POLITECNICO DI TORINO

Master of Science in Mechanical Engineering

Master Thesis

Development of a Neural Network based Energy Management System (EMS) for a plug-in Hybrid Electric Vehicle

Supervisors:
Prof. Federico MILLO
Prof. Luciano ROLANDO
Ing. Luca PULVIRENTI

Candidate:
Luigi TRESCA

October 2021

To my family

La mente non è un vaso da riempire, ma un fuoco da accendere

Plutarco

Abstract

The increasingly stringent limitations in terms of CO_2 and pollutant emissions imposed by Regulators, such as the European Union (EU), are leading car manufacturers to increase the electrification level of their fleets. Hybrid Electric Vehicles (HEVs) can be seen as an intermediate step in the transition from the traditional Internal Combustion Engine (ICE) vehicles towards the fully electric ones.

Actually, thanks to their capacity for exploiting the benefits of both propulsion systems while mitigating their drawbacks, hybrid powertrains are receiving lots of attention and are continually increasing their market share.

However, the introduction of, at least, an auxiliary electric machine introduces an additional degree of freedom that, to fully exploit all the benefits provided by the electrification, has to be managed by an ad-hoc designed powertrain control strategy.

In the past, several strategies have been proposed to design the high-level controller, named Energy Management System (EMS): i.e., the layer controlling the power split among the actuators. Thanks to the increased potentialities of Artificial Intelligence (AI) techniques in effectively solving complex parameterization tasks, the design of the EMS exploiting AI is being deeply investigated in the literature.

In this framework, this thesis concerns the design of an EMS through the exploitation of deep learning techniques: the AI models allow to describe high non-linear relationship among the data characterizing the problem. For achieving sub-optimal results, the AI models have been trained off-line on a wide range of potential driving and traffic scenarios by providing the optimal solutions given by an optimization control algorithm, namely Dynamic Programming (DP).

The proposed methodology is tested, by means of numerical simulation, on a plug-in HEV, available on the European market: the model had been previously developed in GT-SUITE and Simulink environment and validated against experimental data.

The simulations proved that the proposed approach is able to achieve quasi-optimal results in terms of fuel consumption minimization while, at the same time, being theoretically feasible in a vehicle Electronic Control Unit (ECU).

Sommario

Le aziende del comparto automobilistico stanno incrementando il livello di elettrificazione della loro flotta di veicoli, spinte dalle sempre più stringenti limitazioni in termini di emissioni di CO_2 ed inquinanti imposte dagli Enti Regolatori, come l'Unione Europea (UE).

I veicoli a trazione ibrida possono essere visti come una soluzione intermedia fra i tradizionali veicoli dotati di motore a combustione interna e i veicoli dotati di trazione completamente elettrica.

I powertrain ibridi, grazie alla loro capacità di enfatizzare i benefici di entrambi i sistemi di propulsione e di limitarne gli svantaggi, stanno ricevendo molte attenzioni da parte delle case costruttrici, diffondendosi sempre più sul mercato autonomistico.

Tuttavia, l'introduzione di una o più macchine elettriche ausiliarie aumenta la complessità del sistema di propulsione, aggiungendo un ulteriore grado di libertà che deve essere correttamente gestito tramite un dedicato sistema di controllo del powertrain. per poter sfruttare a pieno le potenzialità fornite dall'elettrificazione.

Negli anni, numerose strategie sono state proposte per progettare il controllore di alto livello, chiamato Energy Management System (EMS), ovvero il livello dedicato alla gestione della suddivisione di potenza (power split) da richiedere agli attuatori per soddisfare la richiesta di potenza proveniente dal veicolo. Grazie alle crescenti potenzialità dell'Intelligenza Artificiale (IA) nel risolvere complessi problemi di parametrizzazione, in letteratura sono state proposte diverse soluzioni che sfruttano l'IA nella progettazione dell'EMS di veicoli ibridi.

In questo lavoro di tesi viene proposta la progettazione dell'EMS utilizzando tecniche di deep learning: l'utilizzo di modelli di IA permette di descrivere relazioni fortemente non-lineari tra i dati che caratterizzano il problema. L'obiettivo è quello di ottenere una soluzione sub-ottima, allenando il modello di IA tramite un vasto database di potenziali scenari di guida e traffico, utilizzando come target la soluzione ottima ottenuta tramite un algoritmo di ottimizzazione, chiamato Dynamic Programming (DP).

La metodologia proposta è stata testata, tramite strumenti di simulazione numerica, su un veicolo ibrido plug-in, disponibile sul mercato europeo, il cui modello è stato precedentemente sviluppato in GT-SUITE e Simulink e validato su una grande quantità di dati sperimentali.

L'approccio modellistico proposto è in grado di fornire una soluzione quasi-ottima in termini di minimizzazione dei consumi di combustibile ed è, teoricamente, implementabile su una centralina elettronica di un veicolo.

Index

Symbols

Abbreviations

AC:	Alternate Current
ADAM:	Adaptive Moment Estimation Method
AI:	Artificial Intelligence
ANN:	Artificial Neural Network
AT:	Automatic Transmission
AWD:	All Wheel Drive
BAS:	Belt Alternator Starter
BEV:	Battery Electric Vehicle
biLSTM:	Bidirectional Long Short-Term Memory
BMEP:	Brake Mean Effective Pressure
BSFC:	Brake Specific Fuel Consumption
CD:	Charge Depleting
CFD:	Computational Fluid Dynamic
CS:	Charge Sustaining
CV:	Conventional Vehicle
DC:	Direct Current
DNN:	Deep-Neural Network
DP:	Dynamic Programming
DRL:	Deep-Reinforcement Learning
ECMS:	Equivalent Consumption Minimization Strategy
ECU:	Electronic Control Unit
EM:	Electric Machine
EMS:	Energy Management System
EPA:	Environmental Protection Agency
EU:	European Union
EV:	Electric Vehicle
FEAD:	Front-End accessory Drive

FHEV: Full Hybrid Electric Vehicle

FTP75: EPA Federal Test Procedure

GDM: Gradient Descent Method

GHG: Greenhouse Gas

GRU: Gated Recurrent Unit

HEV: Hybrid Electric Vehicle

HV: High Voltage

ICE: Internal Combustion Engine

ICEV: Internal Combustion Engine Vehicle

kNN: k-Nearest Neighbours

Li-NMC: Lithium-Nickel Manganese Cobalt oxide

LSTM: Long Short-Term Memory

MGU: Motor Generator Unit

ML: Machine Learning

NEDC: New European Driving Cycle

NN: Neural Network

NN-EMS. Neural Network-Energy Management System

OOL: Optimal Operating Line

PEMS: Portable Emissions Monitoring System

PHEV: Plug-in Hybrid Electric Vehicle

PID: Proportional-Integrative-Derivative

PM: Permanent Magnet

PMSP: Permanent Magnet Synchronous Machine

RB: Rule Based

RDE: Real Driving Emissions

RL: Reinforcement Learning

RMSE: Root Mean Square Error

RMSPROP: Root Mean Square Propagation Method

RNN: Recurrent Neural Network

RPM: Revolutions Per Minute

RWD: Rear Wheel Drive

SGDC: Stochastic Gradient Descent Method

SOC: State of Charge

SVM: Support Vector Machine

TC:	Torque Converter
TTR:	Through-The-Road
V2X:	Vehicle-to-Everything
WLTC:	Worldwide Harmonized Light Vehicle Test Cycle

Definitions

$\dot{m}_f$:	Fuel rate
C_0, C_1, C_2:	Coast down coefficients
C_d:	Aerodynamic drag coefficient
F_{acc}:	Acceleration force
F_{aero}:	Aerodynamic resistance
F_{brake}:	Brake force
F_{grade}:	Road slope resistance
F_{pwt}:	Powertrain tractive force
F_{res}:	Resistance force
F_{roll}:	Rolling resistance
F_{tot}:	Total tractive force at the wheel
I_{EM}:	Electric motor inertia
I_{ICE}:	Internal combustion engine inertia
I_v^2:	Square velocity energetic index
$I_{wh,f}$:	Front wheel inertia
$I_{wh,r}$:	Rear wheel inertia
M_{eq}:	Equivalent vehicle mass
M_{veh}:	Vehicle mass
$P_{EL,gen}$:	Electrical power request by generator unit in a series HEV
P_{EM}:	Electrical power requested by motor generator
P_{eng}:	Mechanical power requested by internal combustion engine
P_{gb}:	Inlet gearbox power
P_{gen}:	Mechanical power requested by motor generator
Q_{max}:	Maximum battery charge
$R_{h,parallel}$:	Parallel HEV hybridization ratio
$R_{h,series}$:	Series HEV hybridization ratio
SoC_0:	Initial state of charge
SoC_f:	Final state of charge
c_1:	Cost function scale factor

c_{roll}: Rolling resistance coefficient

d_E: Euclidean distance

d_c: Chebyshev distance

i_{fd}: Final drive ratio

i_{gb}: Gearbox ratio

m_f: Total fuel consumption

r_{wh}: Wheel radius

$\bar{\bar{h}}$: Candidate output vector

β_1: Gradient decay factor

β_2: Square decay factor of the moving average

η_{gb}: Total gearbox efficiency

η_{gen}: Total motor generator efficiency

ρ_{air}: Air density

ω_{gb}: Gearbox rotational speed

ω_{wh}: Wheel rotational speed

CO_2: Carbon dioxide

$\cos(\theta)$: Cosine distance

$Cost$: Cost function

$E(y)$: Expected value

$E(\theta)$: Error function

J: Dynamic programming cost function

NO_x: Nitrogen oxide

$P(A|B)$: Conditional probability

$Q(t)$: Instantaneous battery charge

T: Total travel time

V: ICE displacement

a: Acceleration

b: Bias

e: ICE state

$f(\cdot)$: Activation function

f_t: Forget gate

g: Gravitational acceleration

g_t: Cell candidate gate

i:	Electric battery current
i_t:	Input gate
n:	ICE speed in rad/s
o_t:	Output gate
t:	Time
u:	Dynamic programming control variable
v:	Longitudinal vehicle speed
$\overline{W}$:	Weight vector
$\bar{h}$:	Output vector
$\bar{x}$:	Input vector
$\bar{z}$:	Update gate vector
α:	Learning rate
γ:	Momentum
δ:	Road grade
θ:	Parameter vector
σ:	Standard deviation

List of Tables

List of Figures

1 Hybrid Electric Vehicles

1.1 Global Warming Problem and Emission Regulations

Nowadays, global warming is one of the most serious problems that humanity has to face.

The emissions in the atmosphere of Greenhouse gases (GHG), elements that block the reflected solar radiation on the ground, are the major cause of the increment of the earth temperature. These emissions are related to the energy sector, from energy generation industries to buildings heating sectors to road transportation sector [1].

The latter is responsible for 23% of energy-related CO_2 emissions, which is the most common greenhouse gas because it is one the product of the combustion process of fossil fuels and has been pointed out as the main responsible for global warming [2].

To tackle this problem, in the past years, a series of CO_2 emissions limitations have been imposed from the European Union (EU), in Europe, to car manufacturers. Firstly, these limitations were not mandatory but based on a voluntary agreement, with some awards for those who were able to respect them.

However, since 2015, a fleet average CO_2 emissions target of 130 g/km, on NEDC cycle, was imposed on all passenger cars manufacturers, with a penalty of 95€ per vehicle for each g/km exceeding the target.

The target is intended for decreasing with the new regulations and, from 2020, is set to 95 g/km and it is supposed to reduce by 15% by 2025 and by 37.5% compared to 1990 levels by 2030 [3][4].

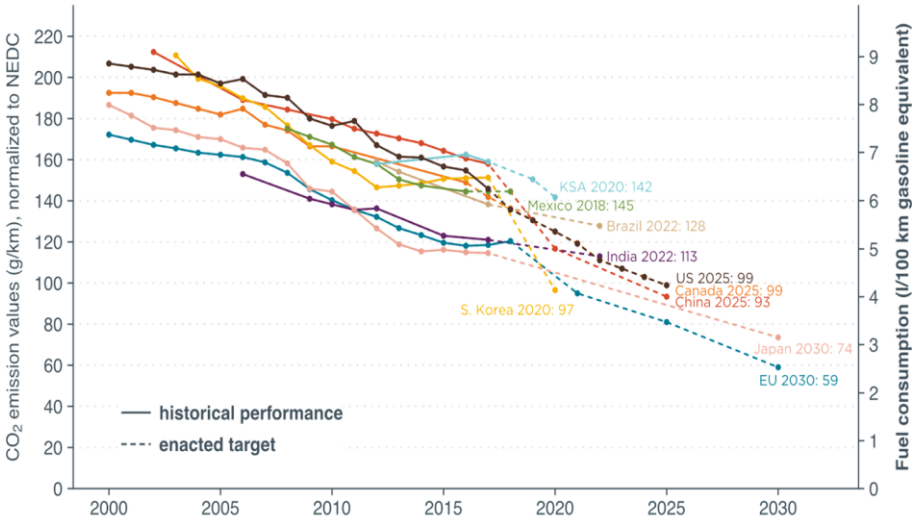

Figure 1.1: CO_2 emission target trend

A stricter limitation has been proposed on 14.07.2021, called *FIT for 55*. This climate package intends to reach the 55% of the target imposed in 1990 within 2030 in terms of CO_2 emissions, with the goal of reaching the carbon neutrality within 2050. It is a very optimistic perspective since the temporal target is very close to achieve a so drastic limitation.

Even if the modern internal combustion engines have reached remarkable values of efficiency, the target is very strict and new propulsions forms have to be adopted to reach this target, also because, in a driving cycle, the engine is forced to work at low-efficiency points for most of the time, an aspect that nullifies the increment in terms of efficiency [2]. A hybrid powertrain could help the internal combustion engine to move these operating points to more efficient zones, increasing the average efficiency of the engine and so decreasing the CO_2 emissions.

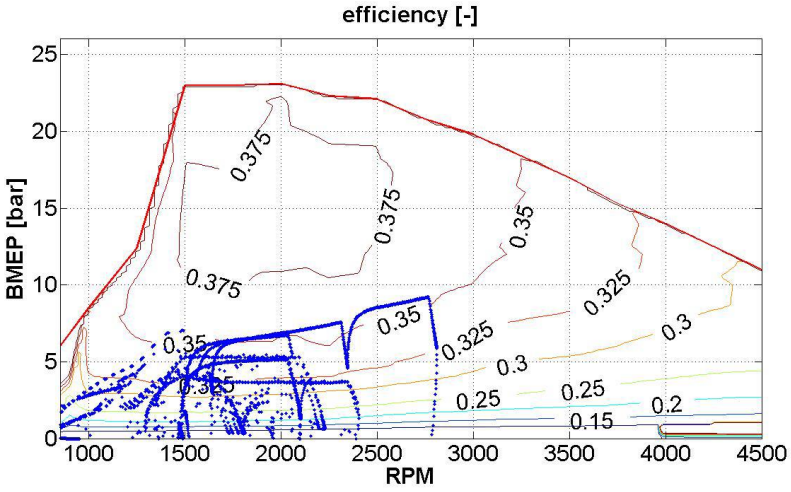

Figure 1.2: Engine operating points in a traditional pure ICE powertrain application[2]

Powertrain electrification can play a key role since it can because it can combine the advantages of two propulsion systems, such as the electric and the thermal ones, it allows using a smaller internal combustion engine (downsizing), saving more fuel and so having a softer contribution in terms of CO_2 emissions.

A pure electric powertrain approach is not ready to fully penetrate the market for two main reasons: the low energetic density of the electric energy, compared to the fossil fuels, that limits the range of electric vehicle and impose to use a very heavy pack of batteries to store the energy on board. The other point is the infrastructure that is not ready yet to sustain a great number of electric cars that have to recharge their batteries, both due to the limited quantity of recharge points, both to the energy grid that is not ready to face continuous peak of energy demands.

Considering both internal combustion engine efficiency limits and fully electric vehicle problems, nowadays one of the most interesting solution to reduce CO_2 emission in the atmosphere is to drive the vehicle with a hybrid powertrain.

1.2 Hybrid Propulsion System Description

Any propulsion system that combines two or more sources of energy, installed onboard, is defined as hybrid propulsion system [2][5].

Different sources of power could be exploited, however, for traction application, the most common are the traditional fossil fuels, burned in an internal combustion engine, and electric energy, used by one or more electric motors, intending to transform the electric energy stored in the battery, in mechanical energy.

This kind of propulsion system has the advantage of reducing fuel consumption, compared with a traditional internal combustion engine propulsion system. The required power from the load can be obtained by exploiting both the electric machine and the internal combustion engine, and it is possible also to exploit the electric machine to avoid that the internal combustion engine operating points are located in low-efficiency regions, allowing to increase the overall efficiency of the engine, that is smaller compared to the efficiency of the electric motor.

This goal can be achieved in two different ways:

- Moving the operating points through the electric machine: the electric machine provides a torque, that can be a motor torque or a braking torque, to modify the internal combustion engine torque, moving its operating point trying to reach the highest efficiency region at a fixed rotational speed.
- Downsizing: having more energy sources onboard allows to use of a smaller internal combustion engine that can work at higher torque because higher load usually means higher efficiency.

Another advantage is related to the presence of the electric machine that can be used as an electric current generator to decelerate the vehicle when the required power is negative. It allows to transform and exploit the kinetic energy that, in a traditional mechanical braking system, will be lost, to recharge the battery. This functionality is called regenerative braking.

However, increasing the number of energy sources installed onboard, increases the complexity of the system both from a mechanical and a control point of view [5][6]

The latter is a crucial aspect because the energy management system must be optimized to exploit the full potential of this powertrain. But this operation is not easy and depends on different variables (e.g., powertrain architecture, plug-in vehicle, electric machine size, etc.).

1.3 Powertrain Architecture

As it is possible to understand from the above sections, a hybrid powertrain is not a fixed solution, but it can be designed in different ways, depending on the request of the vehicle.
Due to the presence of these degrees of freedom, hybrid powertrain architectures need to be classified, to distinguish architectures with different features.
The first classification of HEV can be done considering how all the power sources installed on board are connected. Different powertrain architectures enable different functionality of the vehicle [5][6].

- **Series architecture**: in this configuration, there is no mechanical link between the internal combustion engine and the wheels. This configuration allows the internal combustion engine to operate in a narrow power range near to the optimum efficiency. Moreover, the internal combustion engine has to be coupled to an electric machine, to transform the mechanical energy from the fuel into electric energy that can be stored in the battery.
  To drive the vehicle a second electric machine is present and works mainly as a motor and has to be designed according to the peak power that the vehicle can require. This architecture has the drawback of double energy conversion from chemical to electrical, with the couple engine-first electric machine, and a second conversion from electrical to mechanical with the main electric motor; thus, it is not so efficient from the energetic point of view. Furthermore, using two electric machines introduce another complexity from the design point of view.

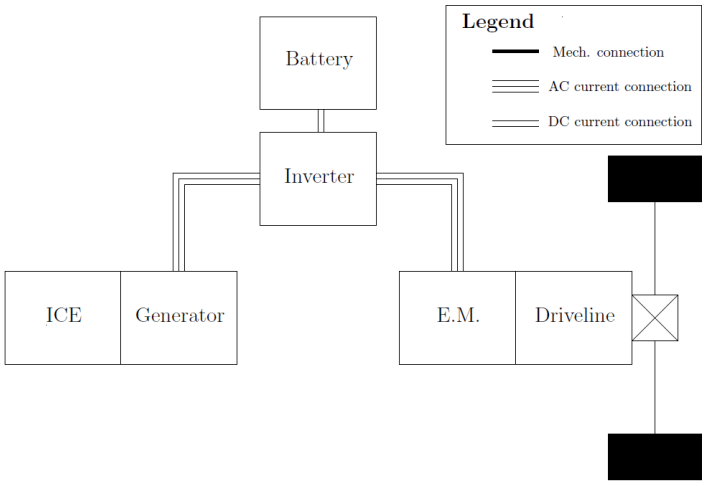

Figure 1.3: Scheme of a series hybrid architecture[2]

All these drawbacks lead to use the architecture only as a range extender configuration, that is a particular battery electric vehicle, that uses the engine only if the battery is empty, increasing the range of the vehicle to reach a recharge point, avoiding the so-called "recharge anxiety" of the users.

