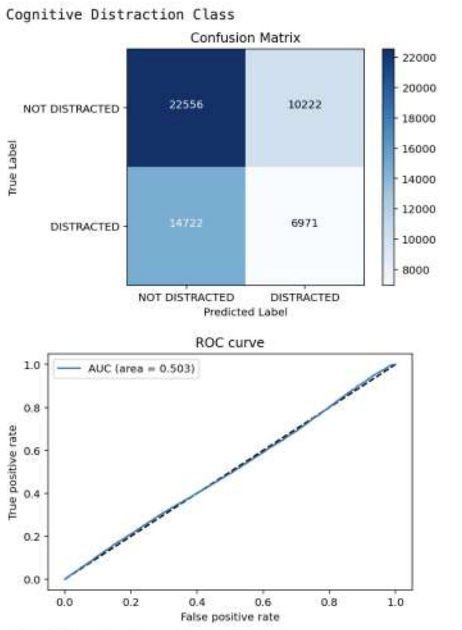
```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     47919
           1       0.83      0.86      0.84      6552

    accuracy                           0.96     54471
   macro avg       0.91      0.92      0.91     54471
weighted avg       0.96      0.96      0.96     54471
```

(a)



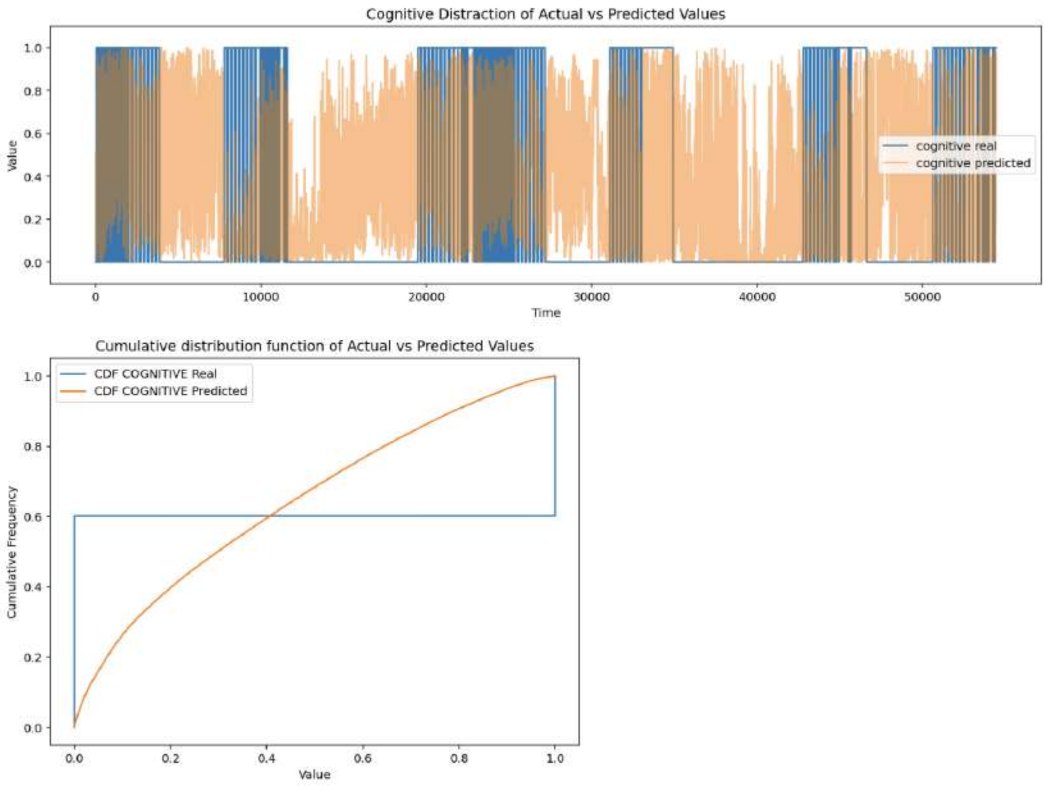
(b)

Figure 6.8: CNN Eyes-off-road result (a) Confusion Matrix, ROC curve ad classification Report (b) Actual vs Predicted Plot and Cumulative Distribution Function



```
Classification Report
              precision    recall  f1-score   support

           0       0.61      0.69      0.64     32778
           1       0.41      0.32      0.36     21693

    accuracy                           0.54     54471
   macro avg       0.51      0.50      0.50     54471
weighted avg       0.53      0.54      0.53     54471
```

(a)



(b)

Figure 6.9: CNN Cognitive result (a) Confusion Matrix, ROC curve ad classification Report (b) Actual vs Predicted Plot and Cumulative Distribution Function

During the development we checked frame by frame (from the videos recorded by the simulator) the consistency of the predictions and it was deduced that many times the classifier behaves even better than what f1-score tells us on eyes-off-road identification (Figure 6.10).

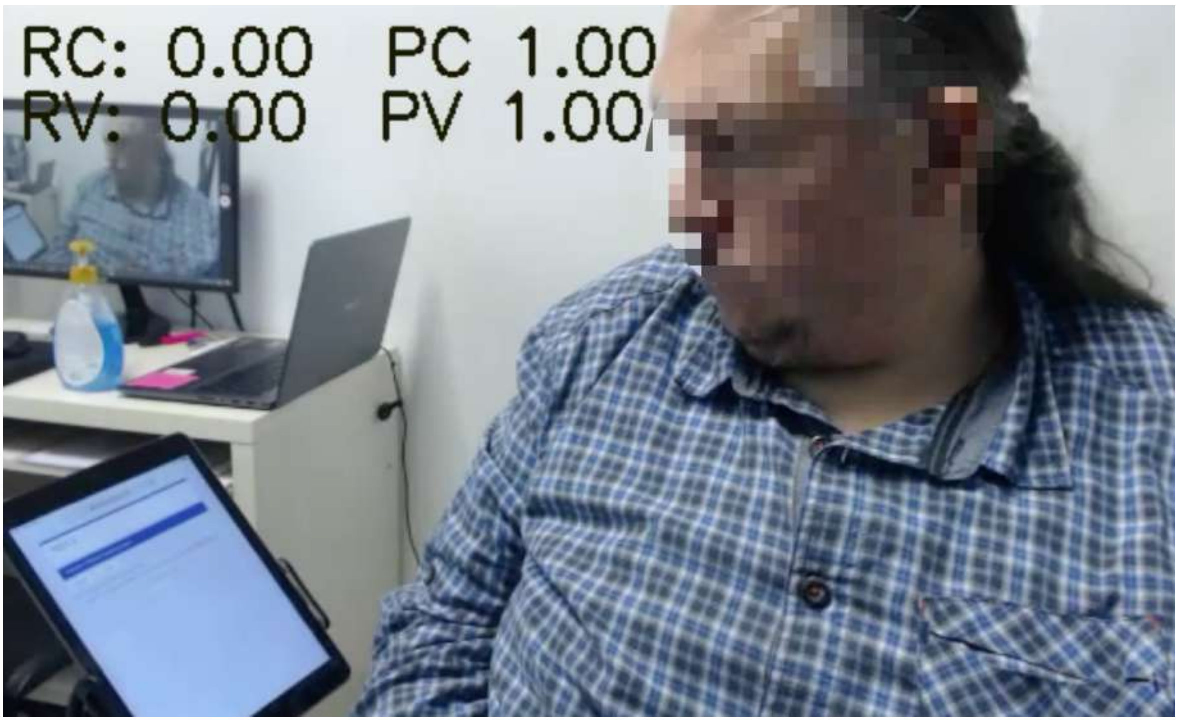


Figure 6.10: Example of a frame in witch there is eyes-off-road distraction, and our classifier detect it, but it isn't labelled as distracted. RC: Real Cognitive, RV: Real Eyes-off-road, PC: Predicted Cognitive, PV: Predicted Eyes-off-road

In fact, in many cases the eyes-off-road value returned by the classifier is 1 due to the gaze or position of the head turned too far to the right or left.

However, these correctly classified moments are not labelled as an eyes-off-road condition. This happens above all because during the labelling of the data it was considered to label as eyes-off-road only the moments included during the "visual distraction" session (ie in 2 out of 4 driving sessions).

This represents a bias, as many participants during the test looked away and/or turned their heads for reasons unrelated to the tests. These moments, however, are identified by our classifier as can be seen in figure 6.11.

In the graph, we see actual values (blue) vs predicted ones (yellow). We can see that at about 1600 there is a peak and seeing the video, the driver actually turned his head.