It is possible to define a parameter that, based on the relative size of the machines installed onboard, define a degree of electrification of the vehicle, called *hybridization ratio* [2][5][6].

$$R_{h,series} = \frac{P_{EL,GEN}}{P_{EM}} \tag{1.1}$$

Concerning the series architecture, it ranges from a value of 0 that represents a pure electric vehicle to a value of 1 that represents the electric transmission.

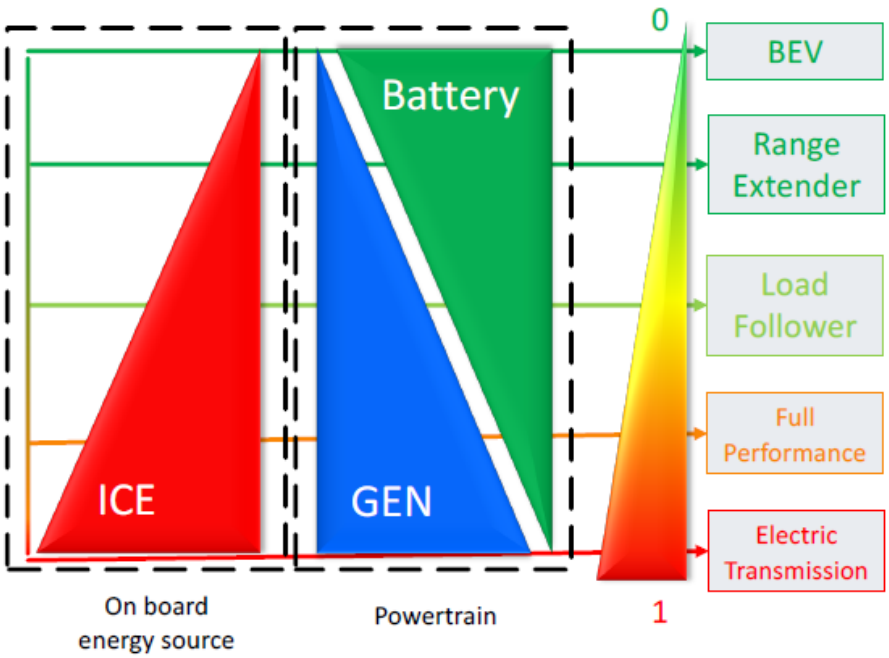

Figure 1.4: Series hybridization ratio[6]

- **Parallel architecture**: in this configuration, it is possible to perform power splits between the different power sources installed on board since the wheels are connected both at the internal combustion engine and the electric machine.

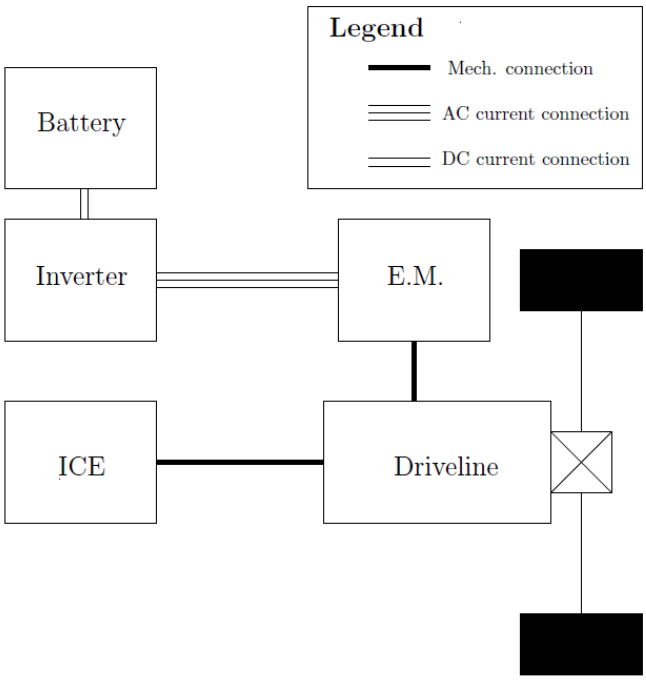

Figure 1.5: Scheme of a parallel hybrid architecture[2]

This architecture, compared to the series one, eliminates the inefficiency related to the double energy conversion and allows to downsize ~~of~~ both the machines because the peak power can be obtained using both the machines that sum their powers if parallel mode propulsion is used. Another advantage is that only one electric machine is enough to exploit the engine to recharge the battery, working as a generator, and to drive the vehicle, working as a motor in the parallel mode, obtaining a simpler mechanical design than the series architecture, but a more complex control system since the possibility of multiple propulsion methods.

For a parallel architecture too, it is possible to define a *parallel hybridization ratio* [2][5][6], that has the same meaning of the series one:

$$R_{h,parallel} = \frac{P_{EM}}{P_{ICE} + P_{EM}} \qquad 1.2$$

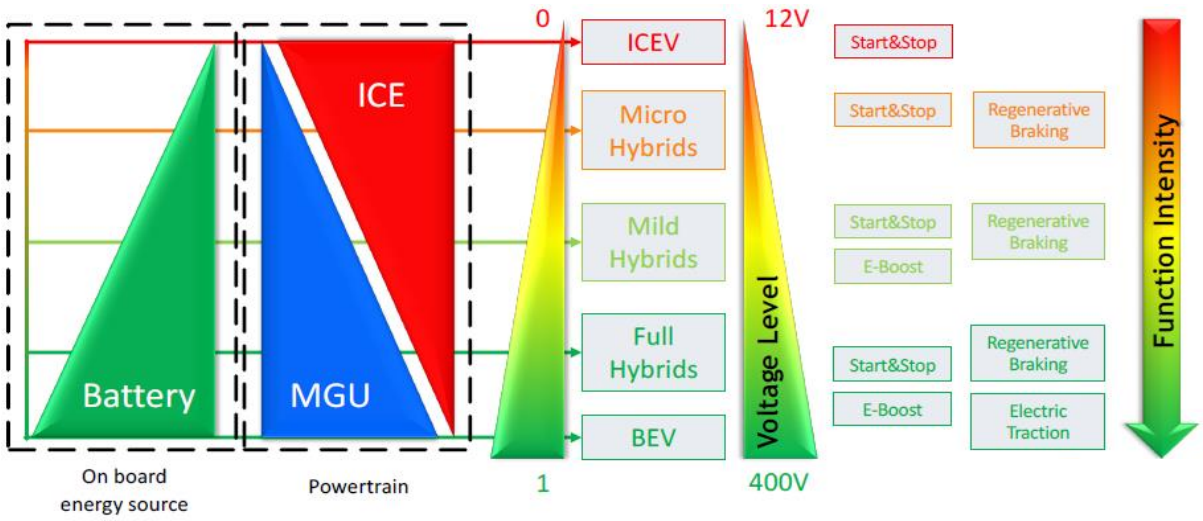

Figure 1.6: Parallel hybridization ratio[6]

It ranges from 0, representing a traditional vehicle, to 1 which is a full electric vehicle. The intermediate values represent the possible hybrid configurations that enable more functionality if the parallel hybridization ratio increases because the size of the electric machine is higher, spacing from the micro-hybrid, in which the electric machine is very small and allow just the start and stop and regenerative braking functionality, permitting to save fuel at idle operation, to the full hybrid, in which it is also possible to define a pure electric range, that is the distance the vehicle can cover with a pure electric propulsion method.

- **Complex architecture**: this architecture is obtained by increasing the number of traction systems, enabling the possibility of obtaining a parallel and a series architecture on the same

vehicle. This increases the complexity of the mechanical design and the control system too. If all the machines are linked together using a system of clutches, a series/parallel architecture is obtained, while if a planetary gearbox is used a power-split architecture is obtained.

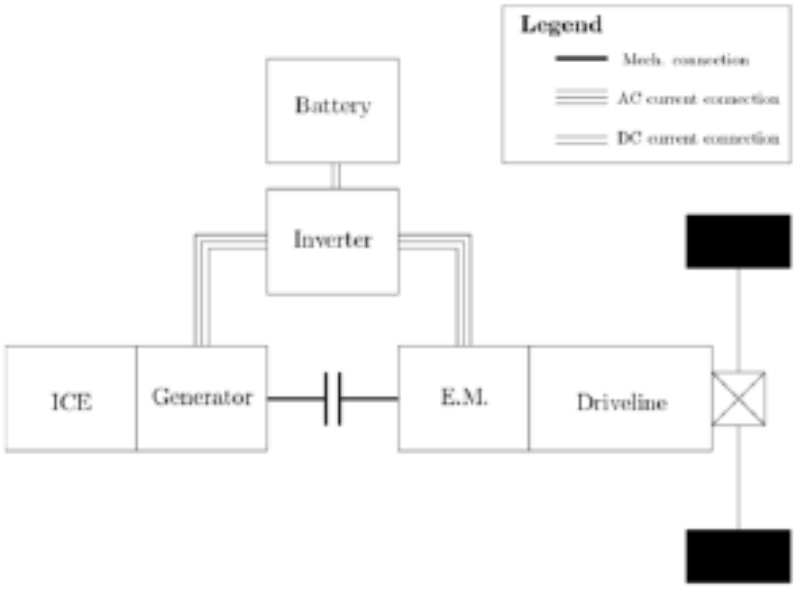

Figure 1.7: Scheme of series/parallel configuration[2]

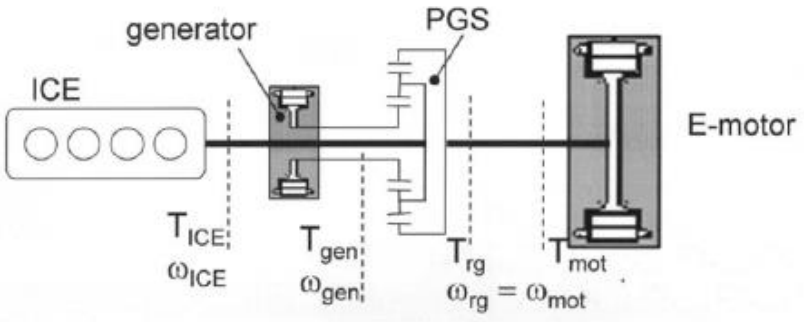

Figure 1.8: Scheme of a power-split configuration[2]

1.4 Electric Machine Position in a Parallel Architecture

In a parallel configuration, an additional classification is defined, considering the electric machine position [4][6]:

- **P0**: the electric machine is connected to the engine through a belt on the Front-End Accessory Drive (FEAD), obtaining the so-called Belt Alternator Starter (BAS). The size of the electric machine is limited and so just regenerative braking and start and stop, with a little engine assistant, are possible.

- **P1**: the electric machine is directly connected to the crankshaft and the same functionalities of a P0 configuration are possible, with the advantage of eliminating a source of losses such as the belt.
- **P2**: the electric machine is positioned between the engine and the transmission and has a greater size than the above-mentioned configurations, enabling more hybrid functionalities. A clutch allows disconnecting the engine from the electric machine when the pure electric propulsion mode is exploited. Since both the machines are located on the same shaft, if the clutch is closed, they have the same speed.
- **P3**: the transmission is located between the electric machine and the internal combustion engine. This configuration allows to better exploit the characteristic of the electric machine since it can work at a higher speed than the engine, if the transmission ratio allows to exploit this feature, and the transmission allows the machines to rotate at different speeds. A smaller electric machine can be used because its size is defined by the torque and, due to the higher speed, the same power can be obtained with a smaller torque and so a smaller machine. The additional degree of freedom of electric motor speed increases the complexity of this configuration compared to the above-mentioned.
- **P4**: the two machines are not mechanically connected because they are located on different axles, but their connection is Through-The-Road (TTR). This architecture allows having a four wheels drive vehicle, without a central differential gear.

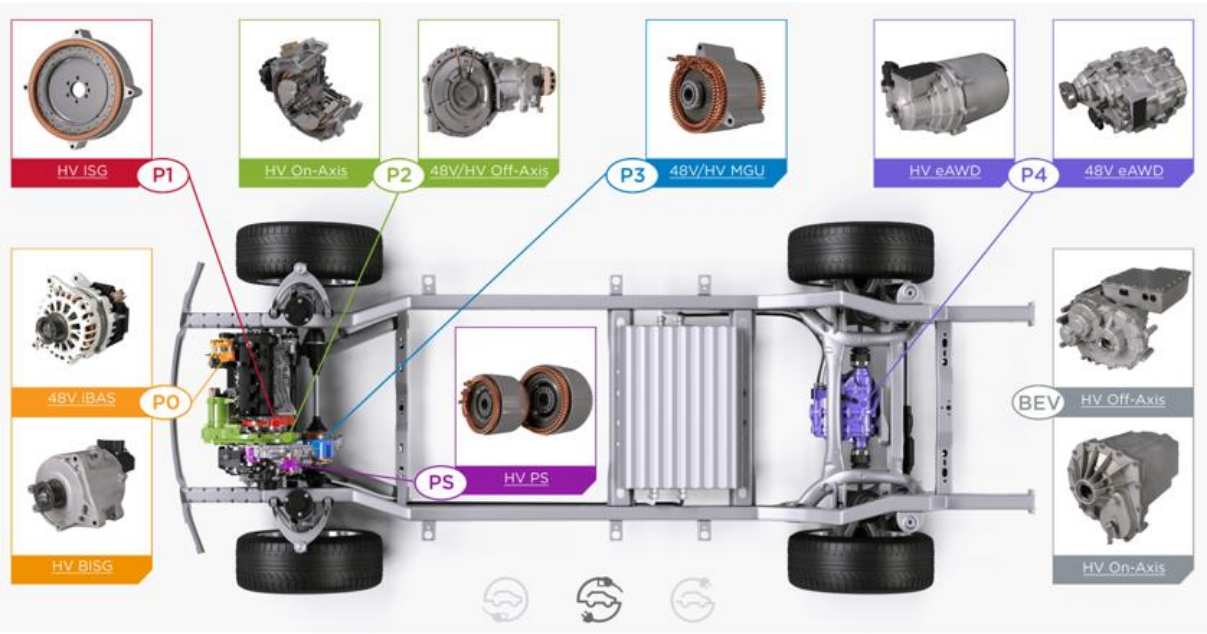

Figure 1.9: Electric Motor Position[6]

2 Energy Management System

In a hybrid electric vehicle, the introduction of one or more power actuators, such as the electric machine(s), introduces an additional degree of freedom that increases the complexity of the powertrain control system compared to a conventional ICE powertrain. In the latter, a low-level controller is sufficient to establish the injected amount of fuel based on the driver request in terms of accelerator and brake pedal position [7].

For an HEV the addition of a high-level controller is necessary, and it must control the power split among the different actuators installed onboard.

This second controller is called Energy Management System (EMS) and it is composed of two parts, as shown in figure 1.1 [2][8]:

- **Supervisory controller**: decides the best operating mode considering the operating conditions of the vehicle and the powertrain.
- **Energy Management System**: decides the power split among the actuators based on the operating conditions of the vehicle and of the powertrain and the information coming from the supervisory controller.

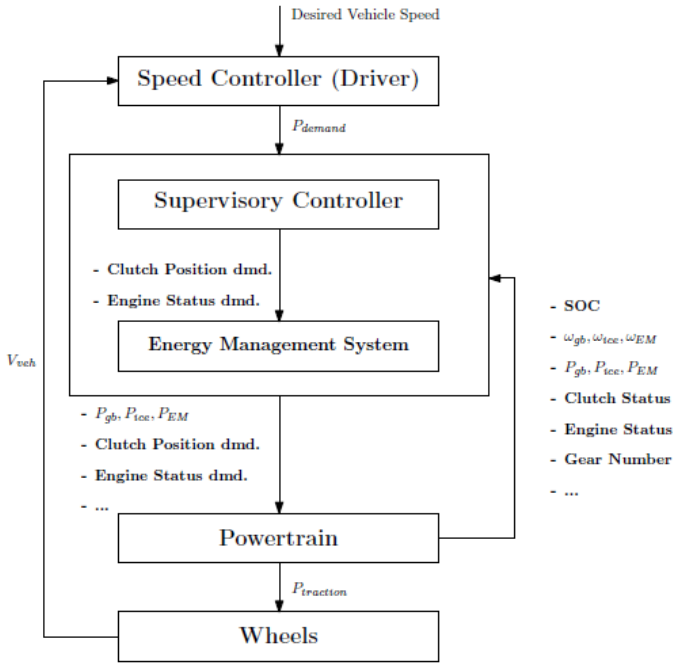

Figure 2.1: EMS structure[2]

The goal of the EMS is to optimize the power split based on some predefined targets to be achieved. The most common optimization strategy is to let the ICE works in its best efficiency regions, to minimize fuel consumption, but it is not the only one. A hybrid power split can be also optimized to minimize pollutant emissions, a strategy that can be relevant for a diesel engine, due to its high NO_x emissions.
Several energy management system control strategies have been proposed in the literature and a first classification can be done splitting them into two categories [2][6].

2.1 Traditional EMS Control Strategies

These kinds of control strategies are based on an already known control algorithm that defines the optimal power split control strategy. It is possible to split them into three categories [2][9]:

- **Global optimization strategies**: the dynamic nature of the system is considered for optimization and an optimal solution is found over a predefined driving cycle, which must be *a-priori known*. It is a global optimization because it considers the entire driving cycle, both in the past and in the future: it makes this optimization strategy unfeasible on a real-time application. The best known is *Dynamic Programming* (DP).
- **Static optimization strategies**: since the global optimization strategies need the full knowledge of the driving cycle, the static optimization strategies idea is based on the knowledge of just short-term time horizon of the past and, if possible, of the future of the driving cycle, to realise a local optimization. The best known is the *Equivalent Consumption Minimization Strategy* (ECMS) that realises a local optimization at each instant by using instantaneous information of the powertrain energy flows. The main idea is that the electric energy can be converted into virtual fuel consumption, to be summed to the effective fuel consumption of the engine, using a factor, called *equivalence factor*. The sum of the two-fuel consumption is the function that has to be locally minimized [10][11].
- **Rule-based control strategies**: based on the introduction of a set of rules that, using as input a set of meaningful parameters, decide the power split. They are not truly optimization strategies, since any function to be optimized is defined, but they are in the form of if-then-else, and efficiency maps or fuzzy-control logic methods may be used in the implementation. The main advantage is their high computational speed, compared to the other methods, and it is possible to analyse a huge number of situations on which a single rule can be used. Many situations are analysed and more accurate is the control strategy. [2][6]

2.2 Dynamic Programming

As above-mentioned, dynamic programming is a global optimization technique, [12][13] one of the optimal control strategies capable of providing the optimal solution to problems of any complexity level. It is based on Bellman's principle of optimality:

"The principle of optimality suggests that an optimal policy can be constructed in piecemeal fashion, first constructing an optimal policy for the "tail subproblem" involving the last stage, then extending the optimal policy involving the last two stages and continuing in this manner until an optimal policy for the entire problem is found." [6][14]

In a hybrid electric vehicle control problem, dynamic programming can be used to find the optimal control strategy on the entire driving cycle, but it is impossible to be implemented in real-time on a true vehicle due to two main limitations.

First of all, the dynamic programming algorithm works in two steps, firstly it solves the backward problems, thus starting from the last time step and proceeding backwards until the first-time step has been reached. This is the reason for the mandatory *a-priori knowledge* of the driving cycle. After the backward operation, the algorithm proceeds in a forward direction-finding the best control trajectory that minimizes the global cost function.