Another problem encountered in the dataset can be seen from the graph in the figure 6.11, at about 1200 we have a peak in yellow (therefore predicted) but the peak in blue

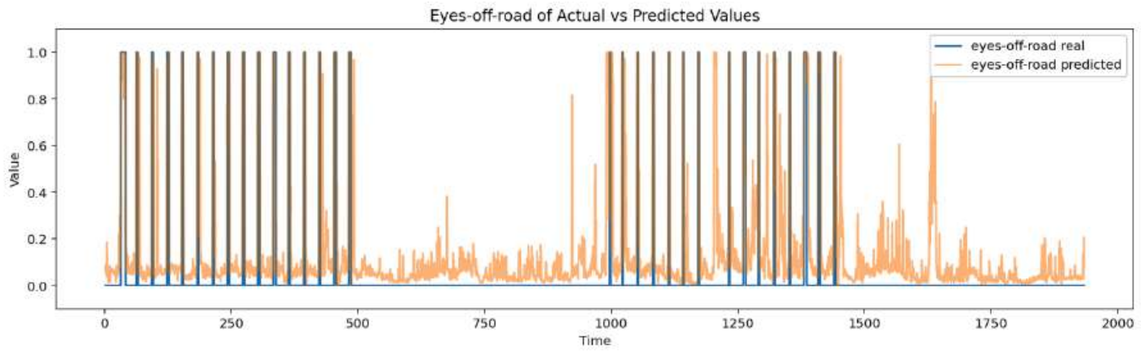


Figure 6.11: Example of eyes-off-road actual vs predicted values

(therefore actual) is missing.

In some cases, in fact, the labelling is missing even during the visual distraction session. in blue (so actual). In some cases, in fact, the labelling is missing even during the visual distraction session.

Furthermore, as can be seen from the confusion matrix of the eyes-off-road condition, there would also be cases in which the classifier does not correctly classify some instants in which the eyes-off-road condition exists.

Inspecting these moments as well, it was possible to verify that in reality in many cases these are very fast head movements, made by experienced drivers which therefore imply few variations in the data in an entire window and the labelling value remains 1 even while these small head movements.

This is also for cases in which the driver takes his eyes-off-road, and for a moment returns to look at the road and then turns his head again.

In labelling, this phenomenon is seen as two different moments of eyes-off-road, but in the data processing carried out by us, going to amplify the data through a window, these moments (in which therefore there is a few moments of non-distraction) are classified as continuous eyes-off-road but in some cases, since the output is a value between 0 and 1, the returned value does not exceed the decision threshold of 0.8. In fact, looking at the ROC curve, we realize that it is very close to the ROC heaven, the area under the curve is very high.

According to this analysis, the classifier in the eyes-off-road condition performs very well.

This is due to the fact that eyes-off-road distraction is a very simple task to retrieve with the front camera; in fact, even with a very simple MLP Neural Network model, on cross-validation (leave-one-out) the maximum f1 score obtained was 92% and an average value of 85% and it doesn't grow with more complex deep learning methods, in that it couldn't be done better with the dataset used. In fact, even with the 1D-CNN network no better results have been obtained.

On the other hand, the models reach poor performance and not precisely determine the decision boundary for cognitive distraction. In fact, we are unable to determine which factors it uses to make the prediction. In fact, the prediction returns a random value, as if it were random guessing.

The ROC curve, indeed, is very close to the black dashed line that represents the random classifier.

This is due to the scenario used in the acquisition of data from the simulator, which was too simple and trivial (without other vehicles, without any obstacles) which led the participants not to experience much cognitive distraction and therefore it was not possible to differentiate between the data acquired during the standard driving phase and during the driving phase with cognitive distraction (N-Back test).

An additional problem of the dataset used is that there is a high bias. As already mentioned, during the acquisition of data, in the Baseline driving phase, in which the only task should be driving, some users, due to boredom, spoke to other people in the room that was not well isolated.

In order to reduce the bias, we have eliminated some participants from data. Another source of bias in the dataset is due to the fact that the moments tagged with cognitive distraction are too dense and include an amount of data that exceeds half of those acquired and in addition, there is a clear shift from 0 to 1 (distracted and not), for each type of distraction induced.

Therefore, since the period is considered too long, it also includes times when the distraction is not actually high, and this leads to excessive confusion for the model.

A further factor that most likely does not allow a good classification is due to the fact that response to distraction is highly personal and subjective, so the normal behaviour of one driver can be similar to the distracted behaviour of another driver.

This cause in the dataset a very high variance can be determined by the fact that during learning, a much more precise classification is obtained for the training examples and instead reach poor results during the testing and validation phase.

In fact, for very high model complexity (a high-variance model), the training data is over-fit, which means that the model classify the training data very well, but returns inadequate performances for any previously unseen data.

To solve this problem, we would have to acquire much more data from many different drivers. This would allow the network to generalize better and not specialize too much in training examples.

We also noticed that in the double distraction session, there is a lot of variation in the data than in the other two sessions. While in the purely visual distraction session we have an instant distraction for a few seconds (time needed to perform the task on the on-board computer), and while in the cognitive-only distraction session (during the n-back test) we still have more attention on the road, in the double distraction session, the errors caused by the visual distraction and the continuous cognitive distraction (trying to remember the sequence that is dictated) causes a very high cognitive load that takes away space for safe driving.

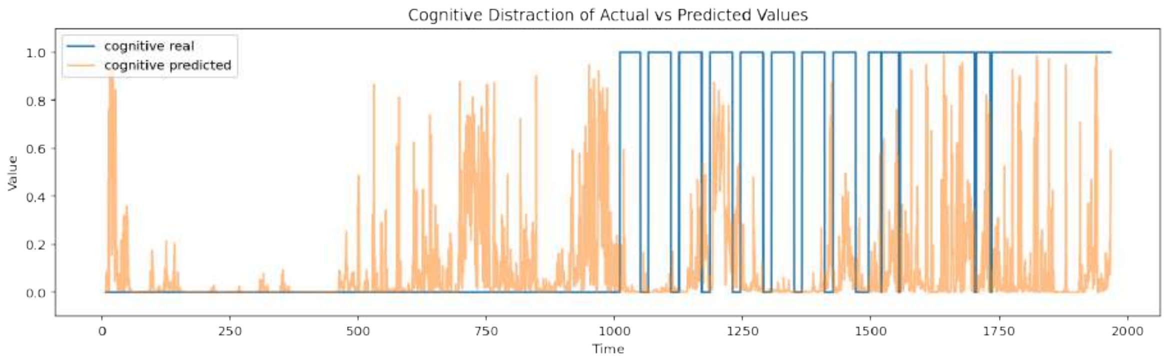


Figure 6.12: Early stopping point in a model being learning

As we can see in figure 6.12, the visual distraction prevails over the cognitive one, because it is easier to identify. In fact, we note how in this case, the prediction of our model concentrates in the visual distraction sessions (in the image on the time axis about 500 to 1000) and in the double distraction session (about 1500 to 2000).

In the first quarter of the figure we see the guide in the baseline (which after all is recognized), while in the third quarter we see the only cognitive distraction session and the prediction is low as if we were in the baseline.

Despite the poor quality result, it was decided to inspect the classifications by identifying the features that most influence the classification in order to better understand what the model is learning from the data to determine the output.

Starting from these features, we compare them with recent research and subsequently we tried to determine if the distribution of the data is the same for driving in the baseline and during cognitive distraction and if it was different from the eyes-off-road session.

6.3 Inference With Features Importance

Although the algorithm performance metrics suggest that our driver behaviour algorithms can't differentiate very well cognitive distraction, it was decided to inspect the classifications by identifying the features that most influence the classification and check if they are consistent with recent research reported in Section 3.1.3.

In this world of ever-increasing data, we use all kinds of complex ensemble and deep learning algorithms to achieve the highest possible accuracy. It's sometimes magical how these models predict, classify, recognize, and track unknown data. Accomplishing this magic and more have been and would be the goal of intensive research and development in the data science community that takes the name of eXplainable AI (XAI).

There are mainly two state-of-the-art open-source explainability techniques for machine learning models, namely LIME [61] and SHAP [60].

For LIME, an explanation is a local linear approximation of the model's behaviours. While the model may be very complex globally, it is easier to approximate it around the vicinity of a particular instance. While treating the model as a black box, LIME perturbs the instance desired to explain and learn a sparse linear model around it, as an explanation.

SHAP (SHapley Additive exPlanations) is yet another additive feature attribution method for model explainability. Its beauty lies in the fact that it unifies all available frameworks for interpreting predictions. SHAP assigns each feature an importance value for a particular prediction.

As its name suggests, SHAP is based on the Shapley value. In theory, a Shapley value considers all possible feature interactions and is global for the underlying machine learning model.

This is very different from LIME where the explanation is only local to the original model. Consequently, SHAP explanations have better properties than LIME. We can interpret individual prediction, or we can calculate Shapley values for all examples and aggregate the effect to profile global feature importance. All we need is access to the model's prediction interface and also the original training dataset in order to do the sampling approximation.

Figure 6.13 shows two beeswarm plot that indicate the most influence features that contribute for the classification calculated through the DeepExplainer provided by SHAP for two different participants.

Only 2 are reported which summarize the general behavior on different training phases. They also summarize the two network outputs considering a single output.