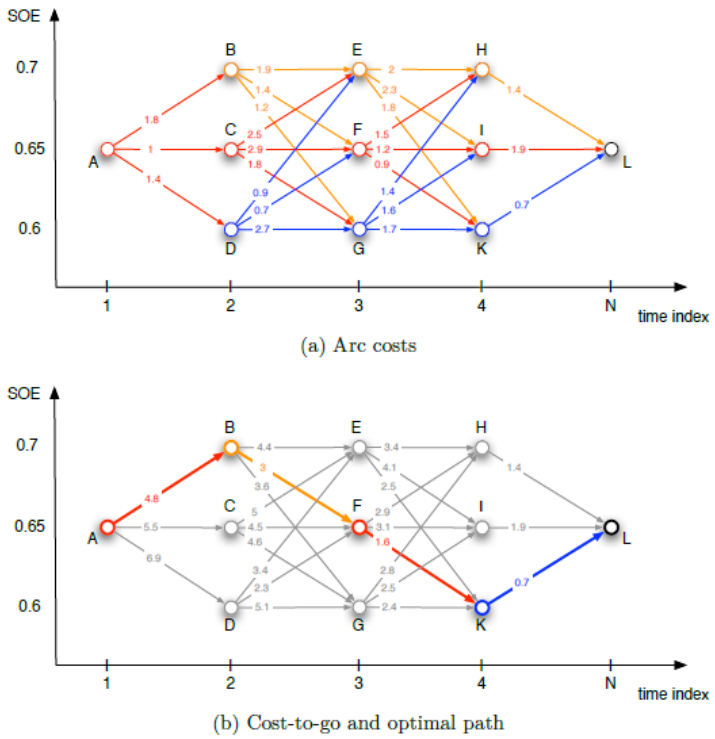

Figure 2.2: Example of DP algorithm procedure[2]

Secondarily, this algorithm is not feasible in a vehicle Electric Control Unit (ECU) due to its high computational burden.

Although these limitations, the DP control algorithm is used as a benchmark for other control strategies to evaluate their performance.

In a hybrid electric vehicle dynamic programming optimization, some *state variables* $x(t)$ and *control variables* $u(t)$ must be defined. The former defines the state of the system, an example is the State of Charge (SOC) of the battery, while the latter is the variable whose values at each time step are defined by the control algorithm (i.e., electric machine power and engine state).

The cost function that has to be minimized is usually defined by Equation 2.1. It is not the only one that can be defined. It allows to minimize the fuel consumption but, for example, one goal of the optimization is to minimize NO_x emissions, introducing a different cost function.

$$J(t, u(t)) = \int_0^T \dot{m}_f(t, u(t))dt \qquad 2.1$$

Where:

- t is the time
- $u(t)$ is the control variable
- $\dot{m}_f$ is the fuel rate

Another important aspect to consider is the discretization of the values that the state and control variables can assume. The optimal solution can be theoretically found only if these variables have a continuous domain, however, it will lead to a huge computationally effort. Thus, a discretization of the domain is required, but a compromise is mandatory, to minimize the computational effort, preserving the performance of the algorithm, being sure to stay as close as possible to the optimal solution [2][13].

2.3 Innovative techniques

The optimal operating points for the engine and the decision of the most appropriate power split value in a hybrid powertrain is not a simple decision and a clear correlation is not easy to find. The rule-based control strategies are based on a set of rules that change depending on the vehicle demands, but it without defining a clear relationship between power split and the variables that characterize the control problem. [15]

The same consideration can be extended to the other control strategies already analysed, based on an algorithm of optimization such as the dynamic programming, or to the proper optimization of a parament, such as the ECMS.

New horizons in terms of modelling approaches have recently been investigated.

Nowadays the artificial intelligence (AI) is one of the most promising approaches in several technological fields and it can also be exploited in the automotive field to proper model and describe very complex phenomena that characterized the automotive field, from the soot emissions prediction, for example, to the design of a high-level control system.

The artificial intelligence models can predict a numeric value or classify a phenomenon among a certain number of classes, without knowing a clear relationship between the variables that characterized the problem and the related predicted value.

The real challenge of these types of models is to effectively train the AI model, because, during the training phase, the model catches the relationship between the variables that characterize the model, called *features*, and the predicted value that, in the training phase in already known.

It is considered a challenge for two main reasons. The training dataset must be wide enough to let the training algorithm reach a high value of accuracy, and, since these models are applied, as already said, to very tricky problems, the features selection is a crucial phase, and it is not easy as it may appear.

For an EMS design application, two models are most promising in terms of results and real-time application:

- **Neural Networks and Deep Neural Networks**: these models are inspired by how our brain exchanges information. As the brain uses electric impulses to pass the information between the neurons, a neural network exchanges information between the *artificial neurons* in terms of number. The artificial neuron applies a mathematical function to the values that receive in input to compute an output value that passes to the next neuron and so on.

  If the architecture of the network is organized with more than three *layers*, which are a collection of neurons operating together at a specific depth, we talk about *deep neural networks* (DNN).

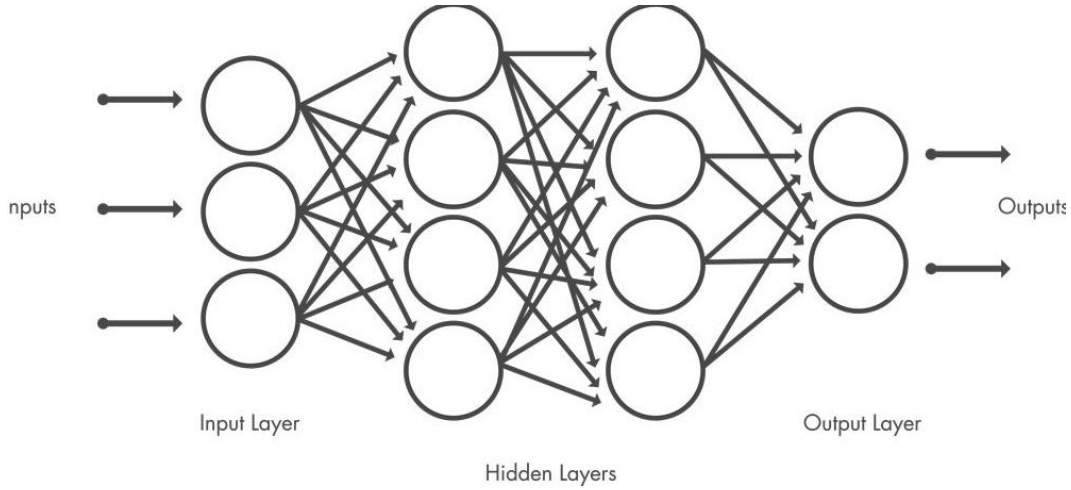

Figure 2.3: Neural Network Architecture[16]

- **Reinforcement Learning** (RL): based on how humans and animals learn, that is a trial-and-error process. In fact, the control problem can be split into two elements, the *agent* and the *environment*. The agent acts on the control variable of the system (i.e., power split) and interacts with the surrounding part of the system, which can be represented by the environment. This learning process does not require a training data set with already known values of the control variables, because it automatically learns through an *award function*. The value of this function is computed after any decision of the agent. The close the solution is to the optimality, the higher the award. Due to its ability to represent a complex non-linear system without any information about the optimal solution, reinforcement learning methods are very suitable for the design of the EMS of a hybrid electric vehicle.

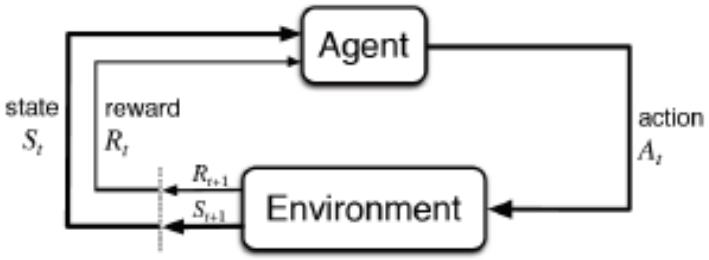

Figure 2.4: Example of agent-environment interaction[17]

- **Deep Reinforcement Learning** (DRL): it is the combination of reinforcement learning and a deep neural network. The reinforcement learning algorithm is used to train the agent, while the deep neural network is exploited to describe the system.

3 AI Theoretical Background

3.1 Introduction to Artificial Intelligence?

Artificial Intelligence (AI) is a wide-ranging branch of computer science concerned with tasks normally performed by human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

This scientific and technological field is quite recent, and the term *artificial intelligence* was coined only in 1956 for the first time by the scientist John McCarty. The first computer program that was able to reproduce a simple human way of thinking was presented at the Dartmouth College workshop in 1956 by Allen Newell and Herbert Simon. It was known as *Logic Theorist,* and it was able to prove logical theorems only using math principles.

After the introduction of the subjects, artificial intelligence had some ups and downs periods because it was seen as a very promising technology, but some ethical and technical problems slowed down the grew of this technology, particularly due to the low computational power of the available computers. From the ethical point of view, the attention was pointed to the danger of creating a machine that was able to think and act without any human control, while from the technological point of view these models required a huge computational effort, due to their complicated algorithm, that, with the computer technology of those years, was practically unfeasible.

Starting from this century, the increase in the computational capabilities, algorithmic improvements and the possibility to access large amounts of data, have arisen the interest in artificial intelligence, which is now exploited in fields not directly linked with computer science: e.g., medical diagnosis, data mining, etc. [18]

The term *deep learning* has dominated the scene in the last years because of increasing attention to the creation of models that can auto-learn directly from data, resuming a mathematical model known as an *artificial neuron*, theorized by Alan Tuning even in 1943, and included in a more complex model known as *neural network*. [19]

Since the general problem of creating intelligent machines is very wide and also the term intelligence is referred to several different human actions, the artificial intelligence field can solve different sub-problems.

It can be used in problem-solving and reasoning models, using algorithms that imitated step-by-step reasoning that humans use in some logical context, or knowledge representation: i.e., the

representation of a different type of information. Another important field is natural language processing, a sub-field that studies how to allow the machines to read and understand human language, as a clear example can be *Alexa* from Amazon or each kind of vocal recognition device. Last but not least, artificial intelligence has been exploited in the learning field, leading to the creation of *machine learning*. This last sub-field of artificial intelligence is the one that is analyzed in this thesis since it is referred to a computer algorithm that improves automatically through experience, let the machine learn directly from the data, requirements that were introduced in the previous chapter when the Energy Management System design innovative technique was introduced.

3.2 Machine Learning

Machine Learning (ML) is a subset of models that are parts of the artificial intelligence group. The main goal of these modelling techniques is to design a model that can find some relationships and learn something from a set of data that it receives as inputs, without any external instruction. The term “learning” is referred to the ability to auto-learn hidden relationships from the input data, while the term “machine” is referred to the presence of a computing model.

However, also the machine learning world is very wide, and it contains several subcategories that are employed very usefully in a large set of applications.

Nowadays machine learning is everywhere around us: from artificial systems that can communicate with humans, understanding and answering to what the user asks to the machine, like Alexa, developed from Amazon or Siri from Apple; it is heavily exploited in the biomedical field to understand, for example, if a TAC image could be dangerous for the patient or not, using machine learning techniques with image classification purpose. Image classification models are also used in the computer vision field (e.g., in autonomous driving vehicles, where the vehicle should correctly recognize a pedestrian crossing the street or the traffic light phase). [20]

Other applications of machine learning are related to the financial field, trying to predict if a stock is going to grow or fall, in the data mining field, for cluster a set of data in a certain number of groups that are linked together by some common features, in the natural language processing, a field that has that aim of process texts and audios trying to extract the most important words or the idea that the text wants to communicate. It is also present in every search engine that, starting from the information gathered from the cookies of each personal computer, send the most attractive advertisements to the user. It is possible to understand that machine learning technology has a huge potential that has been only partly explored yet, due to its novelty.

After this brief introduction to the machine learning world, aimed at introducing the main idea that is behind these models and a list of some application fields, the next section will be analysed the main classification of machine learning models with their principal characteristics.

3.3 Machine Learning Categories

As already presented in the above section, machine learning models are exploited in several fields and, due to the wide difference among each field of applications, also machine learning models are very different from each other both from the algorithm point of view and model results point of view. Machine learning techniques are usually classified into three wide categories, based on how the training data are structured and the feedback available to the learning method. [20][21]

- **Supervised Learning**: data are structured in two sets: input data, which are the input variables to the model, and output data, which are the correct results that the model has to pursue. The goal of this model is to extract a general rule that can associate an output as close as possible to the correct one, to each input.
- **Unsupervised Learning**: data are provided to model only in the form of input data with no correct output associated. These kinds of models, generally, are exploited to find the hidden relationship among the input data, to group the several set with similarities that the model must find. Usually, the unsupervised learning procedure is used coupled to a supervised learning model if the available data do not have enough information about the correct output in classification problems.
- **Reinforcement Learning**: the model can interact with a dynamic environment and has the task to pursue a goal: i.e., a variable that has to be maximized or minimized. These models are based on the concept of rewards. During the training phase, the model generates some control actions that are exploited by the dynamic environment to generate the value of the goal variable.[22] If the value is getting closer to the target the model receives a reward, which means that the model is going in the right direction to find a global minimum to the goal variable. The ability of reinforcement learning models to interact with the dynamic environment makes them very useful in control systems and so also exploitable for energy management system design in hybrid vehicles.

Another classification could be done based on how the output of the model is structured:

- **Classification**: the output can have only discrete values, that are called classes. The machine learning model is a supervised model, that classifies the input values into the correct class.

- **Regression**: in this case, the output is a continuous variable, and the model is a supervised one. An example of a regression problem could be the evaluation of the electric motor torque to be provided in a hybrid electric vehicle.
- **Clustering**: the input provided to the model is divided into a certain number of groups. Clustering models are unsupervised because the groups are not a priori known and all the considerations valid for unsupervised learning models can be extended to clustering models.

3.4 Supervised Learning Algorithms

In the literature, the EMS design problem is usually dealt with reinforcement learning algorithms or supervised learning algorithms, if the artificial intelligence is exploited.
Since in this work supervised learning techniques were adopted for the EMS design problem, this type of model will be described more in detail. In particularly, Long Short-Term Memory (LSTM) Deep Neural Networks (DNN) were employed because they can remember some temporal relationship between a time series input data.
A supervised learning model was chosen since the Dynamic Programming (described in section 2.2) can be used for generating a set of optimal results: i.e., the outputs for training the neural networks, consisting of the optimal solution on different driving cycles. The procedure to create the model is described in section 4.6
As above-mentioned, a supervised learning algorithm takes a known set of input data and known responses to the data, training a model to generate reasonable predictions for the responses to new input data. Several supervised learning algorithms can be applied with a trial and error procedure to find the most suitable one. [20][21] The selection of the most appropriate algorithm is a process of trial and error and a trade-off between the specific characteristic of the algorithms, speed, memory allocation and accuracy.
In the following section, the classification and regression algorithms initially considered for this study are briefly described.

3.4.1 Logistic Regression

This model can be used only for binary classification problems with two classes. It works fitting a model that can predict the probability that a response belongs to one class or the other. This model is very simple both to be interpreted and to be trained, however, it has some limits: it only works with binary classification problems and with data that can be quite easily divided into two main classes. [20][21].

Figure 3.1: Example of logistic regression application[23]

3.4.2 K-Nearest Neighbour

k-Nearest Neighbour model (kNN)[24] begins by randomly choosing k (number of classes) centres into an n-dimensional space (number of features). It assigns each point to the nearest centre, recomputing them as the centre of mass for all points assigned to it. This process is repeated until the process converges to a stable value of the computed centres.

KNN algorithm categorizes objects based on the classes of their nearest neighbours in the dataset, assuming that objects near each other are similar [20][21].

A parameter that can be tuned on this model is the distance calculation by means the n-closest objects are found:

- **Euclidean**: $d_E = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$
- **Cosine**: $\cos(\theta) = \frac{\sum_{k=1}^{n} p_k q_k}{\sqrt{\sum_{k=1}^{n} p_k^2}\sqrt{\sum_{k=1}^{n} q_k^2}}$
- **Chebyshev**: $d_C(p,q) = \max_k\{|p_k - q_k|\}$

This model is useful to easily establish benchmark learning rules, but with the drawbacks of high prediction speed and high memory usage.

Figure 3.2: Example of kNN model application[23]

3.4.3 Support Vector Machine

In Support Vector Machine (SVM),[25] a linear decision boundary, called hyperplane, Separating the points of the different classes.

The hyperplane has a linear formulation as shown in Equation 3.1:

$$\sum_{k=1}^{n} w_k x_k + b = 0 \qquad 3.1$$

The hyperplane is fitted into an n-dimensional space, where n is the number of features of the dataset. The hyperplane with the largest margin between the two classes is selected. [20][21]

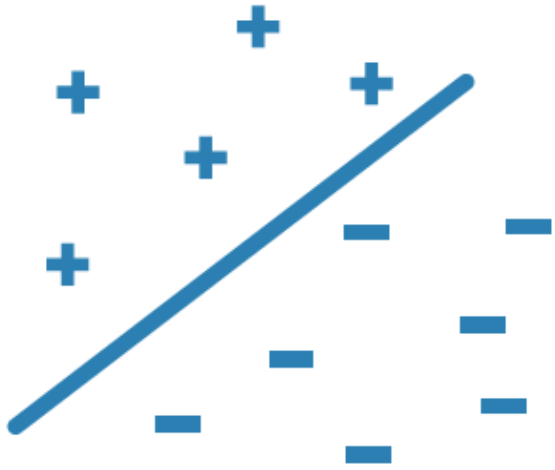

Figure 3.3: Example of SVM application[23]

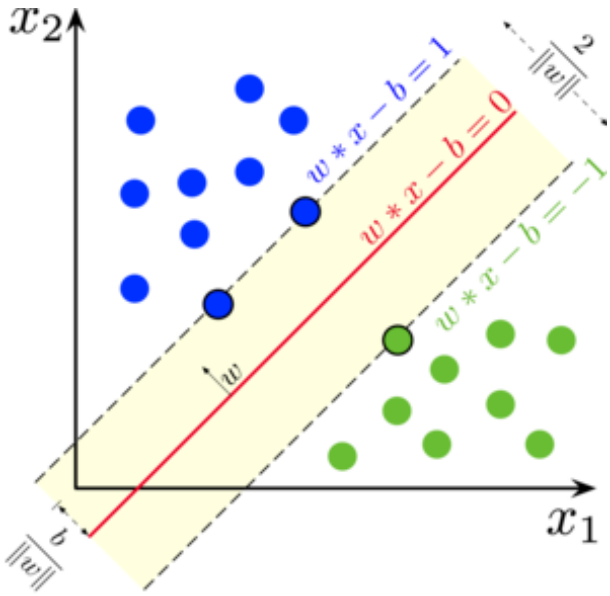

Figure 3.4: hyperplane with highest margin selection[26]

One of the major issues of this model is that is not highly accurate with non linearly separable data. In this case, two main solutions are used.

A loss function, as indicated in Equation 3.2, can be used to penalize the points on the wrong side of the hyperplane:

$$\max\left(0,1 - y_k\left(\sum_{k=1}^{n} w_k x_k - b\right)\right) \tag{3.2}$$

Where:

- y_k is the kth target.
- $\sum_{k=1}^{n} w_k x_k - b$ is the kth output from the traditional hyperplane computation.

If the data is on the right side the function will return 0, while in the other case a value that is proportional to the distance from the margin set by the hyperplane.

The goal of the optimization is to minimize the loss function, in Equation 3.3:

$$\left[\frac{1}{n}\sum_{k=1}^{n} \max\left(0,1 - y_k\left(\sum_{k=1}^{n} w_k x_k - b\right)\right)\right] + \lambda \sum_{k=1}^{n} \sqrt{w_k^2} \tag{3.3}$$

The parameter λ determines the trade-off between increasing the margin size and ensuring that point $\boldsymbol{x_i}$ lies on the correct side of the margin.

The other solution is to use a kernel transformation to transform nonlinearly separable data into higher dimensions where a linear decision boundary can be found. In other words, a change of reference frame is used to describe the data and, into the new reference frame, they can be described as linearly separable.

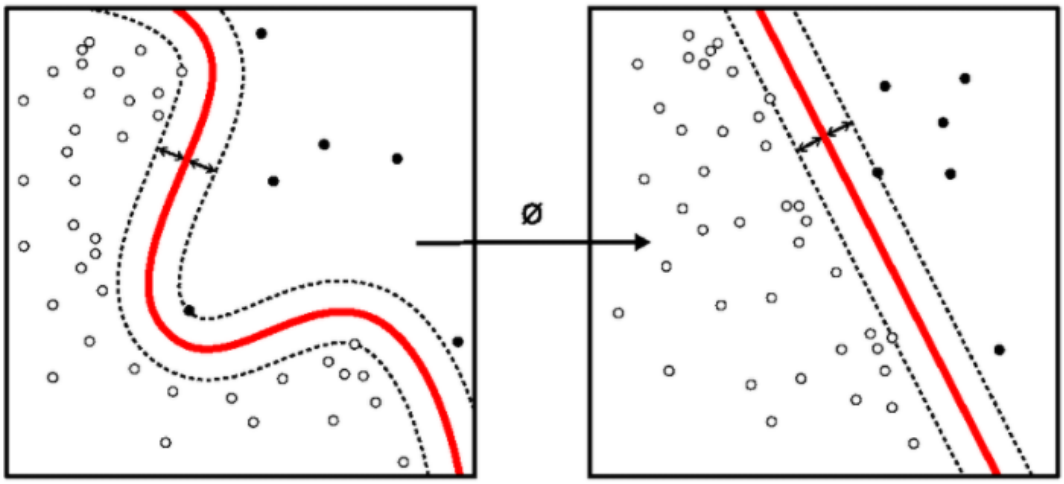

Figure 3.5: SVM Kernel application. Data are transformed and described into the new reference frame, through a transform function ϕ[26]

Another limitation of this model is that, since a single hyperplane is used to separate the data, the classic SVM algorithm can be exploited only for a binary classification problem. However also this limit can be overcome by fitting more than one hyperplane, allowing multiclass classification problems, with a technique called error-correcting output codes.

3.4.4 Naïve Bayes

A Naïve Bayes classifier[27] uses a probabilistic approach to classify data, based on the application of Bayes' theorem, stated mathematically by Equation 3.4:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.4}$$

Where A and B are two events and $P(B) \neq 0$. It allows to compute the conditional probability, that is the probability of event A, for example, occurring given the event B is appended and can be flagged as true in a logic perspective [20].

A Naïve Bayes classifier is based on a fundamental assumption: the presence of a particular feature in a class is unrelated to the presence of any other features. In other words, the presence of a single feature in a class is not related to the presence of any other feature in the same class.
Assumed this statement, the classifier classifies new data based on the highest probability of its belonging to a particular class, computed the probability distribution of each class in the features space during the training phase.
A typical application of the classifier is in many financial and medical fields since it is very powerful when the model encounters scenarios different from the training data ones. [21]

Figure 3.6: Example of Naive Bayes classifier application[23]

3.4.5 Discriminant Analysis

This classifier is based on the Gaussian distribution concept. It assumes that different classes generate data based on Gaussian distributions.
The discriminant analysis classifies data by finding linear combinations of features and using boundaries between each class, established through every Gaussian distribution of each class.
The goal of the training phase of this model is to find the parameters to fit a Gaussian distribution for each class to compute the boundaries, which can be linear or quadratic functions.
Based on which part of the features space a new data is located and inside of which boundary, a new data is classified.
This model is very fast to make its predictions, while it involves a large amount of memory during the training phase [20][21].

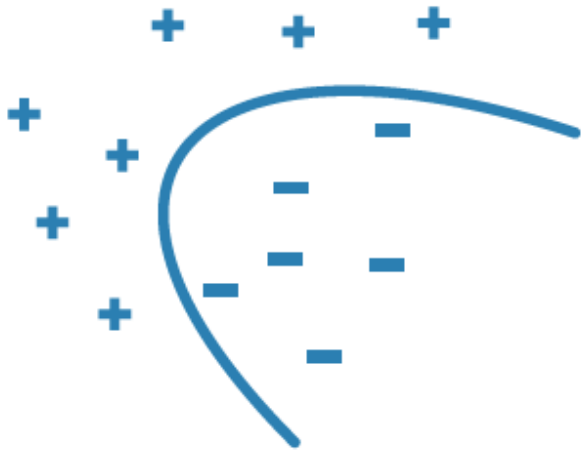

Figure 3.7: Example of discriminant analysis application[23]

3.4.6 Decision Trees and Regression Algorithms

Another common classifier that is used in many applications is the so-called decision tree.
A tree consists of a set of branches that link different conditions, as a flow of if-then-else conditions, whose parameters are set during the training phase. This classifier is quite weak, and the prediction accuracy should not be a crucial requirement. To boost up the performance of a tree classifier, it is possible to combine several weaker decision trees into a stronger ensemble, called bagged and boosted decision trees.

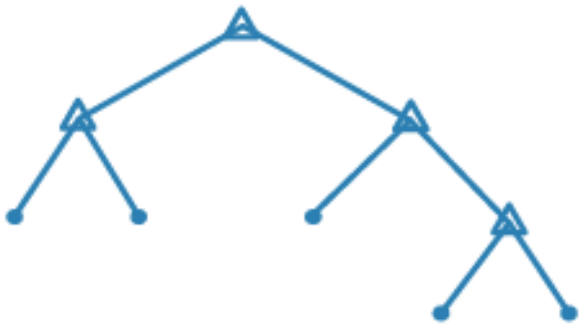

Figure 3.8: Example of decision tree[23]

The most common machine learning algorithm for classification purposes has been briefly described (a separate section is dedicated to neural networks). The following algorithm is used for regression problems, in which the concept of classes cannot be used anymore, since the value that has to be predicted is continuous and not discrete anymore [21].

3.4.7 Linear Regression

The linear regression can be compared to the logistic regression for classification algorithms. It is the easiest regression model that can be used, and it only fits data with a linear behaviour.
Linear regression is a statistical modelling technique used to describe a continuous response variable as a linear function of one or more predictor variables [20][21].
Since the fitted function is linear, the response is:

$$y = a_0 + \sum_{k=1}^{n} a_k x_k \qquad 3.5$$

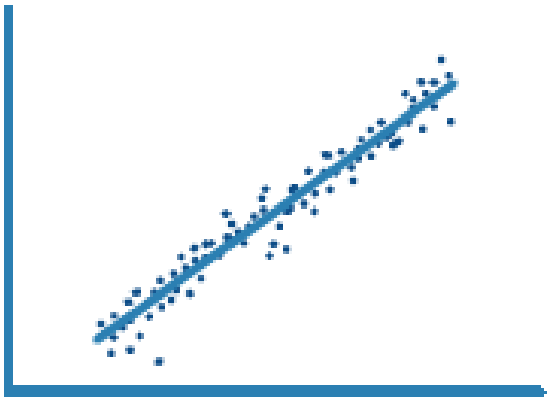

Figure 3.9: Example of linear regression application[23]

3.4.8 Nonlinear Regression

A nonlinear regression technique is based on the same idea of the linear regression one, that is to find an equation that can fit inside the dataset, with the lowest distance between each data point and the regression function.
Nevertheless, a simple linear function is not complex enough to be able to fit a strongly nonlinear data set. In this case, a nonlinear regression function should be used. Generally, these models are assumed to be parametric, where the model is described as a nonlinear equation. The nonlinear term is referred not to the equation itself and so to the variables (features), but to the parameter since the equation is assumed to be nonlinear, while the parameter is not.
To better clarify this aspect, the equation $y = a_0 + b_1 x^2$ is assumed to be a linear function of the fitting parameters, while the equation $y = a_0 x^{a_1}$ is nonlinear of the fitting parameters and can be considered as a nonlinear regression model [20][21].

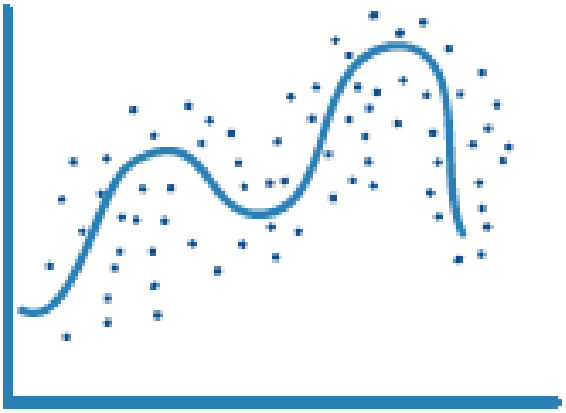

Figure 3.10: Example of nonlinear regression model application[23]

3.4.9 Gaussian Process Regression Model

Gaussian Process Regression Model can be seen as an infinite-dimensional (continuous value as in regression problems) generalization of multivariate normal distribution. It is a stochastic process, such that every finite collection of variables has a multivariate normal distribution (Gaussian distribution), and also every finite linear combination of them is normally distributed. The distribution of a Gaussian process is the joint distribution of all variables, obtaining a distribution over functions with a continuous domain.

Gaussian process regression models are nonparametric models since no equations are used to predict the responses from variables. It is widely used for interpolating spatial data [20][21].

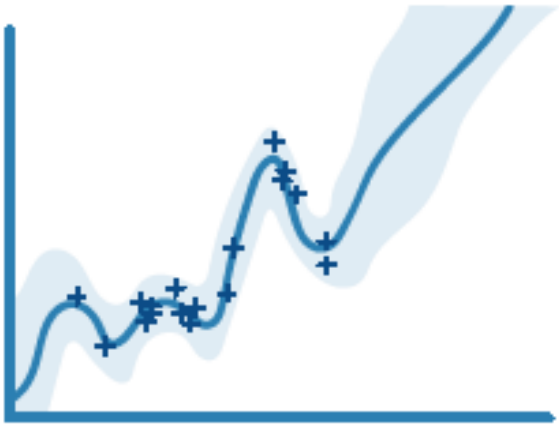

Figure 3.11: Example of GPR model application[23]

3.4.10 SVM Regression

It is the extension of the Support Vector Machine, used for classification tasks, adapted to predict continuous responses. The main difference of the SVM regression if compared to the SVM described

in section 3.4.3 is that, in this case, instead of finding a hyperplane (or more hyperplanes for multiclass problems) that separates data, these algorithms find a model that deviates from the training data by a small value, no greater than a small amount, to make possible the prediction of continuous responses. SVM regression model fits very well with high dimensional data with a great number of features that describe the dataset [20][21].

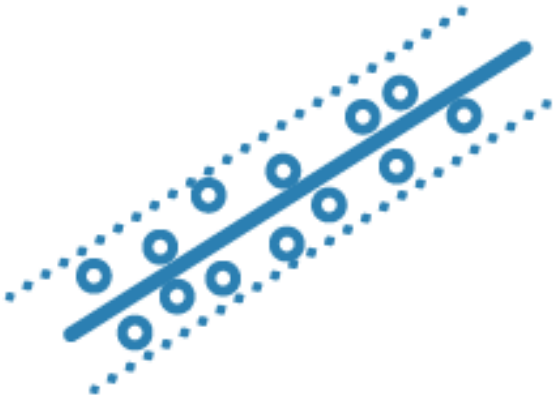

Figure 3.12: Example of SVM regression application[23]

3.4.11 Generalized Linear Model

A generalized linear model is part of the nonlinear model family, but the difference is that it uses linear methods. In fact, during the training phase, it fits a linear combination of the input to a nonlinear function of the outputs. These algorithms are used when the response variable have nonnormal distributions [20][21].

The two functions that are fitted can be represented by Equations 3.6 and 3.7:

$$y_L = a_0 + \sum_{k=1}^{n} a_k x_k \tag{3.6}$$

That is the linear combination of the input (variables of the model) and it is called *linear predictor*.

$$E(y) = g^{-1}(y_L) \tag{3.7}$$

is called *the expected value*, the response to be predicted by the model, while g is the so-called *link function*, which is the nonlinear function applied to the output.

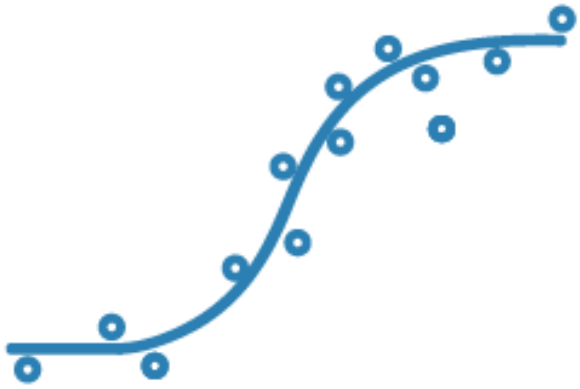

Figure 3.13: Example of generalized linear model application[23]

3.4.12 Regression Tree

These models work exactly as the decision trees, already described in section 3.4.6, modified to be exploited in regression problems [21]

Also, the features of this model are comparable to decision trees ones.

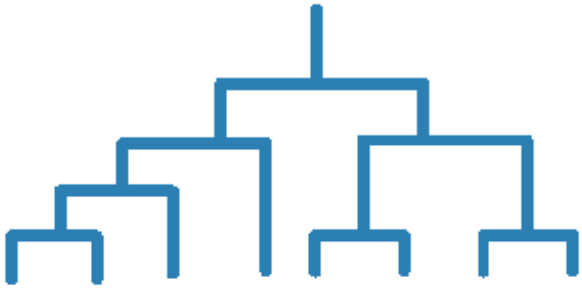

Figure 3.14: Example of regression tree application[23]

3.5 Neural Networks

The last machine learning model that is presented in this thesis work is, concerned with the application on EMS design of a (p)HEV, the most interesting and powerful, and it is called Artificial Neural Networks (ANNs), usually simply called Neural Networks (NNs).

This type of model tries to emulate the operating principle of a brain to copy the human way of thinking. This necessity was born since computers, with traditional programming techniques and also simplified machine learning models, don't easily solve problems that are easily solvable by a human brain, such as some problems of speech recognition or images recognition. To solve these hard computational problems, a new branch of artificial intelligence and machine learning fields has been born, called *deep learning*. Deep Learning models exploit the concept of artificial neural networks,

expanding their complexity using bigger architecture and deeper networks, as we will see later in this section.

Since deep learning models are very computationally demanding, the potentialities of these models have been started to be exploited only in the last decade, thanks to the increasing computational power of computers, to the possibility of using cluster calculation and cloud computing.[19][20]

As already stated, NNs try to mimic the principle of functioning of our brain. The brain works thanks to some structures called neurons, which are linked together and can exchange information with each other utilizing electrical impulses. In the same way, an artificial neuron, also called perceptron, is linked to the other artificial neurons that compose the artificial networks and can exchange information through numbers.

Conceptually speaking, the only difference between biological and artificial neural networks is based on the kind of information exchanged between neurons, electric impulse in the case of biological one, and numbers in the case of the artificial one. Nevertheless, these models are too simplistic to perfectly mimic a biological neural network, but it is just a mathematical model inspired by its operating principles.[19][20]

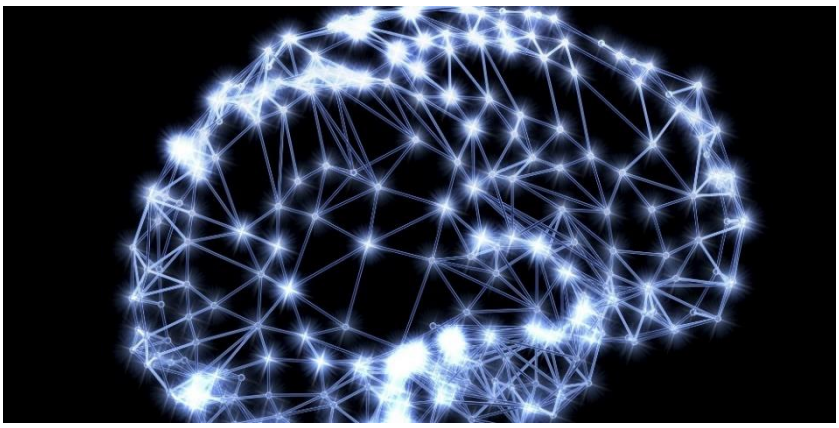

Figure 3.15: Picture of biological neural network

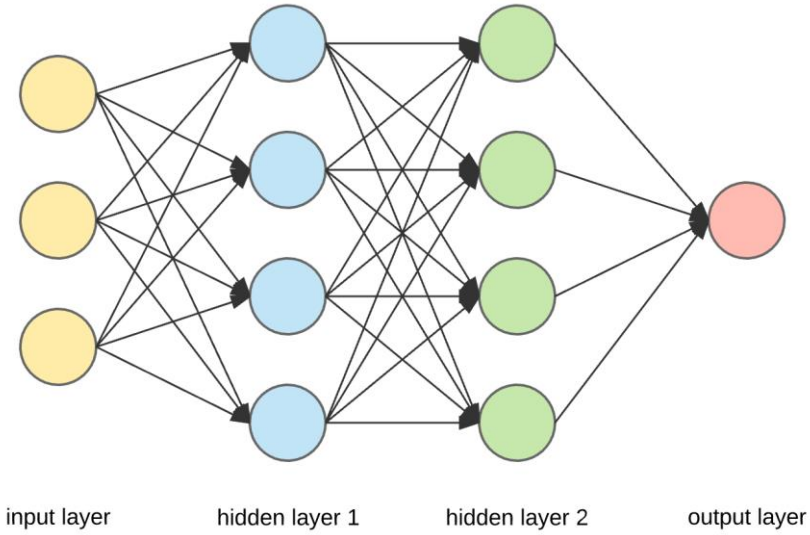

Figure 3.16: Example of artificial neural network

To effectively understand the relationship between the biological and artificial neurons, first of all, it is important to understand how the biological neuron operates. The main goal of this disquisition is not to deeply analyse the structure of a biological neuron, but the mechanism that regulates the flow of information in it.

In a single biological neuron, there are three main subparts:

- **Dendrites**: it is the input channel of the neuron, receiving the flow of information coming from the other neurons of the network.
- **Soma**: also called the cell body, is the central component of the neuron. Its goal is to sums the inputs that the neuron receives around the axon hillock. This sum operation is weighted by the synaptic strength, that is the connection between the neurons in the network. Stronger is this connection and higher will be the weight that multiplies the related input. The weighted sum is later compared to a threshold. If it exceeds this threshold, a so-called action potential is triggered and the neuron can be considered as active.
- **Axon**: the output of the neuron. The action potential travels along the axon and, employing synaptic connection, serve as inputs to other neurons of the network.[28]

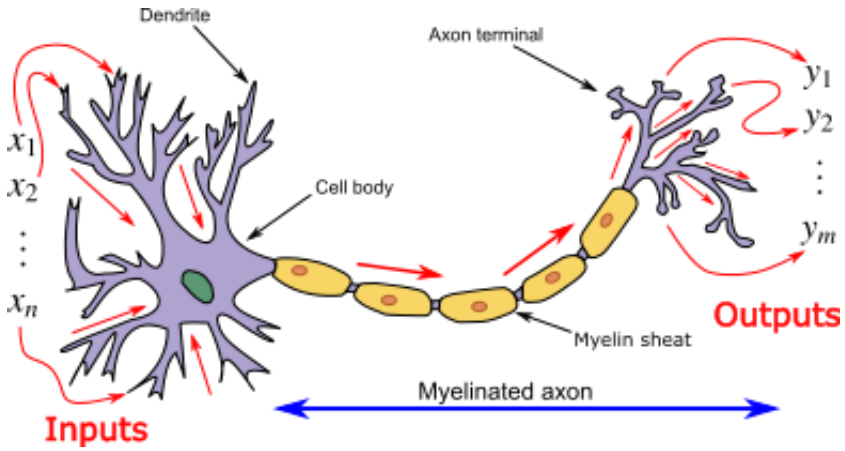

Figure 3.17: Biological neuron model

The artificial network tries to replicate the behaviour of biological neuron and it is composed of an input, an output, a cell that realize a mathematical operation and, finally, an output that is passed to the next neurons.

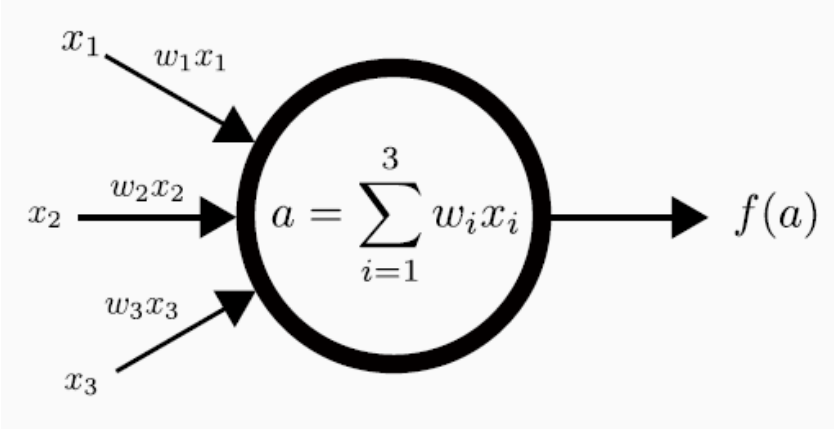

Figure 3.18: Artificial neuron model

The operation that is realized inside the cell of the neuron, or *perceptron*, is the following:

$$a = \bar{x} \cdot \bar{W} + b = \sum_{k=1}^{n} W_i x_i + b \qquad 3.8$$

Where:

- $\bar{x} \in \mathbb{R}^N$ is the *input vector*, that collects the output coming from the other N linked neuron.