In fact, many significant features for the eyes-off-road are not present or if present do not reflect the real distribution necessary for the classification. This has been verified and is caused by the fact that the number of visual distraction events are much fewer than the total number of examples and, above all, much fewer than the moments in which there is cognitive distraction.

Through the beeswarm plots, we can gather a lot of information on the use of data for classification.

In fact, first of all, the features are presented in order of importance; the top features represent the most significant one while the bottom one indicates the twentieth most significant.

Each horizontal feature presents the 1000 most representative windows (chosen with a K-Medoids technique) drawn as points.

The SHAP Value (i.e. the impact of the feature in the classification) is reported on the x-axis. As we can see in the figure, there is a vertical line drawn at 0.

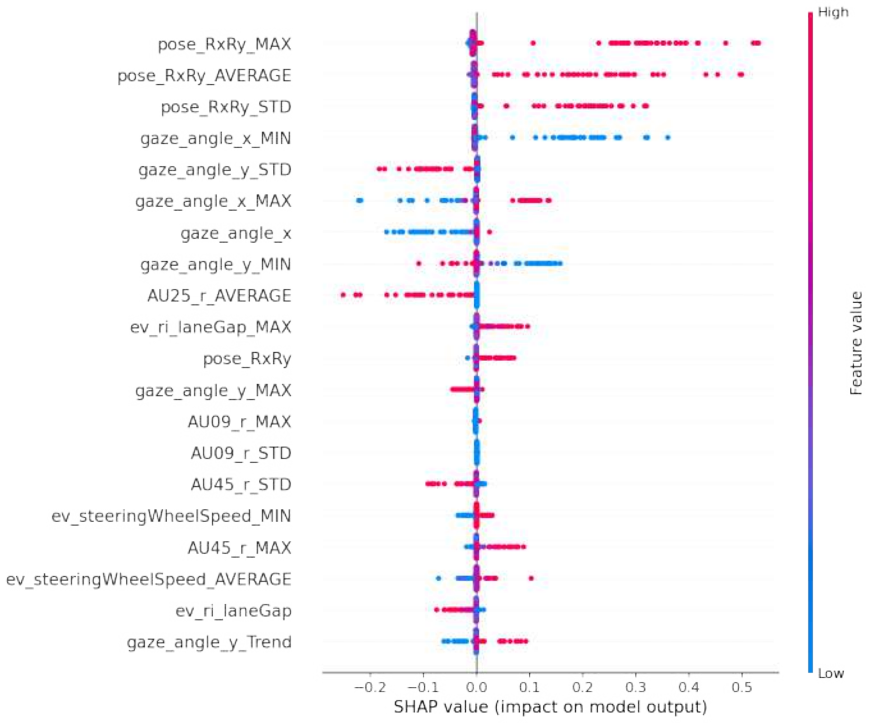
This indicates that the distribution of the feature located to the right of the gray line has a positive impact on the classification, vice versa if the distribution is on the left, it will have a negative impact.

Also from the plot we can determine if the feature value is high or low and in which cases. In fact, based on the color of the points we can determine the high or low distribution of the values.

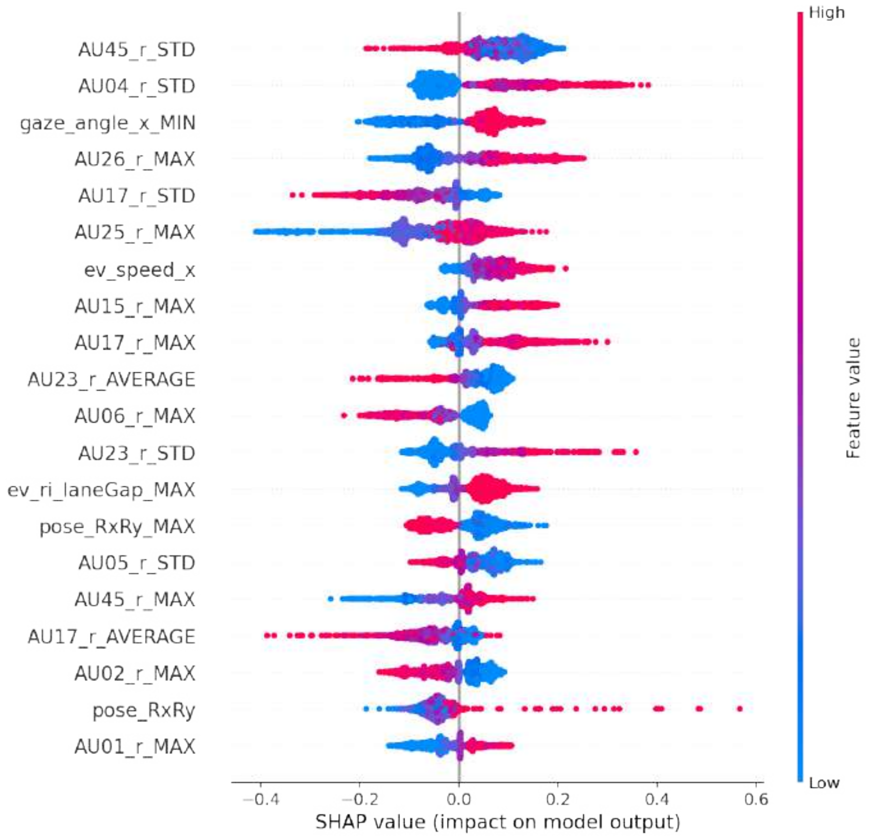
Beeswarm SHAP plot is able to show how small variations and perhaps combinations between these and other features, have an impact on the classification.

In figure 6.13 you can see beeswarm plots for the two output of the neural network. 6.13a shows 20 most important features with their impact for visual distraction classification (or eyes-off-road) and 6.13b shows the 20 ones for cognitive distraction classification.

According to SHAP the most sensitive measures on which the classification is based are in many cases based on some facial muscle expressions (action units) head-pose and some vehicle dynamics. In fact in both the reported features importance beeswarm plot we note AU45 average and max (blink frequency), AU04 std, average and max (brow lowerer), AU17 average (chin raiser), AU25 min (lips part), but also gaze angle x and y min and average (for each eye). Other important features coming from vehicular dynamics, are i.e lane gap max and std, steering wheel speed max, min, speed of longitudinal position (x).

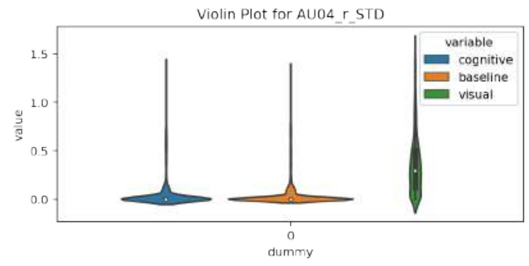


(a) Features Impact for eyes-off-road classification

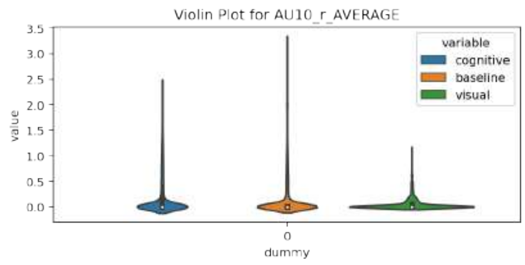


(b) Features Impact for cognitive classification

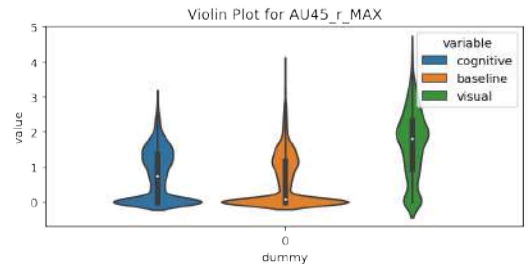
Figure 6.13: Global Features impact on classification of distracted drivers based on SHAP value



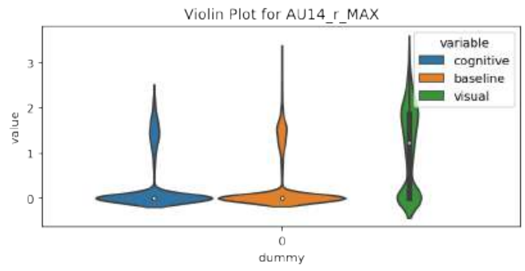
(a) Plot of Blow lowerer standard deviation



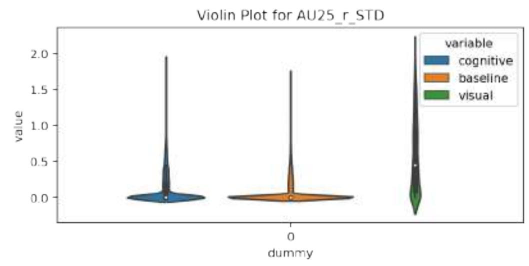
(b) Plot of Upper lip raiser average values