- $\bar{W} \in \mathbb{R}^N$ is the *weight vector*, represent the synaptic strengths and determine how strongly a certain input affects the output.
- $b \in \mathbb{R}$ is the *bias* of the *pre-activation function*, a.

This first part of the model represents what happens in the cell of the biological neuron, the next step is to model the action potential and the activation of the neuron. This is represented by the *activation function* $f(\cdot)$ that operates on the pre-activation function, giving the output from the related neuron.

$$y = f(a) = f\left(\sum_{k=1}^{n} w_i x_i + b\right) \tag{3.9}$$

The goal of the activation function is to determine if a neuron should return a value higher, lower or equal than 0, as the threshold in the biological version. Several activation functions have been proposed in the literature and, each of them, characterizes the neuron, making the neuron more appropriate for a specific application.[19]

The more common functions along with their representation on the x-y plot are listed below:

- Hyperbolic tangent function: $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$ 3.10

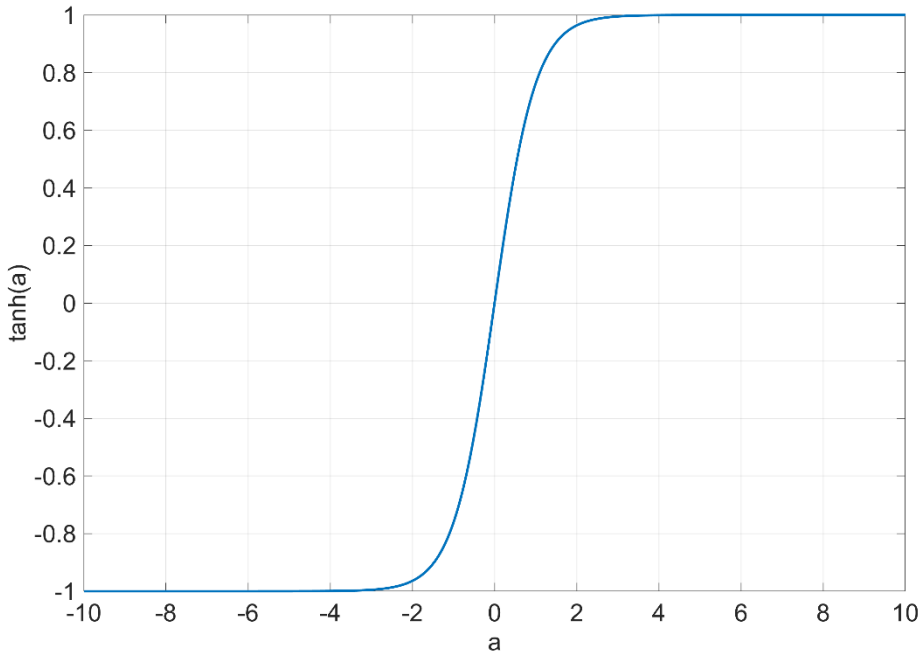

Figure 3.19: hyperbolic tangent function

- Softsign function: $\text{softsign}(a) = \frac{a}{1+|a|}$ 3.11

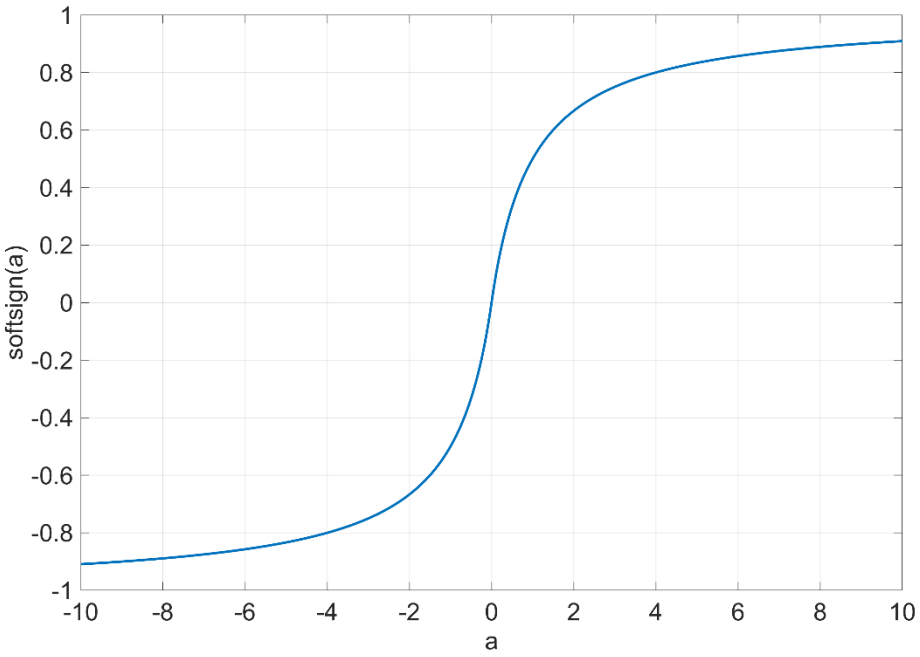

Figure 3.20: Comparison between softsign and hyperbolic function

- Sigmoid function: $\sigma(a) = \frac{1}{1+e^{-a}}$ 3.12

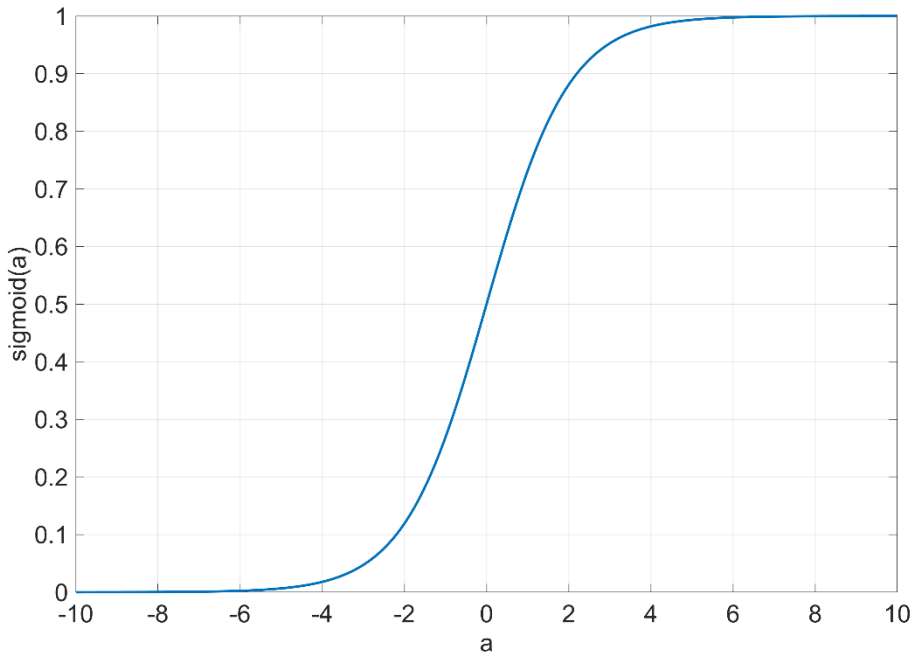

Figure 3.21: Sigmoid function

- Hard-sigmoid function: $\sigma(a) = f(a) = \begin{cases} 0 & a < -2.5 \\ 0.2a + 0.5 \text{ if } -2.5 \leq a \leq 2.5 \\ 1 & a > 2.5 \end{cases}$ 3.13

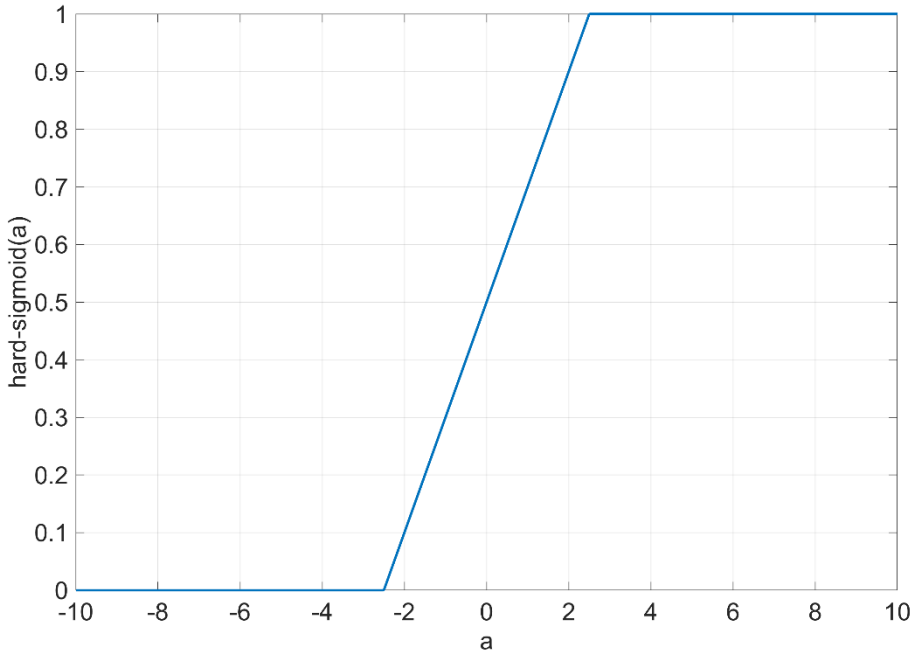

Figure 3.22: Comparison between hard-sigmoid function and sigmoid function

- Threshold function: $f(a) = \begin{cases} k, & a > 0 \\ 0, & x \leq 0 \end{cases}$ 3.14

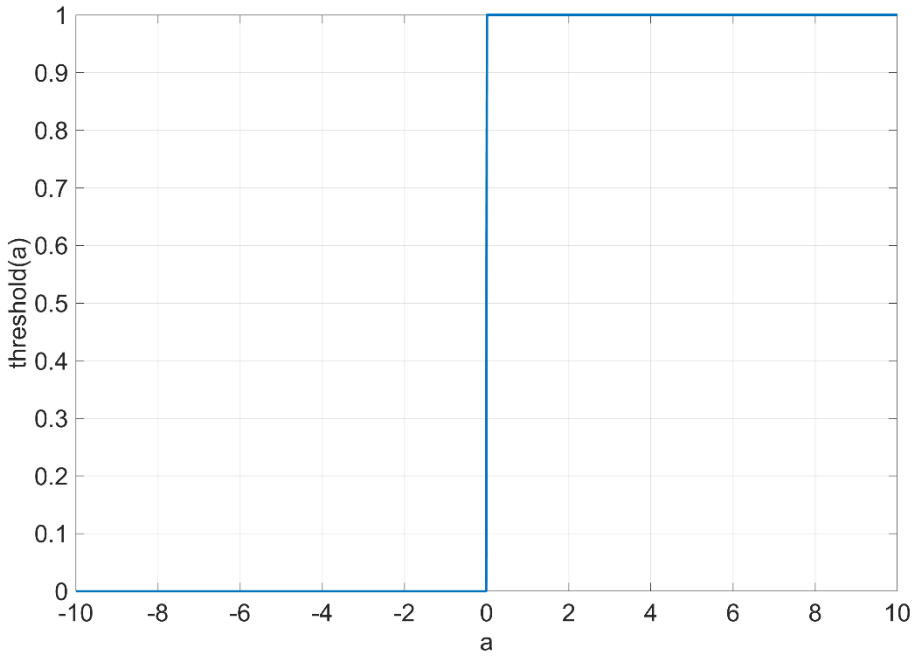

Figure 3.23: Threshold function

However, the weights and bias values must be properly chosen. The neural network model can automatically learn and set the best values for the weight and bias of each neuron in the network through the training phase. Therefore, this type of model can autonomously learn very complex relationships between the inputs of the network and the output of the network itself without any definition from the user. However, the data pre-processing is still at the expense of the designer: it is the trickiest part since it is concerned with selecting the variables, algorithm and parameters of the training phase.[29]

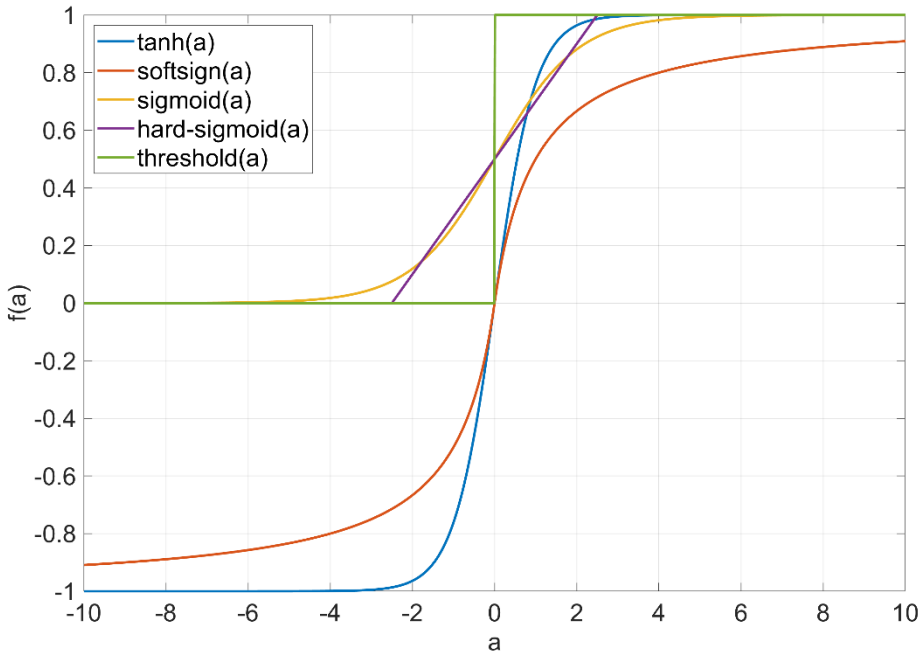

Figure 3.24: Comparison among the above-described activation function

3.5.1 Neural Network Architecture

An artificial neural network is composed of many neurons linked together in a pre-determined way. The “shape” of the neural network and the connections between each neuron establish the neural network architecture.

A layer is composed of a set of neurons that are not linked together, but they can be connected only with other neurons of the previous and the next layer, as shown with the yellow square in Figure 3.25.

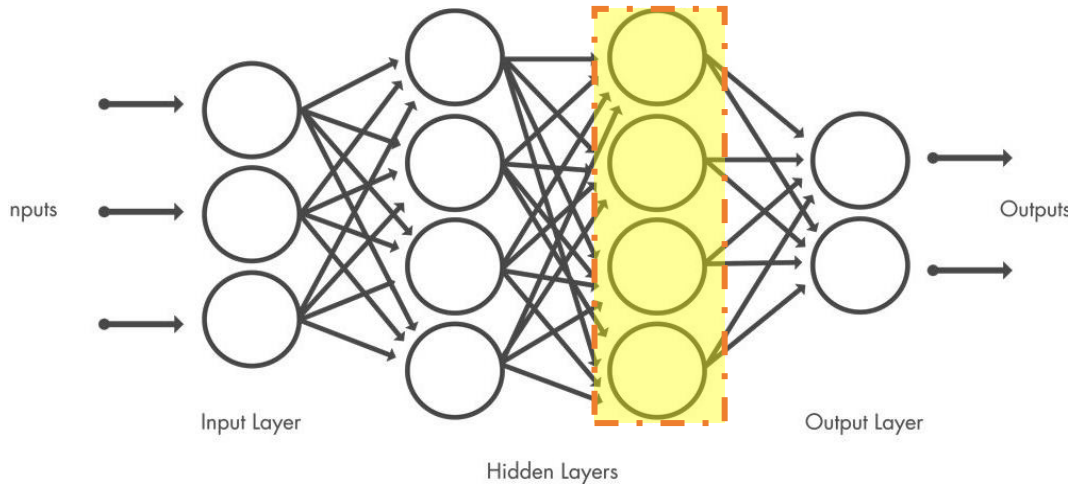

Figure 3.25: Layer representation in the yellow box and layer categories

The layers can be divided into three main categories, based on the function inside the network:

- **Input layer**: the main function of the input layer is to prepare the input data to be processed by the next layers of the network. Its presence is not mandatory, but it is recommended especially in presence of complex networks, with which a step of data pre-processing is mandatory. The input network has to accomplish this task, which can space from data normalization to data mini-batch assembly, as we can see in section 3.5.3.
- **Hidden layer**: all the mathematical computations described in the above section are realized in the neurons of the hidden layer. They can be more than one and they establish the deep of the network, the higher the number of the hidden layers the deeper the network. The concept of a deep neural network is referred to the network that has more than one hidden layer. Of course, increasing the number of hidden layers increases the complexity of the network with all the consequences that will be analysed later.
- **Output layer:** the only task of the output layer is to collect the information coming from the previously hidden layers and process them to be presented as the user expects.

3.5.2 Supervised Neural Network Training Algorithm

Artificial Neural Networks can be applied to solve any problem in the machine learning field. They can solve unsupervised learning problems, with the so-called *self-organizing map*, supervised learning and reinforcement learning problems too.

In this dissertation, supervised learning neural networks will be analysed more in detail because they are utilised in this work.

In this section, the main training algorithm will be described in detail, trying to underline the strengths and the weaknesses of this crucial phase and which are the limits of the learning phase.

The algorithm used to train a network is called *backpropagation algorithm* used in combination with an optimization method that depends on the type of network to be trained.[19] The most common optimization method is called *the Gradient Descent Method* (GDM). In the case of complex networks, e.g., deep neural networks, it requires a too demanding computation effort, therefore a different algorithm is used, called S*tochastic Gradient Descent Method* (SGDM), which works in the same way as GDM algorithm but is less computationally expensive, because it uses only a part of the training dataset, called mini-batch, to computes the derivatives. Other optimization methods used to train a deep neural network are *Adaptive Moment Estimation Method* (Adam) *Method* and *Root Mean Square Propagation Method* (RMSProp), which, compared to traditional SGDM can increase the performance of the training phase, at the expense of a higher computational effort.[30]

The goal of the training phase is to iteratively update the weight vector and bias vector, collected together in the *parameter vector* θ, trying to minimize a predefined error function $E(\theta)$.

In MATLAB environment application the error function, also called *loss function* is the *cross-entropy loss function* for classification tasks and *mean-square-error* for regressive tasks.

The former function can be used only for classification problems because its formulation let to underline the goodness of the assignment of input data to the right class. The goal of classification (deep) neural networks is to compute the probability that the single input data can be classified in each of the available classes. For example, if the classification problem needs to classify data into 9 classes, the neural network returns 9 values of probability that the input data belongs to each of the 9 classes. This operation, in a deep learning architecture, is computed by the *softmax layer*, as it is deeply analysed later in this section, while the correct class, which is the class with the higher probability value, is selected by the output layer, that has just to output the class with the highest probability.[31]

The cross-entropy loss function, previously mentioned is expressed in Equation 3.15 and evaluates the accuracy of the classification:

$$E(\theta) = loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} w_i t_{ni} \ln y_{ni} \qquad 3.15$$

Where:

- K is the number of mutually exclusive classes.
- N is the number of samples, which are the number of input/output training data that the user uses to train the network.

- t_{ni} is an indicator that the nth sample belongs to the ith class, and it is necessary to select the right probability value from the probability vector coming from the softmax layer. This indicator assumes the value of 1 only if, during the sum operation, the function wants to evaluate the probability that the nth sample belongs to the ith class. In all the other cases the value of t_{ni} is 0.
- y_{ni} is the output for sample n for class i from the softmax layer and, as already said in this section, compute the probability that the network associates the nth input with class i.
- w_i is the weight for class i from the softmax layer to the output layer.[32]

During the training phase, this function must be minimized.

The mean-square-error loss function is used for regressive tasks and it is expressed in Equation 3.16.

$$E(\theta) = loss = \frac{1}{N}\sum_{n=1}^{N}(y_n - \widehat{y_n})^2 \tag{3.16}$$

y_n is the output computed by the network while $\widehat{y_n}$ is the correct value provided in the training dataset. Since absolute values are compared in this function, the performance of networks with different scales cannot be compared using this error, therefore a normalized version is preferred, as shown in Equation 3.17.

$$E(\theta) = loss = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{y_n - \widehat{y_n}}{\widehat{y_n}}\right)^2 \tag{3.17}$$

The loss function can be minimized thanks to the optimization algorithm, which updated the biases and the weights of each neuron to converge to the minimum. The three most important algorithms will be described more in detail.

3.5.3 Gradient Descent Method (GDM) and Stochastic Gradient Descent Method (SGDM)

GDM algorithm can only be applied to a very simple neural network, due to its high computational effort. However, it is the reference algorithm from which all the other training algorithms are derived. The basic principle is based on finding the direction in $\mathbb{R}^N$ space, where N is the parameter vector θ dimension, that minimizes the loss function.

The parameters are firstly initialized with small values. By indicating with $g_k(\boldsymbol{x})$ the operation performed in a single neuron, the output of the neural network can be writing as $y(\boldsymbol{x}) = f(\sum_k w_k g_k(\boldsymbol{x}))$.

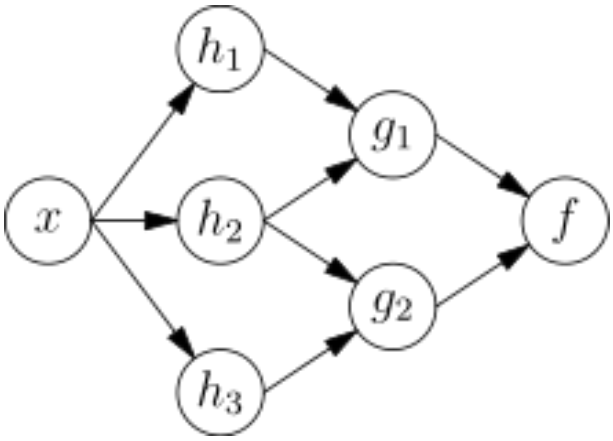

Figure 3.26: Example of a feed-forward neural network

The error function $E(\theta)$ can be obtained by comparing the optimal value y and the output value of the neural network.

After that, the direction minimizing the error function can be found by computing the gradient $-\nabla E(\theta)$, and the parameter vector can be updated.

$$\theta_{l+1} = \theta - \alpha \nabla E(\theta_l) \qquad 3.18$$

θ_{l+1} is the updated parameter vector, while the parameter $\alpha > 0$ is called $learning\ rate$ and is the step done to update the parameters along the minimization direction, as shown in Figure 3.27. The α is one of the hyperparameter of the model, i.e., the parameters that are not automatically learned by the network but must be set by the user or optimized using an optimization method, such as the Bayesian optimization.[19][29] [30]

The smaller the step, the higher the possibility of finding the minimum of the error function, but the training phase should last longer to effectively reach the minimum. The higher the value, the shorter the duration of the training phase but a higher probability to skip the minimum and obtain a parameters vector far from its optimum value. Setting the learning rate value is a trade-off operation between accuracy and computational time of the training phase.

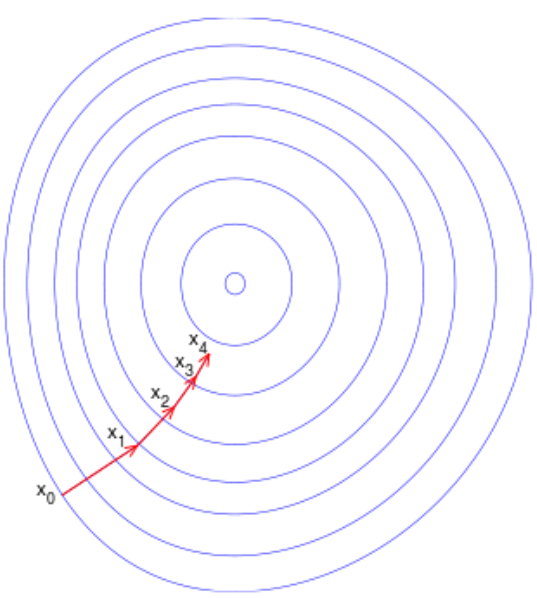

Figure 3.27: GDM illustration in terms of direction towards the minimum[33]

The "backpropagation" name derives from the methodology: after a forward step, consisting of computing the output value y with fixed parameters θ_l, there is a backward step because the parameters update begins from the output of the network at θ_{l+1} values and streams back to the first layer of the network to recompute the output y.

In the case of a deep neural network, since SGM are excessively computational demanding, SGDM are usually employed. The only difference between the two techniques is on the data used to compute the gradient, which is a subset of the entire dataset, called *mini-batch*. This is another hyperparameter that should be optimized.

However, this method can oscillate along the path of the steepest descent towards the optimum. To reduce this, it is possible to add a momentum term that is a movement similar to a rotation, obtaining the *stochastic gradient descent method with momentum* γ, as shown in Equation 3.19.[30]

$$\theta_{l+1} = \theta - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \qquad 3.19$$

With SGDM some problems can lead to stop the training process or to do not reach the convergence towards the minimum. The most common ones are the vanishing gradient problem and the exploding gradient problem, described in section 3.5.4.

3.5.4 Vanishing Gradient Problem and Exploding Gradient Problem

With the stochastic gradient descent method, each weight of the network is updated by a value proportional to the partial derivate of the error function with respect to the current weight in each iteration of training, as stated by Equation 3.18.

Since the typical activation functions, such as tanh or sigmoid functions, can range between (0, 1), the consequence is that also their gradient could stay in the same range. If we consider a deep neural network architecture, the gradient of the error function could vanish because it can assume very small values, avoiding the parameters update and stopping all the training phases. This is a typical characteristic of the deep neural network because, with backpropagation algorithm coupled with SGDM, the gradient of the error function is computed by the chain rule since neural network model can be seen as a function composition of n function, with n that is the number of neurons that composed the network. Since the chain rule computes the partial derivative multiplying n functions that can have values in the range (0, 1), it is easy to understand that this product tends to 0 increasing the number of neurons.[19][20]

The same problem, but with the opposite consequence is when the gradient of the error function tends to infinite values, depending again on the activation functions that are used in the network. In this case, the problem is called *exploding gradient problem*.

Several solutions to these issues have been proposed in the literature. One of the most effective, that can be easily implemented in MATLAB environment is to use some optimization algorithms that do not update the parameters proportional to the gradient magnitude, such as RMSProp and Adam.

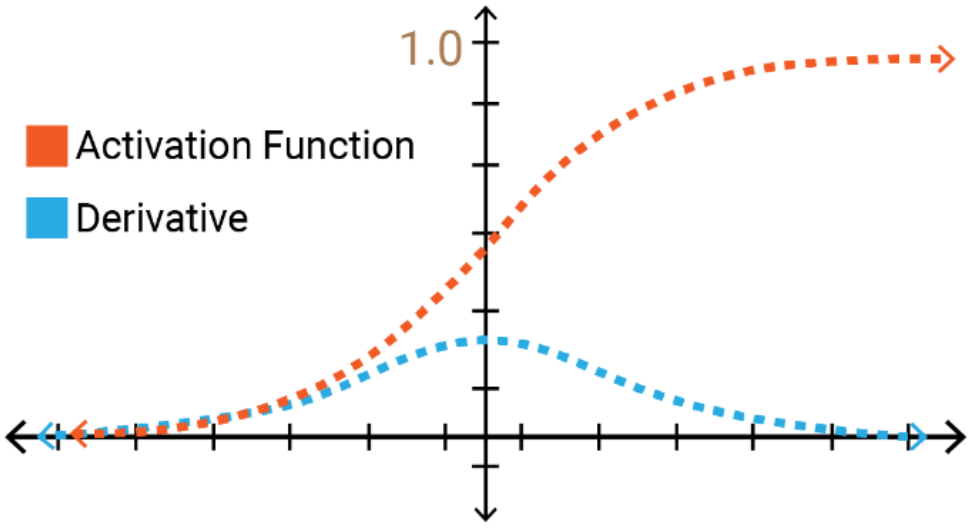

Figure 3.28: Example of activation function affected by the vanishing gradient problem[34]

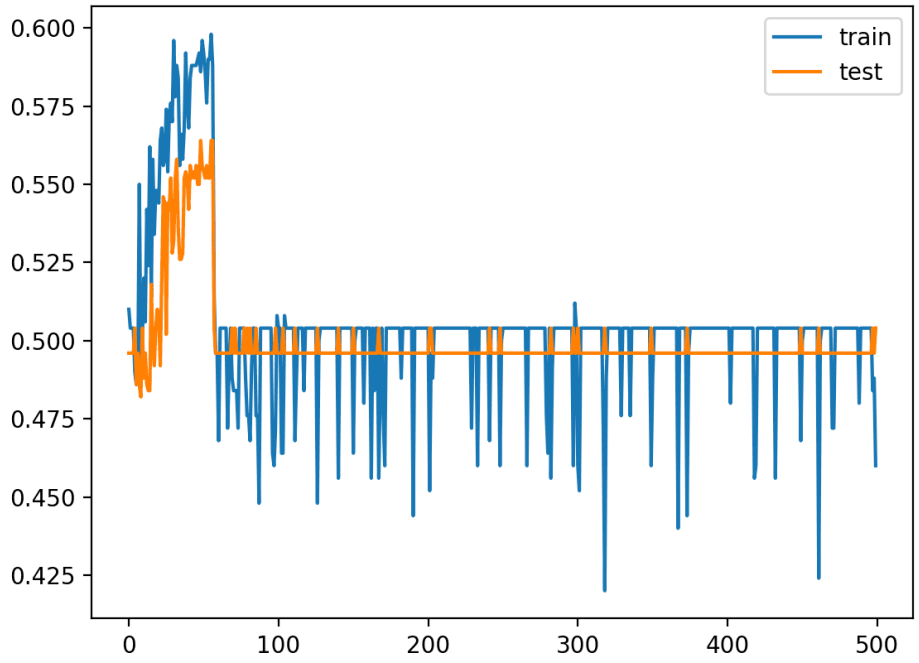

Figure 3.29: Training phase affected by vanishing gradient problem[35]

3.5.5 Root Mean Square Propagation Method (RMSProp)

More sophisticated algorithms are always based on the same basic principles of the GDM algorithm, but with some improvements. RMSProp adds the possibility to use a not fixed learning rate like in SGMD. This can be helpful because if the gradient is very small, the parameters update will be very small too, wasting computational time to compute the gradient and update the parameters without obtaining a correct step towards the minimum. On the contrary, a high gradient can lead to too long updating steps.

The idea behind the RMSProp algorithm is to use a moving average of the element-wise squares of the gradient, automatically scaling the learning rate as shown in Equation 3.21.

A parameter v_l is defined by Equation 3.20:

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2)[\nabla E(\theta_l)]^2 \qquad 3.20$$

β_2 is the squared decay rate factor of the moving average that is usually set to 0.9,0.99 and 0.999. This moving average is used to update each parameter individually.

$$\theta_{l+1} = \theta_l - \frac{\alpha \nabla E(\theta_l)}{\sqrt{v_l} + \epsilon} \quad 3.21$$

With RMSPorp, parameters with large gradient have a higher value of v_l and a correspondingly lower value of the modified learning rate $\frac{\alpha}{\sqrt{v_l}+\epsilon}$. ϵ is a small constant added to avoid division by zero.[30]

3.5.6 Adaptive Moment Estimation (Adam)

Adam algorithm uses a parameter update similar to RMSPropr, using an added momentum term, keeping an element-wise moving average of both parameter gradients and their squared values.

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1)\nabla E(\theta_l) \quad 3.22$$

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2)[\nabla E(\theta_l)]^2 \quad 3.23$$

β_1 is the gradient decay rate factor that has the same meaning of β_2 but is just applied on the momentum term. Adam uses both moving averages to update the network parameters as:

$$\theta_{l+1} = \theta_l - \frac{\alpha \mathrm{m_l}}{\sqrt{v_l} + \epsilon} \quad 3.24$$

The moving average of the gradient enables the parameter updates to pick up momentum in a certain direction, obtaining a different value of gradient. If the gradients are noisy, then the moving average of the gradient becomes smaller, along with the parameters update.[30][36]

3.6 Recurrent Neural Networks

In the previous section, only the *feed-forward* neural networks have been analyzed; it means that data can only flow from the input layer to the output one through the hidden layers. This characteristic avoids the presence of any cycle between the layers themselves: each observer is isolated from the others composing the dataset.

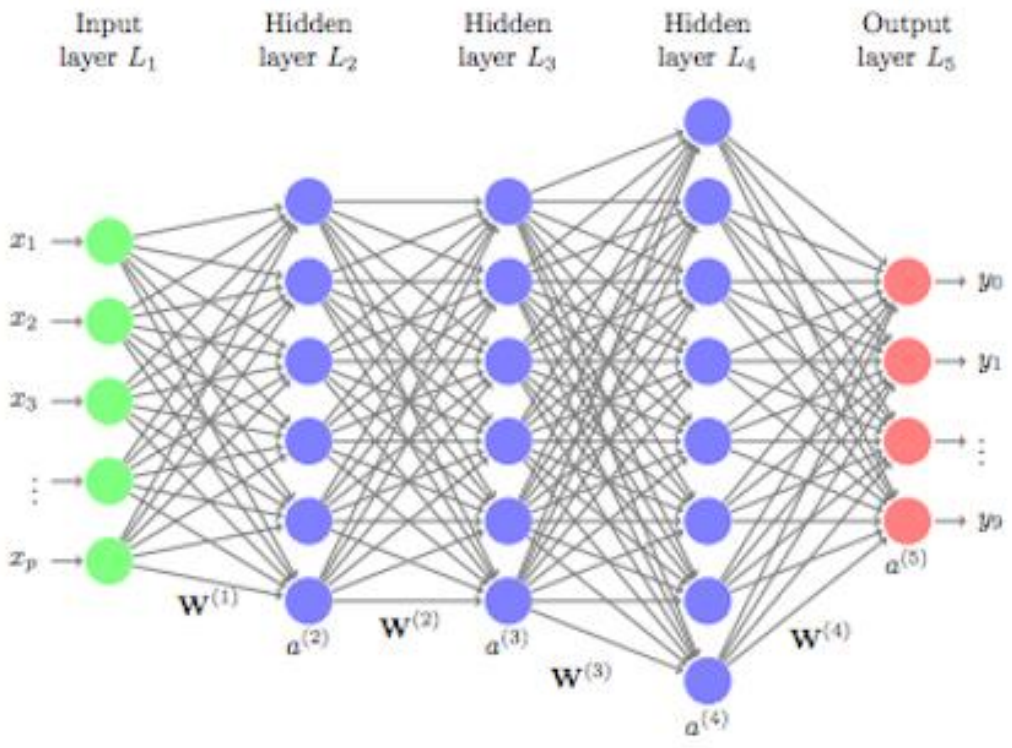

Figure 3.30: Feed-forward deep neural network

However, there are some cases in which the correlations between the observations are fundamental. For example, the vehicle speed at a specific time instant is not standalone, because it is strictly correlated with both the sections before and after the point. Therefore, the energy management system of an HEV can decide to switch on/off the engine, only by considering the entire sequence of speed, SoC, and other powertrain parameters, and not by considering a single time instant.[37]

For this reason, a feed-forward neural network is not optimal for the EMS of HEV application, but a neural network that can work with sequences and allows to consider the trend of the input features is needed.

Recurrent Neural Networks (RNN), on the other end, are a class of artificial neural networks where connections between nodes in a network architecture along are possible also in a temporal sequence, gaining dynamic temporal behaviour.

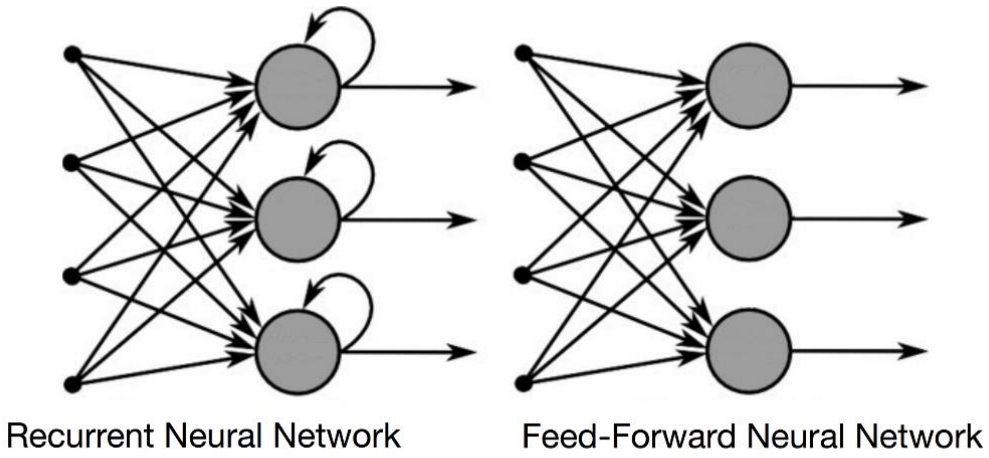

Figure 3.31: Difference between RNN and feed-forward NN. The output from a single neuron is feedback as input for the same neuron at the next time step

As evident from Figure 3.31, where each neuron is connected in a loop: therefore, the output of the previous time step is used as an additional input.
This characteristic of RNN allows the creation of a sort of state memory for each neuron that makes them suitable for dynamic and time-variant applications, such as the energy management system design of a hybrid electric vehicle.[19][38]
These types of networks can also be suited for making predictions on future time steps, starting from a previously known section. In the automotive field, they can be employed for obtaining speed forecasting in real-time applications exploiting Vehicle-to-Everything (V2X) connectivity. More broadly, these networks are also used to classify data from the analysis on time-variant signals (e.g., understand if a machine is damaged from its vibration motion analysis), to extrapolate the mining of a part of a written text (natural language processing) or other similar applications.
Several types of different layers can be used to build the network, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) layers, available also in MATLAB environment.

3.6.1 Gated Recurrent Unit Layer

Introduced by Kyunghyun Cho in 2014, a GRU layer allows considering the previous parts of a sequence exploited to compute the actual step output. This is possible thanks to the presence of the hidden state that contains, at a specific time step t, the output of the GRU layer of the time step. At each time step, the layer adds information to or removes information from the state, using the so-called *gates*.