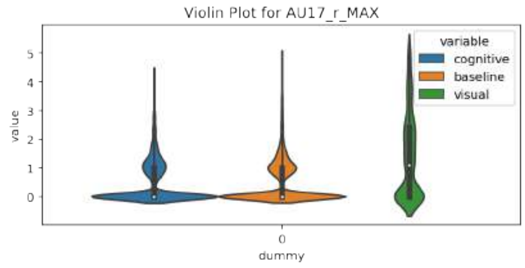
(c) Plot of eyes blinking max values



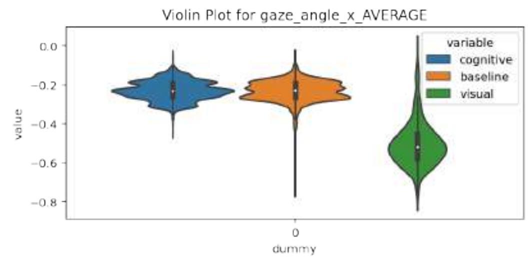
(d) Plot of Dimpler max values



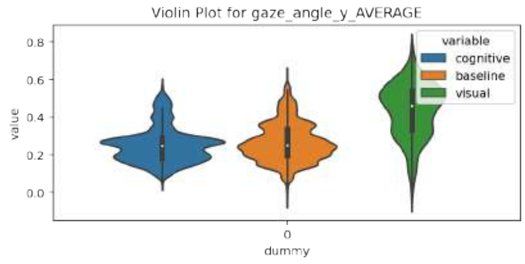
(e) Plot of lips part standard deviation



(f) Plot of chin raiser max values



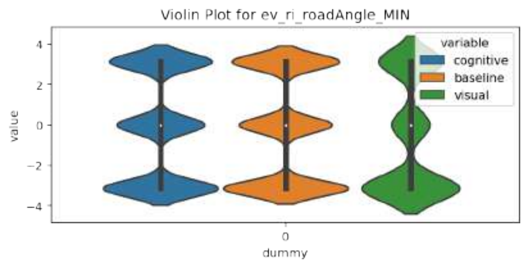
(g) Plot of gaze angle of right eye average values



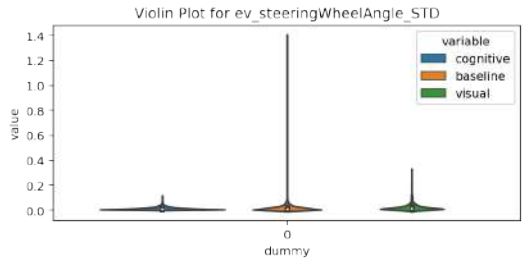
(h) Plot of gaze angle of left eye average values

Figure 6.14: Violin plots on different sessions condition of some global important features returns by SHAP

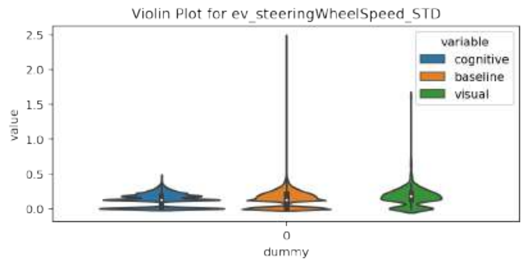
Beyond these findings, further insight can be achieved by analyzing the distribution of features across classes. Although analyzing these distributions in isolation negates the power of machine learning methods to find complex patterns across features, it provides some insight into the patterns in the data associated with classes. Figures 6.14 and 6.15 shows examples of this analysis, some violin plots partitioned by feature and distraction type.



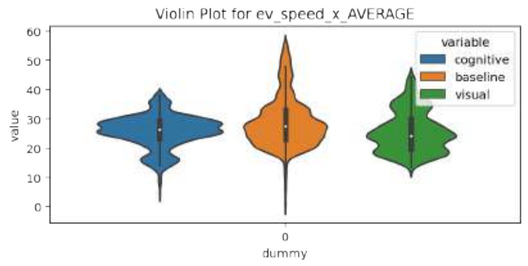
(a) Plot of road angle min values



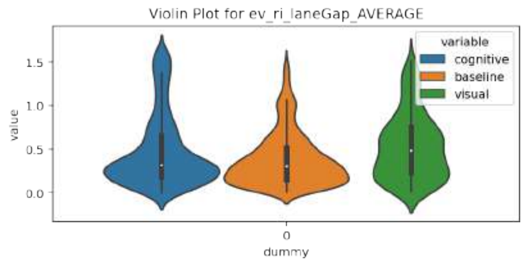
(b) Plot of Steering Wheel Angle standard deviation



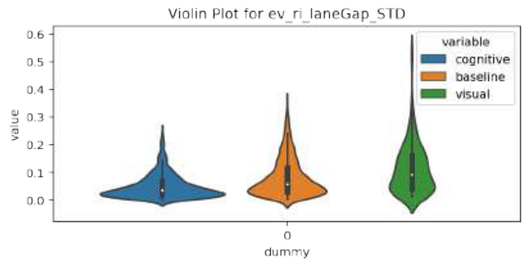
(c) Plot of Steering Wheel Speed standard deviation



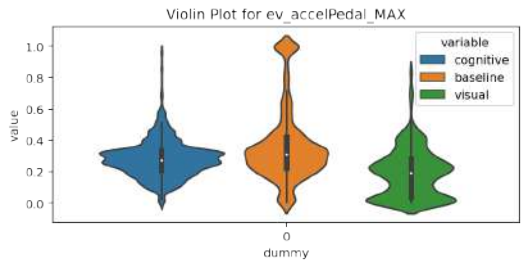
(d) Plot of speed x vector average values



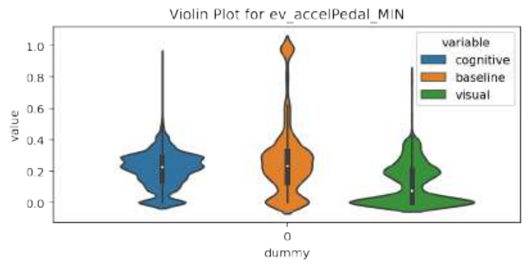
(e) Plot of lane gap average values



(f) Plot of lane gap standard deviation



(g) Plot of accelerator pedal max values



(h) Plot of accelerator pedal min values

Figure 6.15: Violin plots on different sessions condition of some global important features returns by SHAP

Each violin plot shows the distribution of a feature across the vertical axis for the three sessions analyzed (baseline, visual, cognitive). Differences in the means (shown as points on the plot) or the shapes of the distributions between the classes suggest that the measures may be useful indicators of distraction.

6.4 Discussion

The results obtained, however, are in agreement with recent research.

As you can see in 6.14 Physical Measures are plotted, i.e. the features retrieved from the internal camera through OpenFace. It is very important to notice how the distributions are very different in visual distraction driving sessions while there are no obvious differences for cognitive distraction.

This agrees with Papakostas et al. [31]. In fact, they, as in our case, processed the videos produced during data acquisition with OpenFace and determined the importance of facial expressions based on the distraction tasks.

They have shown that during phases of visual distraction (texting or interacting with the GPS) there are some discriminating AUs such as AU04, AU10, AU14, AU26, while during phases of cognitive distraction (listening to the radio and N-back tests) the AUs show few, if any, differences, especially during the N-back test. While during listening to the radio, however, small differences are found in AU25, AU26 and AU17. Looking at our important features 6.13 we notice that almost all of those mentioned are present in the 20 most important ones for the cognitive distraction. This is because in our dataset, for half of the cases of cognitive distraction, they also include visual distraction.

In confirmation of what they have shown, in 6.14a, 6.14b, 6.14d we can see the AU04, AU10 and AU14 respectively; the distributions during visual distraction are very different from the distributions in baseline. Furthermore, from the plot provided by SHAP, we can see how a high value of these features contributes positively to the cognitive distraction classification.

In the last research reported, they demonstrate that AU17 and AU25 affect the cognitive distraction with an increment induced by listening to the radio. In ours important features, AU17 and AU25 appears for the cognitive distraction classification. By inspecting the distributions, shown in 6.14e and 6.14f respectively, we realize that the importance is associated with visual distraction and also for cognitive distraction.

However, in the violin we can see that there is a very small difference in AU25 distributions during the cognitive distraction phase compared to the baseline; during cognitive distraction, the value tends to be higher, while during the baseline the value remains squeezed towards 0. To confirmation of this, we can see in figure 6.13b plots of AU25 max and AU26 max and AU17 have a positive impact on cognitive distraction classification when the value is high. Furthermore, the standard deviation of all of these have a greater impact when their value decreases.

Being the cognitive distraction phase induced through the N-back test, it requires the driver to speak. That said, the difference in the lip part distributions should be very noticeable.

The difference, however, is not very evident because the labelling of cognitive distraction includes a very long period of time in which the driver responds by saying "match" only for a few moments. Also, as already reported in Chapter 4, there are many cases where (albeit for a short time) the drivers spoke during the baseline session. This confuse the model.