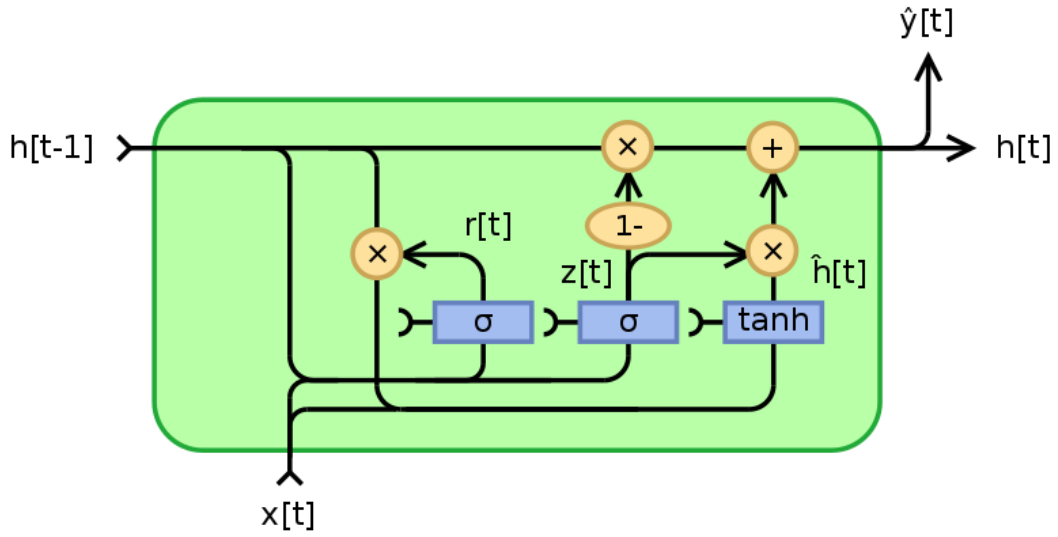

Figure 3.32: Gated Recurrent Unit (GRU) internal architecture

In a gate, the typical operations of a classic perceptron are done, with the presence of weights and bias that, similarly to a classical neural network, are properly tuned during the training phase.
The fully gated version of the GRU layer has two gates, the *reset gate*: which controls the level of state reset, and *the update gate*, which controls the level of state update.
The mathematical computation performed in this layer is expressed in Equations 3.25, 3.26, 3.27 and 3.28:

$$\bar{z_t} = \sigma_g\left(\overline{W_z}\bar{x_t} + \overline{U_z}\overline{h_{t-1}} + b_z\right) \quad 3.25$$

$$\bar{r_t} = \sigma_g\left(\overline{W_r}\bar{x_t} + \overline{U_r}\overline{h_{t-1}} + b_r\right) \quad 3.26$$

$$\bar{\hat{h}_t} = \phi_h\left(\overline{W_h}\bar{x_t} + \overline{U_h}\left(\bar{r_t}\odot\overline{h_{t-1}}\right) + b_h\right) \quad 3.27$$

$$\bar{h_t} = (1 - \bar{z_t})\odot\overline{h_{t-1}} + \bar{z_t}\odot\bar{\hat{h}_t} \quad 3.28$$

Where:

- $\bar{x_t}$: input vector
- $\bar{h_t}$: output vector at time t
- $\bar{\hat{h}_t}$: candidate activation vector, it is the value of the hidden state of the layer, allowing to consider also previous time step output $\overline{h_{t-1}}$
- $\bar{z_t}$: update gate vector, it is the output of the update gate
- $\overline{W}, \overline{U}, b$: parameter matrices and vector with $\overline{W}$ that is the weights matrix while $\overline{U}$ is the recurrent weights matrix
- σ_g, ϕ_h are the activation function for gates and candidate state vector

As it is possible to note from Equation 3.28 the output of the gate $\overline{h_t}$ is computed considering the output of the previous time step $\overline{h_{t-1}}$ and the cell activation vector $\widehat{\overline{h_t}}$ in a weighted average operation with an update gate vector as weight.[39]

3.6.2 Long Short-Term Memory Layer

LSTM layer[40] is an updated version of the classic GRU layer to handle the recurrent neural network. One of the problems of RNN with the GRU layer is its inability to learn long-time dependencies because these neural networks suffer the vanishing or exploding gradient problem. It is particularly evident if the network tries to memorize long-time dependencies information during the training phase with the backpropagation algorithm.
LSTM bypasses this problem since it allows gradients to flow remaining unchanged avoiding its vanishment but not its explosion.

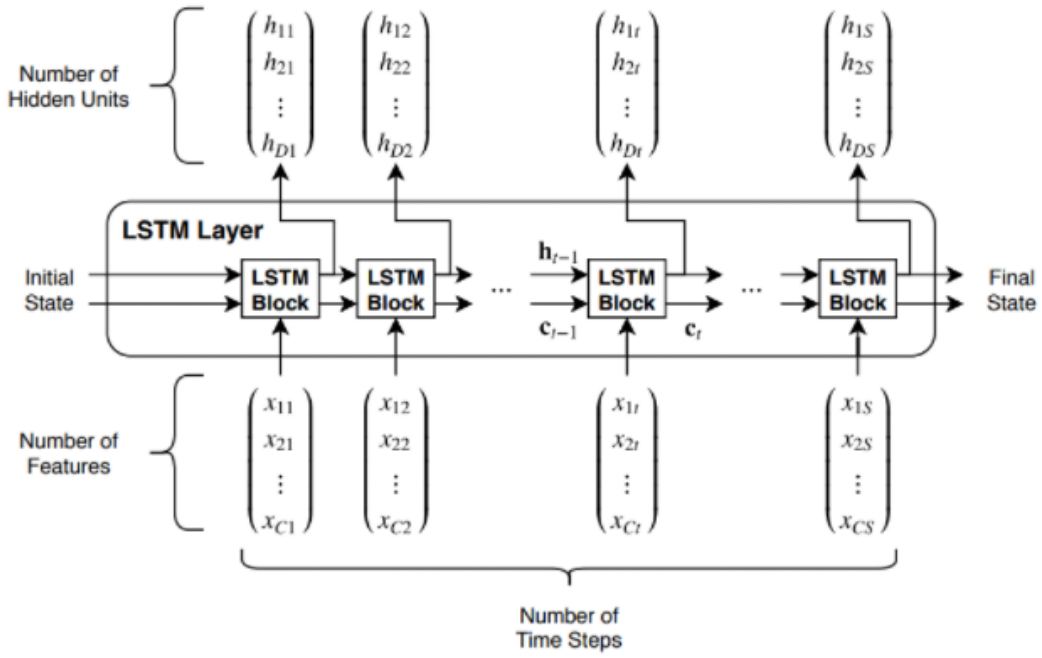

Figure 3.33: LSTM layer flow of information[41]

In Figure 3.33 The flow of information inside an LSTM layer is displayed. At each time step, the layer receives as input a vector of C features: they are used to compute the output and to update the cell state. It is a buffer of memory l that learns long time dependencies in the time-sequence data. The state of the layer consists of the *hidden state* h_{t-1} and the *cell state* c_{t-1}. The cell state contains information learned from the previous time steps and, at each time step, the layer adds information to or removes information from the cell state, using, such as in GRU, the *gates*.

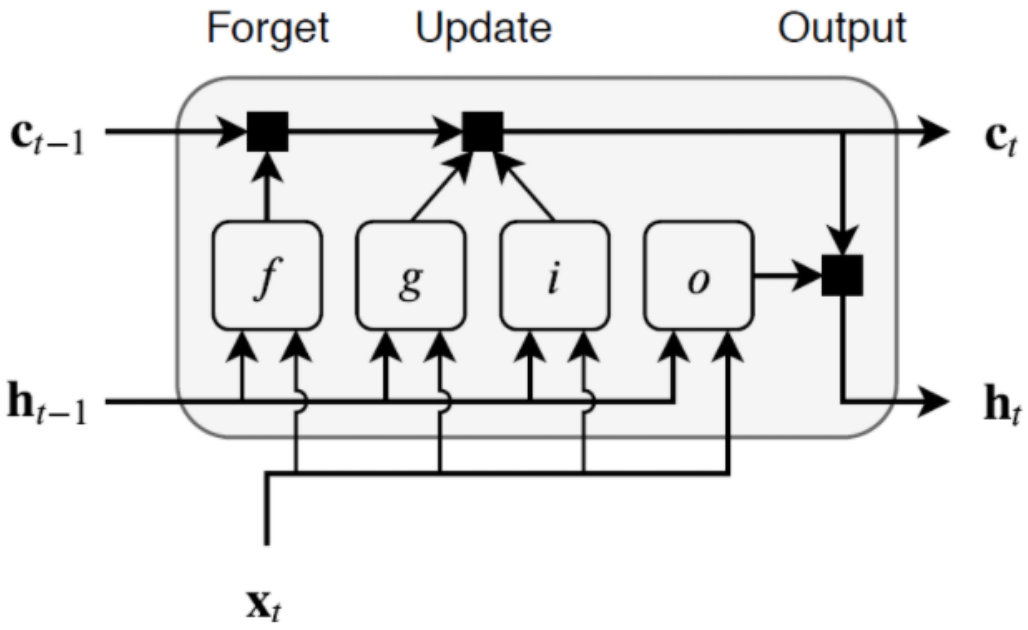

Figure 3.34: LSTM layer internal structure

LSTM layer has four gates Whose weights and bias are tuned during the training phase.

- i is the input gate controlling the level of cell state update
- f is the forget gate and control level of cell state reset
- g is the cell candidate gate and add information to cell state
- o is the output gate and control level of cell state added to hidden state (or output)

As in GRU, the weights matrix contains two different kinds of weights: W referred to the actual time step and R that are the recurrent weights, referred to information coming from previous time steps. The formulas that describe this layer are expressed in Equations 3.29, 3.30, 3.31 and 3.32.

$$i_t = \sigma_g\left(\overline{W_i}\overline{x_t} + \overline{R_i}\overline{h_{t-1}} + b_i\right) \quad 3.29$$

$$f_t = \sigma_g\left(\overline{W_f}\overline{x_t} + \overline{R_f}\overline{h_{t-1}} + b_f\right) \quad 3.30$$

$$g_t = \sigma_c\left(\overline{W_g}\overline{x_t} + \overline{R_g}\overline{h_{t-1}} + b_g\right) \quad 3.31$$

$$o_t = \sigma_g\left(\overline{W_o}\overline{x_t} + \overline{R_o}\overline{h_{t-1}} + b_o\right) \quad 3.32$$

The operations are made in each gate of the layer. It is possible to note that each gate considers information from the input vector at the actual time step and information from the output of the previous time step. The forget gate is able to delete information also from old time steps when it is not relevant anymore. If it is not activated, the cell candidate and input gates continue to add information to the past stored in the cell states.

Finally, the cell state and the output are computed by Equation 3.33.

$$\overline{c_t} = f_t \odot \overline{c_{t-1}} + i_t \odot g_t \overline{h_t} = o_t \odot \sigma_c(\overline{c_t}) \quad 3.33$$

σ_c, σ_g are the activation functions used in the layer, the same introduced for Artificial Neural in section 3.5.[41]

4 Case Study

4.1 Vehicle Specification

The vehicle used to perform all the analysis is a plug-in Hybrid Electric Vehicle, with a P2 architecture, as schematized in Figure 4.1, available on the market.

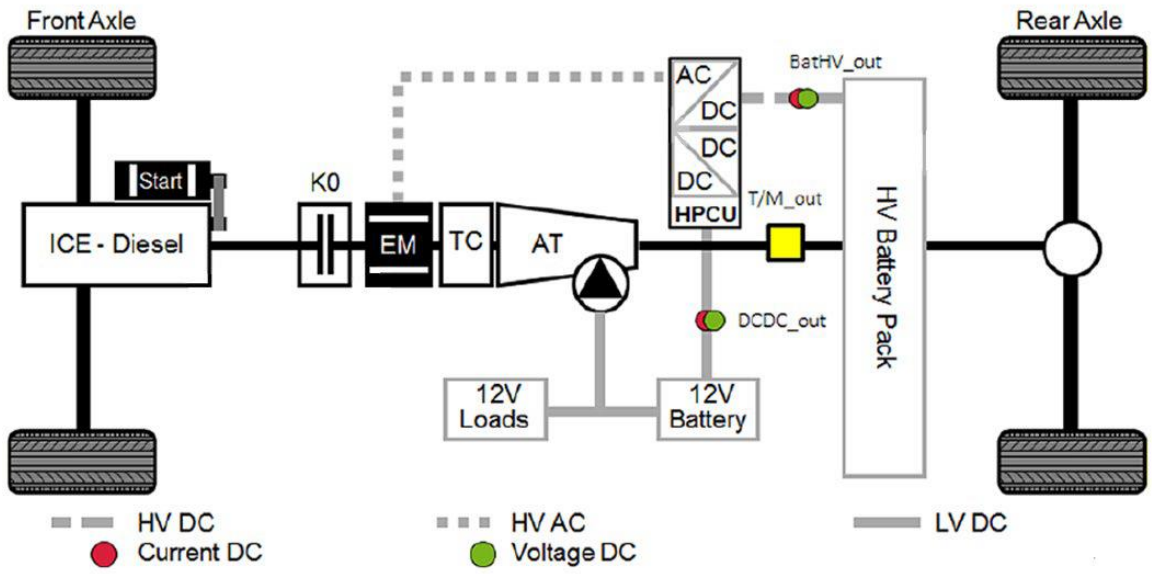

Figure 4.1: Powertrain layout

This vehicle has been subjected to a massive experimental campaign to extract all powertrain parameters needed to characterize how the vehicle powertrain operates and select the relative operating mode during road driving.[42]

The experimental setup consists of an AWD chassis dynamometer, tested on RDE scenarios, with a PEMS equipped vehicle.

The operating modes, available from the control unit are the following, depending on the driver's power demand:

- **Hybrid mode**: ICE and EM act together and all hybrid functions, such as electric driving boost and energy recovery, are available and selected according to the driving conditions.
- **Electric mode**: only the EM propels the vehicle, exploitable mainly in the city centre, where the driver power demand is usually limited with respect to highway driving.

- **E-save**: ICE and EM propel the vehicle, ensuring a charge sustaining operation.
- **Charge**: the ICE provides more energy than the driver's request, constantly charging the battery and the EM does not provide any contribution to vehicle propulsion.

Concerning the vehicle characteristics, it is a P2 Diesel Plug-In Hybrid Electric Vehicle (PHEV), equipped with a conventional 1950 cc Diesel engine, able to provide 143 kW as maximum power with 400 Nm of maximum torque. The electric motor is a 90 kW/440 Nm Permanent Magnet Synchronous (PMSM) EM, that can be uncoupled from the engine employing an auxiliary clutch, that allows using some operating modes with the engine turned off, avoiding useless wasting of energy to drag the engine.

The hybrid powertrain is then followed by a 9-speed automatic transmission (AT) and a torque converter (TC) that transfer the powertrain developed torque to the rear wheels.

The battery is a 13.5 kWh Li-Ion nickel-manganese-cobalt-oxide (Li-NMC) HV battery, that can provide energy to only electric propel the vehicle at a top speed of 130 km/h.

In Table 4.1 the main vehicle characteristics are listed.

Table 4.1: Vehicle specification

Engine	
Engine Type	In-line 4 cyl. Turbo Diesel
Displacement	1950 cc
Max Power/Max Torque	143kW @3800rpm / 400Nm @1600-2800 rpm
Compression Ratio	15.5:1
Transmission	
Transmission Type	9 –AT w/ Torque Converter
Speed Ratios	I 5.36 IV 1.64 VII 0.87 II 3.25 V 1.22 VIII 0.72 III 2.26 VI 1.00 IX 0.61
Reverse-Final Drive	-4.93/2.65
Vehicle	
Curb Weight	2060 kg
Configuration	Rear Wheel Drive (RWD)
Electric Motor	
Electric Motor Type	PM Synchronous motor
Max Power/Max Torque	90 kW / 440 Nm @1750rpm

Max Speed	6000 rpm
High Voltage Battery	
Battery Type	Lithium- NMC
Rated Voltage	365V
Capacity	13.5kWh / 37 Ah
Cooling System	Water Cooled

4.2 Driving Cycles

Deep Learning models requires a huge quantity of data to effectively train the model and obtain satisfactory results during the test phase. A Hybrid Electric Vehicle Energy Management System application requires collecting data in terms of driving cycles that should be afterwards preprocessed to extract the most relevant features.

The driving cycles database is composed of type-approval cycles, such as NEDC, WLTC or FTP75, and some more relevant RDE cycles. The latter category is more relevant than type approval ones, to be passed to the EMS NN-based model because the ultimate goal of this thesis work is to develop an ECU real-time implemented version of the EMS, that should work under real-driving conditions and RDE cycles want to simulate them.

4.2.1 RDE

These driving test cycles have been introduced in the regulation because there is an obvious gap between emissions computed in laboratories on standard type-approval tests, like WLTC, and real driving conditions ones. This is due because, a laboratory environment, cannot consider several factors that are present in normal on-road driving, such as traffics, presents of lights, hard and not constant acceleration, different ambient conditions and so on. Real Driving Emissions (RDE) tests have been introduced to also consider these factors when a vehicle has to be approved by the regulation. RDE tests were introduced by Euro 6d Temp regulation, becoming compulsory from September 2019.

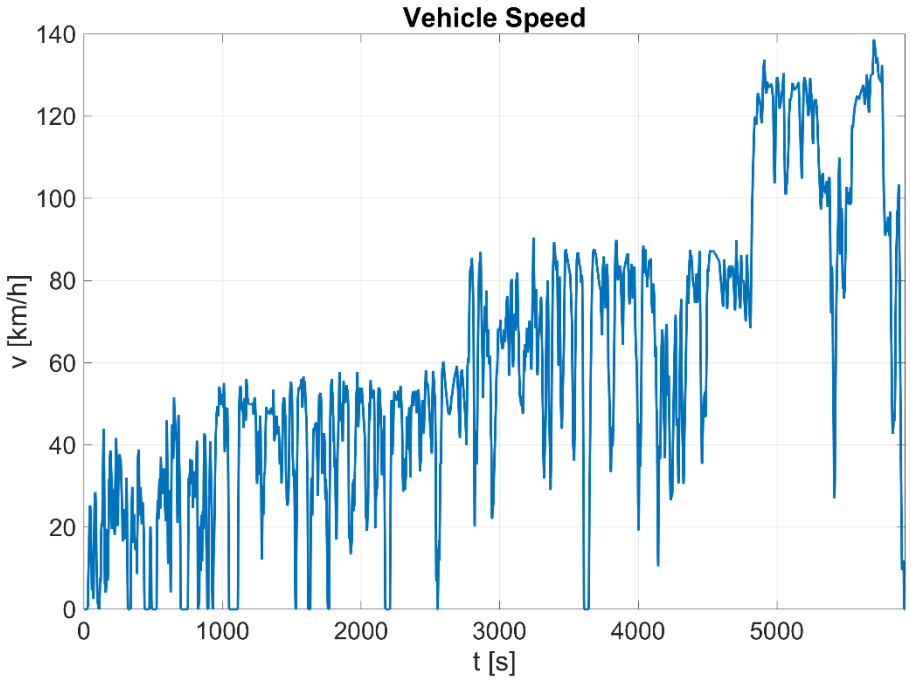

Figure 4.2: Example of RDE Cycle

However, a completely free on-road driving cycle is not suitable for regulations purpose, because, depending on the driving conditions, ICE emissions are obviously different. For this reason, RDE is composed of three segments, Urban, Rural and Motorway, that must respect some imposed constraints, shown in Table 4.2.[4]

Table 4.2: Cycle characteristic to be an RDE compliant driving cycle

Segment	Percentage of the total distance	Minimum Distance	Instantaneous speed	Average speed
Urban	29-44%	16 km	$v \leq 60\ km/h$	$15\ km/h < v \leq 40\ km/h$
Rural	23-43%	16 km	$60\ km/h < v \leq 90\ km/h$	$60\ km/h < v \leq 90\ km/h$
Motorway	23-43%	16 km	$v > 90\ km/h$	$v > 90\ km/h$

Concerning LSTM neural network training, since this type of network allows to consider past information in output computation, the type of driving cycle that the network takes as input is important during the training phase. Thus, RDE cycles are the most suitable for network training applications. Nevertheless, a deep learning application requires a huge quantity of training data, and

due to the limited quantity of RDE cycles available, some type-approval driving cycles have been used to train the network, coupled with an RDE database expansion procedure.

4.3 Database Expansion

The driving cycles database is composed of 34 cycles, consisting of RDE, performed around the city of Turin, and type-approval ones. However, they cannot be all used because some cycles are not suitable for LSTM network training for several reasons, such as the presence of only one driving section (i.e., highway part) neglecting the other, or because some of them are not able to represent an on-road driving condition due to not representative vehicle speed trends.

To obviate at driving cycles database poorness, a database expansion strategy has been adopted. This procedure creates some artificial RDE cycles, starting from all the 34 available cycles.

They have been divided into patterns that start from stand-still and finish at stand-still to be afterwards randomly combined. Each pattern is then categorized in the relative driving section (Urban, Rural, Highway), considering the speed limits of RDE sections and an energetic index, $I_{v^2} = \frac{\int_{t_i}^{t_f} v^2(t)dt}{t_f - t_i}$, that in [43] is demonstrating to be able to correctly classify the sub-pattern category.