Another feature that is important to the prediction is blinking 6.14c. As we can see, the distribution of values in these plots for the visual distraction session is very different from the distribution in the baseline session and it appear in the 20 greatest impact features for the visual distraction classification (figure 6.13a) and it have an positive impact when the value is high.

However, there is a lot of research showing a strong influence of the N-Back task in eyes blinking; Čegovnik et al. [48] have shown that with the increase of N in the N-Back task there is a substantial increase in blinking. In confirmation of this, deeper analysis on 6.14c shows that there is a small difference in the two distributions and it is inherent the means.

In fact, in the cognitive session, the mean appears to be higher than in the baseline session. In confirmation of this, plot in figure 6.13b show us the impact of AU45. When the value is high, the impact on the cognitive distraction classification increased. This is also demonstrated in [19].

Another very noticeable difference is the gaze. We do not notice differences between cognitive phase and baseline, but trivially we find differences between baseline and visual distraction (which we remember being eyes-off-road).

When we take our eyes off the road obviously our gaze will change direction. That's what we see in the plots 6.14g and 6.14h.

There are also some differences in vehicle dynamics in recent research that we have confirmed.

In [47] for example, the Standard deviation of lane gap is the most sensitive feature to distraction along with speed and steering mean measures.

In our case, these features appear in the 20 most influential features. Figure 6.15f shows Standard deviation of lane gap and figure 6.15e shows Average values of lane gap;

In these we can see the differences in the mean and in the distribution in the 3 different sessions. There is a major difference for the visual distraction session but we also notice a small difference in the cognitive distraction session when the value appear to decrease. However, in the plots provided in figure 6.13 we note that the lane gap value has a positive impact when the value is high. This is due to the fact that the number of cases of cognitive distraction are much higher than that of visual distraction, and we also include moments in which driver is also visually distracted and in which the driver is relatively attentive.

On the other hand, our steering wheel speed std 6.15c and mean shows no significant differences except in the form of the distribution of the visual distraction session. Despite this we note that plot in figure 6.13a, shows that an higher value of steering wheel speed min and average contribute positively in the classification. This confirms the study done in [47].

In the latter research and in [19] was also demonstrated correlation with distraction and speed. In particolar, when we engages in a cognitive distraction. In particular, when we are cognitively distracted we tend to maintain a constant and relatively low speed, while when we are distracted manually or visibly, we tend to let go of the accelerator and then slow down.

In our case, as we see in the plot 6.15d, the averages are equal but there is a difference in the shape of the distributions.

We can see how in the phase of cognitive distraction the distribution seems to be slightly more concentrated towards the center so we tend to remain at a constant speed.

While, as also demonstrated in [23], during visual distraction there is a tendency to have lower speeds. In the baseline instead, the value is uniformly distributed with a long tail upwards which indicates cases in which the driver accelerates more.

The analysis carried out on the features of greatest impact for the model, however, the speed is among the most important but has a positive impact when the value is average. When the value is low it has a negative contribution.

This could be due to the very long labeling periods and to the motorway scenario which does not allow to grasp substantial differences in speed.

Furthermore, the poor generalization of the model and therefore the differences in the driving styles of the various participants could be fundamental.

The accelerator pedal appears to be an additional important feature in identifying distraction and is very related to speed.

In fact, as you can see in 6.15g and 6.15h, where the distributions of the MIN and MAX accelerator pedal values are shown (which we remember to be a value between 0 and 1), we see notable differences for all three sessions.

During the visual distraction, as for speed, there is a tendency to press the pedal less, in fact, the shape of the distribution both for the MAX and MIN values is concentrated downwards and differs greatly from the other two sessions. This is because when we take our eyes off the road, we instinctively let go of the accelerator pedal. This leads to the collapse of values in the distribution.

In the cognitive distraction session, instead, we can see how the pressure on the pedal remains constant and concentrated on a low value but with few, if any, values of 0. This is because when we cognitively distract ourselves we tend to fix our behaviour and not to exceed.

In fact, seeing the distribution during the baseline, we can see how the value is more uniformly distributed and also comes to have a long tail towards the maximum value.

But despite this in the features of greatest impact for the classification this feature

However, we could not find confirmation for the steering wheel angle. Some recent research by Engström et al. [27] or by Jin et al. [19] suggest that driving under cognitive distraction induced by a 2-back test increased steering angle variance.

Figure 6.15b shows the distribution of the steering wheel angle standard deviation in our dataset. While the variations in the data are almost imperceptible and the averages appear to be the same, there is a small difference which could mean the opposite of the research mentioned above.

It would seem that during the phase of cognitive distraction the steering remains more fixed compared to the baseline (even if of minimal and almost intangible values). In fact, this feature does not appear in the features with greater impact in the classification

Chapter 7

Real-Time System Interaction

Since the general goal of Next Perception is the creation of a Driver Monitoring System (DMS) made up of multiple intelligent components that interact and collaborate with each other to determine the complex state of the driver (Section 1.2), our contribution was the development of the DDC. The goal is to put our module in communication with the simulator, which in turn will be connected to the remaining modules of the DMS.

Meanwhile, the driver distraction detection warning system, in a real environment, comprises a driver distraction detection sensor assembly, an in-vehicle computer processor and a driver distraction warning assembly, wherein the driver distraction detection sensor assembly detects and collects behaviour and action information of the driver and sends the behaviour and action information to the in-vehicle computer processor.

In our case, the real-time system is simulated, in fact, to integrate our classifier in the simulator environment, it was decided to create an embedded system that communicates with the simulator through the Publish / Subscribe information exchange system. In this way, our system will have both the computational and the communication side, independent from that of the simulator.

Publish/subscribe is a style of messaging application in which the providers of information (publishers) have no direct link to specific consumers of that information (subscribers), but the interactions between publishers and subscribers are controlled by pub/sub-brokers.

In a publish/subscribe system, a publisher does not need to know who uses the information (publication) that it provides, and a subscriber does not need to know who provides the information that it receives as the result of a subscription. Publications are sent from publishers to the pub/sub-broker, subscriptions are sent from subscribers to the pub/sub-broker, and the pub/sub-broker forwards the publications to the subscribers.

A typical publish/subscribe system has more than one publisher and more than one subscriber. An application can be both a publisher and a subscriber.

This style of messaging contrasts with a point-to-point style of messaging application, in which the application that sends messages needs to know the destinations of the messages that it sends.

Specifically, the MQTT Broker was used. In the diagram shown in figure 7.1 you can see the interaction and data exchange by the broker.

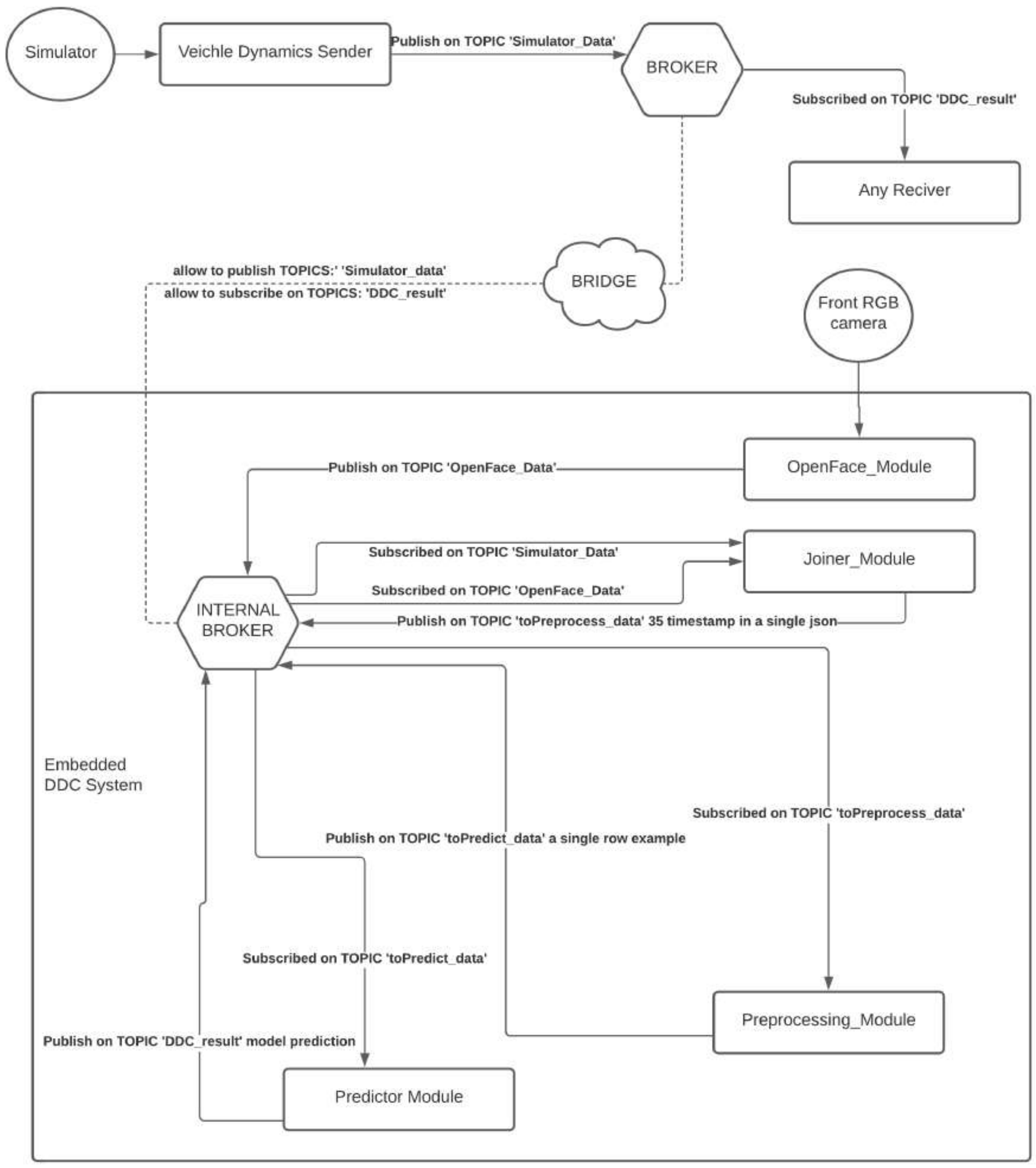

Simulator
Veichle Dynamics Sender
Publish on TOPIC 'Simulator_Data'
BROKER
Subscribed on TOPIC 'DDC_result'
Any Reciver
allow to publish TOPICS:' 'Simulator_data'
allow to subscribe on TOPICS: 'DDC_result'
BRIDGE
Front RGB camera
OpenFace_Module
Publish on TOPIC 'OpenFace_Data'
Subscribed on TOPIC 'Simulator_Data'
Joiner_Module
Subscribed on TOPIC 'OpenFace_Data'
INTERNAL BROKER
Publish on TOPIC 'toPreprocess_data' 35 timestamp in a single json
Embedded DDC System
Subscribed on TOPIC 'toPreprocess_data'
Publish on TOPIC 'toPredict_data' a single row example
Subscribed on TOPIC 'toPredict_data'
Preprocessing_Module
Publish on TOPIC 'DDC_result' model prediction
Predictor Module

Figure 7.1: Real-Time System Interaction with MQTT Brokers

As you can see in the diagram, the first circle, named ‘Simulator’, represents the sources of information coming from the CAN bus system of the simulator.

In the rectangular boxes instead, there are all the clients. The first one that we can see is ‘Vehicle Dynamics Sender’, registered to the MQTT Broker of RE:Lab, directly connected to the Simulator.

The ‘Vehicle Dynamics Sender’ has the purpose of sending internal and external vehicle dynamics data through the TOPIC ’Simulator_Data’ This Broker is connected between a bridge with another broker, used for executing internal operations.

The RE:Lab Broker broker is only responsible for sending the data and accepts only the results of the prediction as input.

This way it is unaware of everything the internal broker does. In fact, all the operations performed by the internal broker are seen as a ’black box’ to which it is sufficient to send the data with the two TOPICs and which returns the results of the prediction with a special TOPIC.

Our black box system is a Raspberry in which we have installed the Unix operating system and developed the components necessary for its operation.

As a first version, it was decided to include the video camera within our module, in fact in the figure it can be seen that it is directly connected to the Openface module.

Next, we can see the Internal Broker that puts in communication different modules. The first client that is activated is OpenFace_Module. It is responsible for capturing the video and processing it through OpenFace to extract the necessary features for the classifier from the driver’s face. Furthermore, OpenFace Sender takes care of publishing the data extracted through the TOPIC ’OpenFace_Data’.

Right after, there is the Joiner Module which is a client that is subscribed to topics of sending data (‘OpenFace_data’ and ‘Simulator_data’) and waits until it receives 35 instances from both topics and, when it receives them, sends them through the topic ’toPrepocess_data’. Subscribed on this topic there is the Preprocessing_Module which when it receives the packet containing 35 lines proceeds with the data windowing, in the same way described in the learning phase.

Having obtained a single example to predict starting from 35 instances (remembering that the window size is 35), it is sent with the TOPIC ‘toPredict_data’ to the broker.

Subscribed to this topic there is another client called Predictor_Module which having received the single data to be predicted, perform scaling, prediction, and finally, publish the result of the prediction on the topic ‘DDC_Module’.

Recall that this topic is configured with the bridge in such a way that it is seen by the main broker of RE: Lab (and not only by our internal broker).

This system will allow us to fine-tune the models and check their behaviour in real-time. Figure 7.2 shows a photo captured after the installation of our module inside the simulator environment.

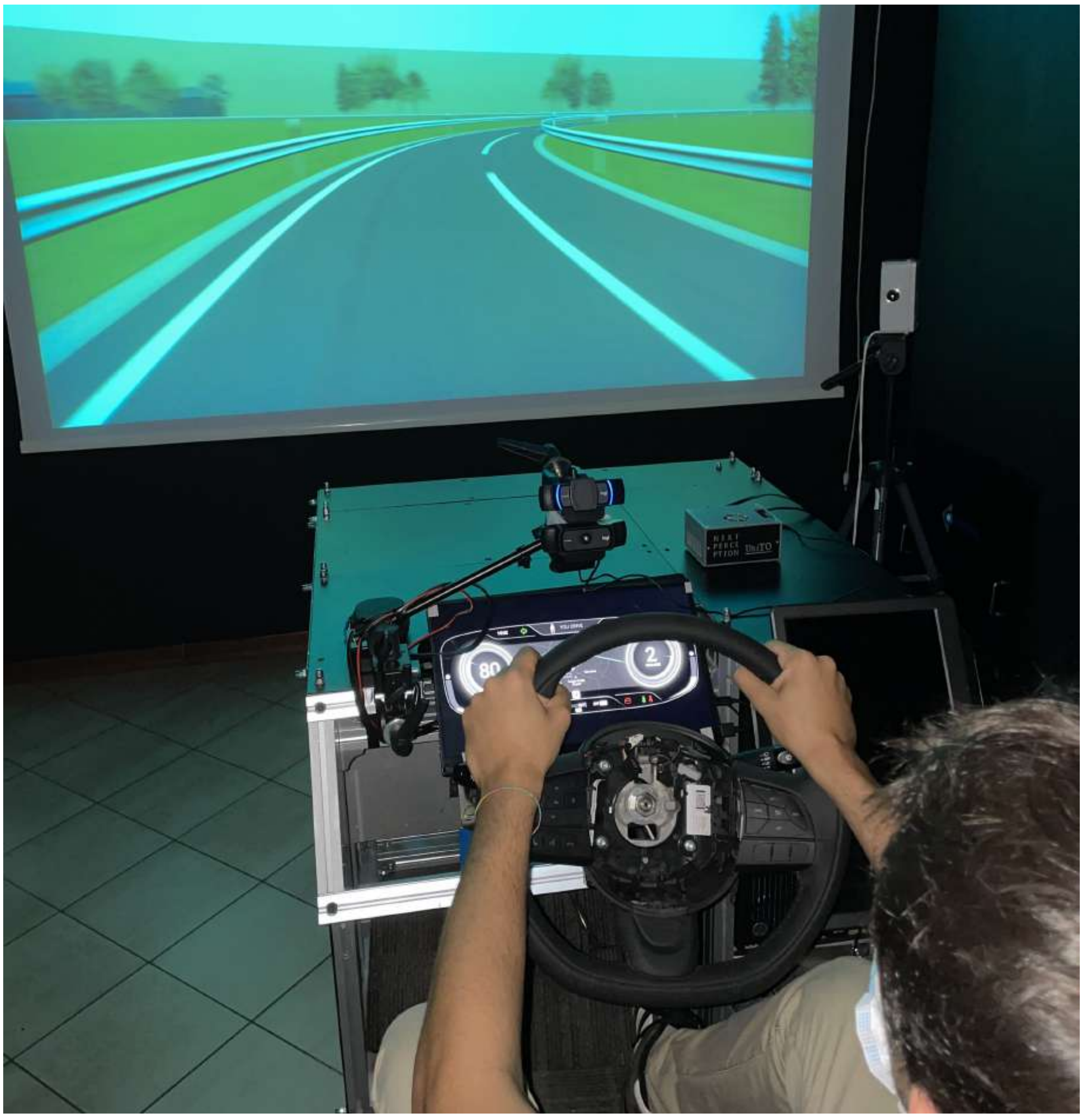


Figure 7.2: Real-Time System installed on simulator. Above the speedometer you can see the video camera and behind on the right you can see our module as a silver box with a black panel in front.

Furthermore, this also allows, in the Next Perception project context, to subscribe several clients in order to receive the predictions in real-time.

Chapter 8

Conclusion and Future Work

Driver distraction is one of the leading causes of car fatal accidents. Many solutions have been suggested to diminish driver distraction and increase driver safety. Using machine learning and deep learning methods to detect driver behaviour is one of these solutions.

In this thesis, a review of the recent research was first carried out to better understand which features to use in order to determine driving distraction.

Using a dataset produced by 26 different participants in which visual and cognitive distraction are induced, we proceeded by carrying out a feature selection (manual and automatic) and subsequently the amount of information of each single data point was increased, carrying out processing through the sliding windows' technique considering about 1.5 seconds before, thus recovering more details and statistical values such as Maximum, STD, Trend in a window.

After that, we used this dataset to train traditional machine learning techniques. The best reported are Multilayer Perceptron and 1D Convolutional neural network.

Finally, we have implemented the MLP model in an embedded system that is able to communicate with the simulator and determine whether the driver is distracted or not.

The results obtained can be very interesting from a certain point of view, as we realize that the implemented architectures do their job quite well for the eyes-off-road distraction more than the metrics tell us. The same thing is not true for cognitive distraction.

In fact, despite creating more sophisticated deep architectures and changing the learning algorithm, we don't get improvements.

After a thorough analysis, we realized that the eyes-off-road distraction labelling has some missing moments, therefore there are cases in which there is a distraction but they not were labelled as distracted, or cases of eyes-off-road not induced from the test, due to the fact that the labelling was done only in the moments included in the distraction sessions. So we are not guiding the network in the training operation in the correct way.

But even so, the final model implemented in the embedded system predicts quite well the eyes-off-road condition but with the available data, it isn't possible to obtain an f1-score higher than 92 %.

This is a plus for our model, which therefore managed to learn better than we were teaching it.

On the other hand, all models trained reach poor performance and does not precisely determine the decision boundary for cognitive distraction.

In fact, one of the results / objectives achieved in this thesis was to understand that the experiment's setting does not allow to retrieve information from the data that allow us to build a classifier for cognitive distraction.

This thesis work has shown how having an unrealistic and very simple scenario in the acquisition of data from the simulator, with very long labelling periods of cognitive distraction, acquired even in a context that is not entirely isolated for which cognitive distraction occurs even in cases of baseline driving, leads us not to be able to determine exactly the decision boundary for cognitive distraction.

After an inference analysis with the most important features on which is based the classifier, we determine that in a driving scenario without other cars and on the highway without different tasks to execute, the n-back task of cognitive distraction does not change significantly driving measurements and driver behaviour that in recent research are affected from distraction.

In addition, the extreme simplicity of the scenario, which therefore does not include intersections, traffic lights, speed limit, other cars or pedestrians, and experiment settings, does not allow us to capture and understand, for example, the variations in reaction times, braking times, time to collision, that are very important comparing cases of cognitive distraction to cases of normal driving.

Furthermore, we realize that the response to cognitive distraction is highly personal and subjective, so the normal behaviour of one driver can be similar (too similar for a model) to the distracted behaviour of another driver. This leads us to learn spurious correlations which then leads us to poor quality predictions for examples not seen previously.

With obvious and consistent effects, visual distraction can be estimated by a general model that describes the instantaneous changes of drivers' eye glance patterns, head pose, lane gap and other vehicle dynamics data and fits all drivers.

The estimated visual distraction strongly relates to crash risk and so is very important in order to prevent cars accidents. In contrast, the effects of cognitive distraction are subtle, inconsistent and cumulative.

The detection of cognitive distraction considers a comprehensive view of driver behaviour across a relatively long period of time and needs to be personalized for individual drivers.

Data mining methods are a promising approach to capture such complex relationships and having a large amount of data to analyze, should aid the recognition and elucidation of cognitive distraction effects.

As future work, in this research, we will focus on the acquisition of new data, starting from some clarifications such as the presence of other vehicles, a larger circuit with more obstacles, an isolated room in which to take the test, pedestrians that cross the road and other realistic factors.

This would allow us to consider some additional features that may be significant for the classification of the cognitive distraction, such as reaction time, time to collision, breaking events, etc.

Furthermore, in the context of research, it is important to simulate reality in the best possible way, in order to create automatic systems capable of recognizing reality well.

As final step, the Driver Distraction Classifier will be fine-tuned on the driver cockpit environment integrated into the simulator with the embedded system.

List of Figures

List of Tables

Bibliography

[1] National Highway Traffic Safety Administration (NHTSA) (2020), *https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812926.*

[2] Data Mining and Knowledge Discovery , Abanda et. al, 'A review on distance based time series classification' (2019), *https://doi.org/10.1007/s10618-018-0596-4.*

[3] Data Mining and Knowledge Discovery , Fawaz et. al, 'Deep learning for time series classification: a review' (2019), *https://doi.org/10.1007/s10618-019-00619-1.*

[4] IEEE, Conference of System of Systems Engineering (SoSE), Kluwak et. al, 'Gait Classification using LSTM Networks for Tagging System' (2020), *http://dx.doi.org/10.1109/SoSE50414.2020.9130487.*

[5] Journal of the American Medical Informatics Association , Catling et. al, 'Temporal convolutional networks allow early prediction of events in critical care' (2020), *10.1093/jamia/ocz205.*

[6] Springer, Journal of Big Data , Shorten et. al, 'A survey on Image Data Augmentation for Deep Learning' (2019), *https://doi.org/10.1186/s40537-019-0197-0.*

[7] The Fifth International Conference on Big Data, Big Data Facility, Martone et. al, 'A Feature Extraction Framework for Time Series Analysis' (2019), *https://www.researchgate.net/publication/332037382.*

[8] arXiv, Signal Processing and Machine Learning, Wen et. al, 'Time Series Data Augmentation for Deep Learning: A Survey' (2020), *arXiv:2002.12478.*

[9] arXiv, Gao et. al, 'RobustTAD: Robust Time Series Anomaly Detection via Decomposition and Convolutional Neural Networks' (2020), *arXiv:2002.09545.*

[10] arXiv, Electrical Engineering and Systems Science, Park et. al, 'SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition' (2019), *https://arxiv.org/abs/1904.08779.*

[11] PLoS ONE, Iwana et. al, 'An empirical survey of data augmentation for time series classification with neural networks' (2021), *https://doi.org/10.1371/journal.pone.0254841.*

[12] German Association of the Automotive Industry (VDA) (2019). *https://europe.autonews.com/automakers/german-industry-invest-68-billion-electriccars-and-automation.*

[13] IDTechEx research "Autonomous Cars and Robotaxis 2020-2040" (2018) *https://www.idtechex.com/fr/research-report/autonomous-cars-and-robotaxis-2020-2040-players-technologies-and-market-forecast/701.*

[14] SAE International. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (2021) *https://www.sae.org/standards/content/j3016_202104/.*

[15] Overview of the National Highway Traffic Safety Administration's, Driver Distraction Program. (2020) *https://www.nhtsa.gov/sites/nhtsa.gov/files/811299.pdf.*

[16] Accident Analysis and Prevention, M. Née, B. Contrand, L. Orriols, C. Gil-Jardiné, C. Galéra, and E. Lagarde, "Road safety and distraction, results from a responsibility case-control study among a sample of road users interviewed at the emergency room," (2019) *http://website60s.com/upload/files/4-road-safety-and-distraction-results-from-a-responsibility-case-control-study-among-a-sample-of-road-users-interviewed-at-the-emergency-room.pdf.*

[17] Frontiers in Human Neuroscience, Monika Lohani, Brennan R. Payne and David L. Strayer "A Review of Psychophysiological Measures to Assess Cognitive States in Real-World Driving" (2019) *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6434408/.*

[18] IOP Conference Series: Materials Science and Engineering, S Pradeep Kumar, Suganiya Murugan, Jerritta Selvaraj and Arun Sahayadhas "Detecting driver mental fatigue based on Electroencephalogram (EEG) signals during simulated driving" (2021) *https://iopscience.iop.org/article/10.1088/1757-899X/1070/1/012096/meta.*

[19] Hindawi, Journal of Advanced Transportation, Lisheng Jin et. al "Effect of Cognitive Distraction on Physiological Measures and Driving Performance in Traditional and Mixed Traffic Environments" (2021) *https://www.hindawi.com/journals/jat/2021/6739071/.*

[20] Accident Analysis & Prevention, Chang Wang et. al "What is the difference in driver's lateral control ability during naturalistic distracted driving and normal driving? A case study on a real highway" (2019) *https://www.sciencedirect.com/science/article/abs/pii/S0001457518305165.*

[21] ScienceDirect, Safety Science, Pushpa Choudhary et al "Effects of phone use on driving performance: A comparative analysis of young and professional drivers" (2019) *https://www.sciencedirect.com/science/article/pii/S092575351731771X.*

[22] Journal of Transportation Safety & Security, Yanli Ma et al. "Support vector machines for the identification of real-time

driving distraction using in-vehicle information systems" (2020) *https://www.tandfonline.com/doi/full/10.1080/19439962.2020.1774019.*

[23] ScienceDirect, Accident Analysis & Prevention, Kentaro Lio et al "Examining driver distraction in the context of driving speed: An observational study using disruptive technology and naturalistic data" (2021) *https://www.sciencedirect.com/science/article/pii/S0001457521000142.*

[24] Journal Open Engineering, Paweł Droździel et al "Drivers 'reaction time research in the conditions in the real traffic" (2020) *https://www.degruyter.com/document/doi/10.1515/eng-2020-0004/html.*

[25] ACM Conference on User Modeling, Adaptation and Personalization, Sebastian Zepf et al "Studying Personalized Just-in-time Auditory Breathing Guides and Potential Safety Implications during Simulated Driving" (2020) *https://www.degruyter.com/document/doi/10.1515/eng-2020-0004/html.*

[26] ScienceDirect, Accident Analysis & Prevention, Natakorn Phuksuksakul et al "Factors affecting behavior of mobile phone use while driving and effect of mobile phone use on driving performance" (2021) *https://www.sciencedirect.com/science/article/pii/S0001457520317656#!.*

[27] ScienceDirect, Applied Ergonomics, Damee Choi et al "Effects of cognitive and visual loads on driving performance after take-over request (TOR) in automated driving" (2020) *https://www.sciencedirect.com/science/article/pii/S0003687018303296.*

[28] IEEE, Sensors Journal, Imen Jegham et al "Soft Spatial Attention-Based Multimodal Driver Action Recognition Using Deep Learning" (2021) *https://ieeexplore.ieee.org/abstract/document/9177118.*

[29] ScienceDirect, Transportation Research Interdisciplinary Perspectives, Anh Son Le et al "Evaluating driver cognitive distraction by eye tracking: From simulator to driving" (2020) *https://www.sciencedirect.com/science/article/pii/S2590198219300867.*

[30] Electronics, Agapito Ledezma et al "Implementing a Gaze Tracking Algorithm for Improving Advanced Driver Assistance Systems" (2021) *https://doi.org/10.3390/electronics10121480.*

[31] ACM, Papakostas et al "Understanding Driving Distractions: A Multimodal Analysis on Distraction Characterization" (2021) *https://dl.acm.org/doi/pdf/10.1145/3397481.3450635.*

[32] ACM, Riani et al "Towards Detecting Levels of Alertness in Drivers Using Multiple Modalities" (2021) *https://dl.acm.org/doi/pdf/10.1145/3389189.3389192.*

[33] ScienceDirect, Progress in Energy and Combustion Science, Mortaza Aghbashlo et al "Machine learning technology in biodiesel research: A review" (2021) *https://www.sciencedirect.com/science/article/abs/pii/S0360128521000022.*

[34] ScienceDirect, Current Opinion in Neurobiology, Hausmanna et al "Measuring and modeling the motor system with machine learning" (2021) *https://www.sciencedirect.com/science/article/pii/S0959438821000519*.

[35] SpringerLink, Hausmanna et al "Coronavirus disease (COVID-19) cases analysis using machine-learning applications" (2021) *https://link.springer.com/article/10.1007/s13204-021-01868-7*.

[36] Department of Transportation and Infrastructure Studies, Ahangari et al "Predicting driving distraction patterns in different road classes using a support vector machine" (2020) *http://ijtte.com/uploads/2021-01-18/b2dd2cfa-f641-f429ijtte.2021.11(1).06.pdf*.

[37] Transportation Research Board, Samira et al "Distracted Driving Prediction Model Using a Bayesian Network Approach" (2020) *https://trid.trb.org/view/1735584*.

[38] MDPI, Yao et al "Classification of Fatigued and Drunk Driving Based on Decision Tree Methods: A Simulator Study" (2019) *https://www.mdpi.com/1660-4601/16/11/1935*.

[39] Journal of the Transportation Research Board, Ahangari et al "Enhancing the Performance of a Model to Predict Driving Distraction with the Random Forest Classifier" (2021) *https://journals.sagepub.com/doi/full/10.1177/03611981211018695*.

[40] ScienceDirect, Accident Analysis & Prevention, Formosa et al "Predicting real-time traffic conflicts using deep learning" (2020) *https://www.sciencedirect.com/science/article/pii/S000145751930973X*.

[41] ScienceDirect, Accident Analysis & Prevention, Li et al "Short-term prediction of safety and operation impacts of lane changes in oscillations with empirical vehicle trajectories" (2020) *https://www.sciencedirect.com/science/article/pii/S0001457519305019*.

[42] ScienceDirect, Transportation Research Part A: Policy and Practice, Cai et al "Applying a deep learning approach for transportation safety planning by using high-resolution transportation and land use data" (2019) *https://www.sciencedirect.com/science/article/pii/S0965856418310073*.

[43] Journal of Intelligent Transportation Systems, Cui et al "Convolutional neural network for recognizing highway traffic congestion" (2020) *https://www.tandfonline.com/doi/abs/10.1080/15472450.2020.1742121*.

[44] Journal of IEEE, Zhang et al "Research on Traffic Vehicle Behavior Prediction Method Based on Game Theory and HMM" (2020) *https://ieeexplore.ieee.org/abstract/document/8984290*.

[45] ScienceDirect, Accident Analysis & Prevention, Gong et al "Multi-Objective reinforcement learning approach for improving

safety at intersections with adaptive traffic signal control" (2020) *https://www.sciencedirect.com/science/article/pii/S0001457520303948.*

[46] IEEE, 27th Mediterranean Conference on Control and Automation, Hina et al "Machine Learning Techniques for Cognition of Driving Context" (2019) *https://ieeexplore.ieee.org/abstract/document/8798589.*

[47] SAGE Journal, Hum.Factors, McDonald et al "Classification of driver distraction: A comprehensive analysis of feature generation, machine learning, and input measures" (2019) *https://journals.sagepub.com/doi/abs/10.1177/0018720819856454.*

[48] ScienceDirect, Journal Applied Ergonomics, Čegovnik et al "An analysis of the suitability of a low-cost eye tracker for assessing the cognitive load of drivers" (2018) *https://doi.org/10.1016/j.apergo.2017.10.011.*

[49] IEEE Transactions on Neural Networks and Learning Systems, Li et al "Deep neural network for structural prediction and lane detection in traffic Scene" (2018) *https://www.researchgate.net/publication/295076922.*

[50] Springer, Journal Knowl Inf Syst 60, 1549–1564, Botta et al "Real-time detection of driver distraction: random projections for pseudo-inversion-based neural training" (2019) *https://doi.org/10.1007/s10115-019-01339-0.*

[51] ScienceDirect, Accident Analysis & Prevention, Arvin et al "Safety critical event prediction through unified analysis of driver and vehicle volatilities: Application of deep learning methods" (2021) *https://doi.org/10.1016/j.aap.2020.105949.*

[52] IEEE, vol. 9, pp. 60063-60076, Kashevnik et al. "Driver Distraction Detection Methods: A Literature Review and Framework" (2021) *https://doi.org/10.1109/ACCESS.2021.3073599.*

[53] Researchgate, Kouchak et al "Driver Distraction Detection Using Deep Neural Network" (2019) *https://www.researchgate.net/publication/338382688.*

[54] American Journal of Artificial Intelligence, Tamas et al "Real-Time Distracted Drivers Detection Using Deep Learning" (2019) *https://www.researchgate.net/publication/338382688.*

[55] IEEE, Computer Vision and Pattern Recognition, Lugaresi et. al, A. "MediaPipe: A Framework for Perceiving and Processing Reality" (2019) *https://arxiv.org/abs/1906.08172.*

[56] Electrical Engineering and Systems Science, Sengupta et. al, A. "mm-Pose: Real-Time Human Skeletal Posture Estimation using mmWave Radars and CNNs" (2019) *http://dx.doi.org/10.1016/j.patcog.2017.06.009.*

[57] Pattern Recognition, Patacchiola, M., & Cangelosi, A. "Head pose estimation in the wild using Convolutional Neural Networks and adaptive gradient methods" (2017) *http://dx.doi.org/10.1016/j.patcog.2017.06.009.*

[58] IEEE, Baltrusaitis et al "OpenFace 2.0: Facial Behavior Analysis Toolkit" (2018) *https://ieeexplore.ieee.org/document/8373812*.

[59] Towards Data Science, Cheong et al "Four ways to quantify synchrony between time series data" (2019) *https://towardsdatascience.com/four-ways-to-quantify-synchrony-between-time-series-data-b99136c4a9c9*.

[60] arXiv,, Lunderberg et al. "A Unified Approach to Interpreting Model Predictions" (2017) *https://ieeexplore.ieee.org/document/8373812*.

[61] arXiv, Ribiero et al ""Why Should I Trust You?": Explaining the Predictions of Any Classifier" (2016) *https://arxiv.org/abs/1602.04938*.