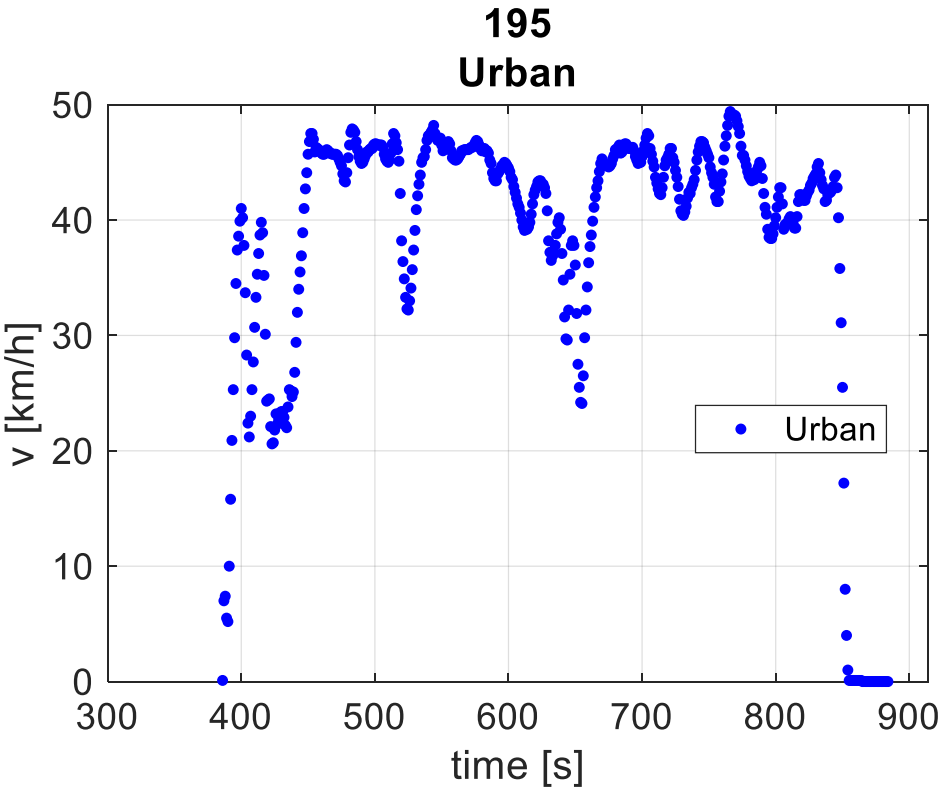

Figure 4.3: Example of urban pattern

Then, a scale factor is applied to each driving pattern, during the entire RDE compliant driving cycle creation, to introduce some differences among the same pattern but used in different cycles.
Finally, the driving patterns are randomly concatenated to create the entire driving cycle, ensuring that RDE compliance characteristics, introduced in the above-written section, are satisfied.
Thanks to the database expansion strategy, a database of 73 driving cycles have been used to train and test the networks, obtaining a remarkable improvement in terms of RMSE and accuracy both in the train and test phase.

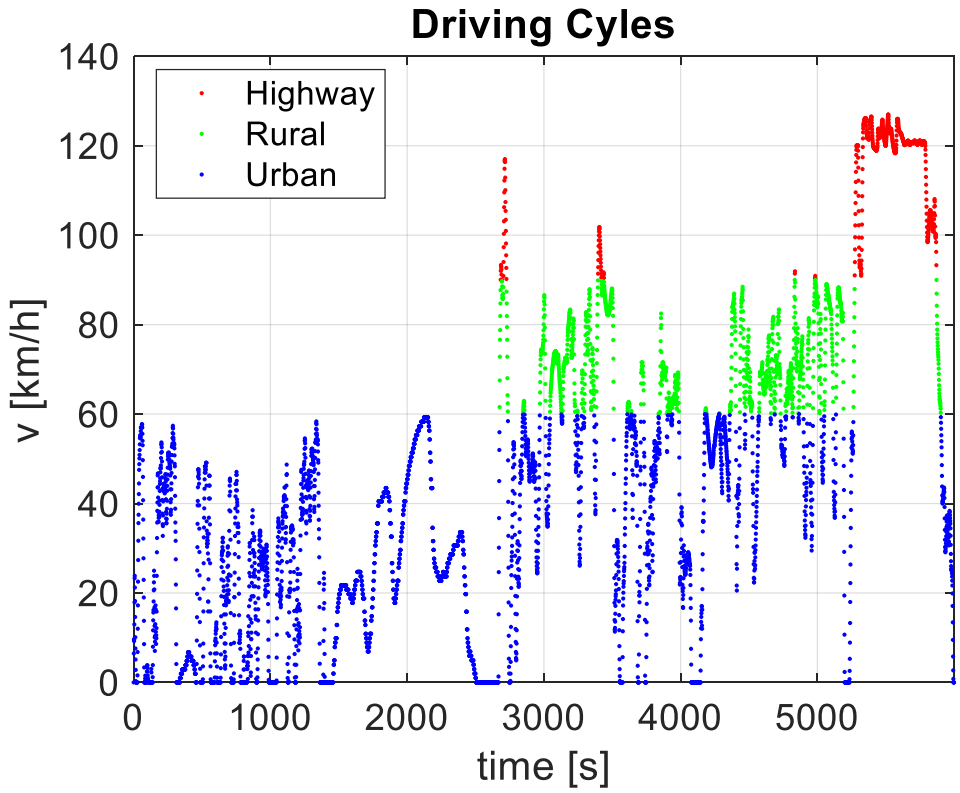

Figure 4.4: Example of RDE compliant generated driving cycle

4.4 Vehicle Modelling

The goal of this thesis work is to develop a control strategy that can minimize the overall fuel consumption of the vehicle on real driving conditions cycles. An additional target is that the designed Energy Management System future implementable on a real ECU, target achievable only if the online computational time of the EMS is suitable with the driving cycle duration.
To properly design the EMS, simulation tools became central to model the entire vehicle and the hybrid powertrain, since a full experimental design is too much expensive due to the huge number of

experimental data needed and costs. The vehicle and powertrain behaviour has been modelled employing GT-SUITE, a mono-dimensional (1D) fluid-dynamic numerical code developed by Gamma Technologies (GT), a leader in the CFD simulation applied to the automotive field. Since fuel consumption and CO_2 emissions are the main interested quantity the so-called *quasi-static* approach has been exploited because it models the machine's behaviour through look-up tables, allowing to also represent the transient behaviour in the form of consecutive steady-states operating points. More sophisticated approaches, such as a full 3D CFD simulation of the engine, also limited to the in-cylinder behaviour characterization, are not suitable for this thesis purpose in terms of computational efforts and a simplified quasi-static approach is accurate enough to correctly predict the total fuel consumption of the vehicle. However, if the goal of the analysis is to predict some phenomena that are related to how the combustion process develops inside the cylinder of the engine, such as some pollutant emissions, like NO_x and soot, this approach is not accurate enough and a more detailed CFD analysis is needed.[44]

Concerning the electric motor modelling approach, as for the internal combustion engine, a map-based methodology has been adopted, describing its behaviour using torque and efficiency maps.[5]

Since the interest in battery performance is increasing due to the higher electrification trend in the automotive field, a well-detailed battery model is as important as the engine or the electric motor model. However, this is not an easy task and, for this thesis purpose, a simple static model, for SOC evaluation has been used. The battery model consists of an equivalent electric circuit made by an ideal voltage generator in series with a resistor, neglecting all the electric dynamic behaviour of the system.[5]

The SOC can be easily evaluated with the following relationship:

$$SOC(t) = \frac{Q(t)}{Q_{max}} = \frac{\int_0^t i(t)dt}{Q_{max}} \tag{4.1}$$

Where:

- $i(t)$ is the instantaneous electric current that flows into or from the batter.
- $Q(t)$ is the actual battery charge.
- Q_{max} is the maximum charge level.

4.4.1 Forward Dynamic Analysis

Concerning vehicle dynamic modelling, a *forward dynamic analysis* has been used to develop the GT-SUITE model. This method solves the longitudinal vehicle dynamics equation, described in

section 4.5, to determine engine speed and torque demand. The vehicle speed is seen as a target value that a Proportional-Integral-Derivative (PID) controller, which represents a driver model, has to reach acting on the accelerator and brake actuation, generating a tractive or braking force. Then, the vehicle acceleration and speed can be computed by means, again, of the vehicle dynamic equation.

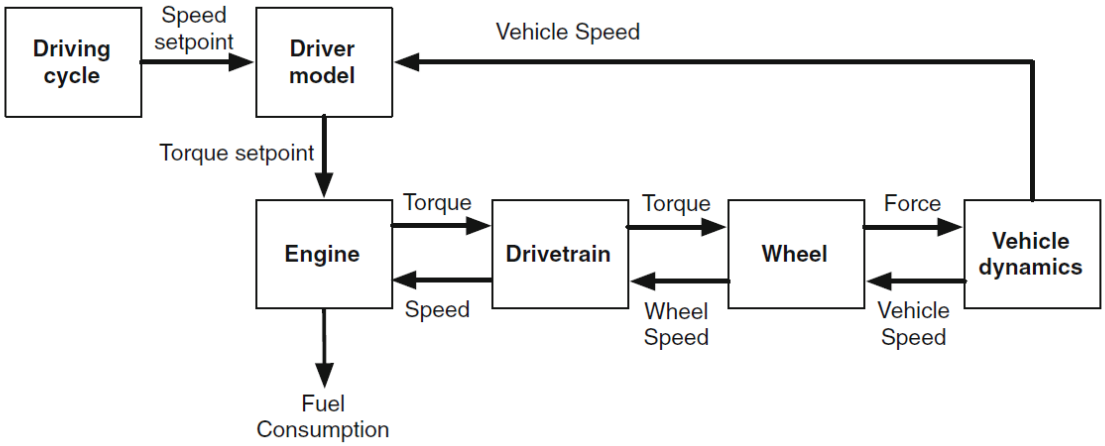

Figure 4.5: Information flow in a forward simulator[5]

Concerning the development of the Energy Management System, it has been designed in Simulink, the market leader in automatic control modelling, a software developed by MathWorks, where a multidomain dynamic system can be easily modelled. Finally, the EMS designed in Simulink has been coupled with the vehicle model, developed in GT-SUITE, and it decides how to split the power demand between the engine and the electric motor.

4.4.2 Backward Kinematic Analysis

The backward approach imposes both vehicle speed and road grade, unlike the *forward* approach that views these quantities as a target to be reached from the driver model.

With backward kinematic analysis, the driving cycle is divided into small time intervals where both speed, acceleration and torques remain constant. This assumption simplifies the model with respect to the forward dynamic approach in terms of computational efforts, but the accuracy of this model is lower.

The internal powertrain dynamics are neglected and a look-up table approach defining efficiencies, fuel rate and power losses is adopted.

Since vehicle speed and acceleration are imposed on each time interval, it is possible to derive both engine, electric motor and wheel speed, through a simple kinematic relationship, neglecting the transient response of these sub-systems and considering all the wheels in pure rolling conditions.

$$\omega_{wh} = \frac{v}{r_{wh}} \quad 4.2$$

$$\omega_{gb} = \omega_{wh} i_{fd} i_{gb} \quad 4.3$$

Where:

- ω_{wh} is the wheel speed.
- v is the vehicle speed.
- ω_{gb} is the inlet gearbox rational speed (electric motor side)
- r_{wh} is the effective wheel rolling radius.
- i_{fd} is the final drive gear ratio.
- i_{gb} is the gearbox gear ratio (depending on the actual gear).

Power demand computation is described in section 4.5. For sake of brevity, in this section is reported the final relationship.

$$P_{gb} = \left(f_0 + M_{veh}\, g\, sin(\delta) + f_1\, v + f_2\, v^2 + M_{eq}\, a\right) v\, \eta_{gb}^{sign\left(-\left(f_0 + M_{veh}\, g\, sin(\delta) + f_1\, v + f_2\, v^2 + M_{eq}\, a\right)\right)} \quad 4.4$$

Where:

- P_{gb} is the gearbox inlet power, seen as a power request from the hybrid powertrain.
- f_0, f_1, f_2 are the coast-down coefficients.
- M_{veh} is the vehicle mass.
- M_{eq} is the equivalent mass of the vehicle considering all the driveline inertia contributions.
- δ is the road slope.
- g is the gravitational force.
- a is the vehicle acceleration.
- η_{gb} is the gearbox efficiency, computed by a map interpolation.

Known the powertrain power request, engine speed and established the power split between ICE and EM, it is possible to easily compute the engine Break Mean Effective Pressure (BMEP) with the following relationship:

$$BMEP = \frac{60\, i\, P_{eng}}{n\, V} \quad 4.5$$

Where:

- P_{eng} is the engine power demand.
- i is the number of revolutions per power stroke.
- n is the engine speed in rad/s.
- V is the engine displacement.

Known BMEP it is possible to compute the instantaneous fuel consumption by map interpolating using engine speed and BMEP values.

The fuel cumulate quantity can be obtained by integrating the instantaneous fuel consumption over the entire driving cycle.

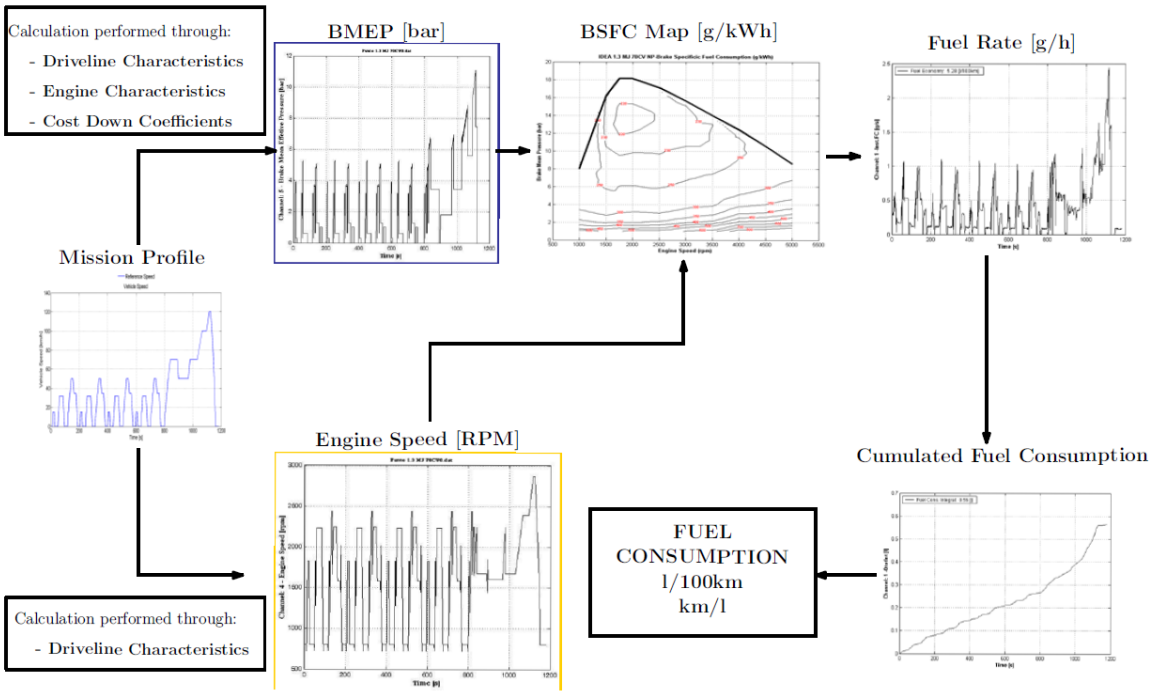

Figure 4.6: Information flow in a backward kinematic approach[2]

The backward kinematic approach has been used to build a simple kinematic model of the entire vehicle to realize a Dynamic Programming optimization on the whole driving cycles database. The results of DP optimization are used as the target values that the neural network-based model has to reach.[2][5]

4.5 Vehicle Equations of Motion

Usually, when the powertrain of the vehicle has to be designed, it is not necessary to investigate all vehicle dynamics in deep, neither a full detailed 1D modelling approach is appropriate. The vehicle

dynamics answers are limited to just vehicle speed, acceleration and power (or torque) demand, that have to be fully satisfied by the powertrain. So, an energetic approach is sufficient to properly descript all the needed vehicle dynamics.

The vehicle can be seen as a single point of mass (lumped in the centre of gravity of the vehicle), characterized by the following equilibrium equations:[5]

$$M_{eq}\frac{dv}{dt} = F_{pwt} + F_{brakes} + F_{roll} + F_{aero} + F_{grade} \quad 4.6$$

$$M_{eq} = M_{veh} + \frac{2\,I_{wh,f} + 2\,I_{wh,r}}{r_{wh}^2} + \frac{I_{eng}}{r_{wh}^2}\,i_{fd}^2\,i_{gb}^2 + \frac{I_{EM}}{r_{wh}^2}\,i_{fd}^2\,i_{gb}^2 \quad 4.7$$

Where:

- F_{pwt} is the powertrain tractive force.
- F_{brakes} is the mechanical braking force (the regenerative braking action is considered in the powertrain force).
- F_{roll} is the rolling resistance.
- F_{aero} is the aerodynamic resistance.
- F_{grade} is the road slope resistance.
- I is referred to the inertia of the related component.

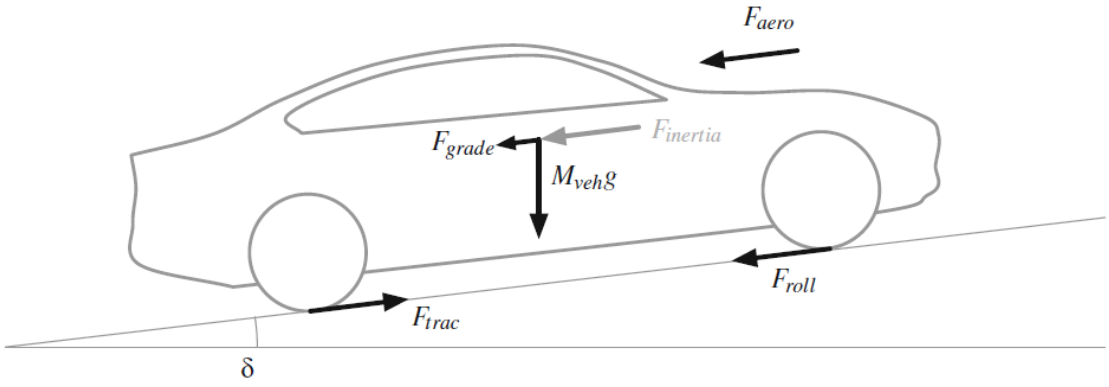

Figure 4.7: Vehicle free body diagram along the longitudinal direction[5]

M_{eq} is an equivalent mass of the vehicle that considers all the inertia contributes from the driveline components, necessary to avoid a too simple model that considers only the mass of the vehicle, neglecting all the rotational inertia contributions.

$\frac{dv}{dt}$ is the acceleration of the vehicle, described in terms of derivative of the speed to the time.
The single resistance components have the following analytical expressions:

$$F_{roll} = c_{roll} M_{veh} g cos(\delta) \quad 4.8$$

Where:

- g is the gravitational acceleration.
- δ is the road slope.
- M_{veh} is the vehicle mass, only this contribution without considering the driveline components inertia effects.
- c_{roll} is the rolling resistance coefficient, that can be modelled as a third-grade polynomial function of vehicle speed.

$$F_{aero} = \frac{1}{2} \rho_{air} A_f C_d v^2 \quad 4.9$$

Where:

- ρ_{air} is the air density.
- A_f is the vehicle frontal section.
- C_d is the aerodynamic drag coefficient.

$$F_{grade} = M_{veh} g sin(\delta) \quad 4.10$$

Since aerodynamic and rolling forces are quite hard to be determined because rolling resistance coefficient and aerodynamic drag coefficient do not have a close form relationship, they are experimentally determined. The two resistive actions are experimentally determined in the so-called *coast-down* test. It consists of a free vehicle deceleration test, a condition in which only these two resistive components act on the vehicle. Measuring the instantaneous vehicle speed is possible to determine the total drag force acting on the vehicle itself.

$$F_{aero+roll} = C_0 + C_1 v + C_2 v^2 \quad 4.11$$

Where C_0, C_1, C_2 are called *Coast-Down Coefficients*.[44]

4.6 Methodology

The Energy Management System of a Hybrid Electric Vehicle can be implemented by exploiting different algorithms or models, optimizing one of the quantities of interest, such as minimizing fuel consumption or pollutant emissions, as stated in Chapter 2.

The main purpose of EMS developed in this thesis work is to minimize the overall fuel consumption of the vehicle on different driving cycles, focusing mainly on RDE cycles since they are the most representative of real on-rode driving conditions.

The EMS should imitate the behaviour of an optimal control algorithm, such as Dynamic Programming, obtaining a sub-optimal solution implementable online, such as a neural network-based supervised learning model.

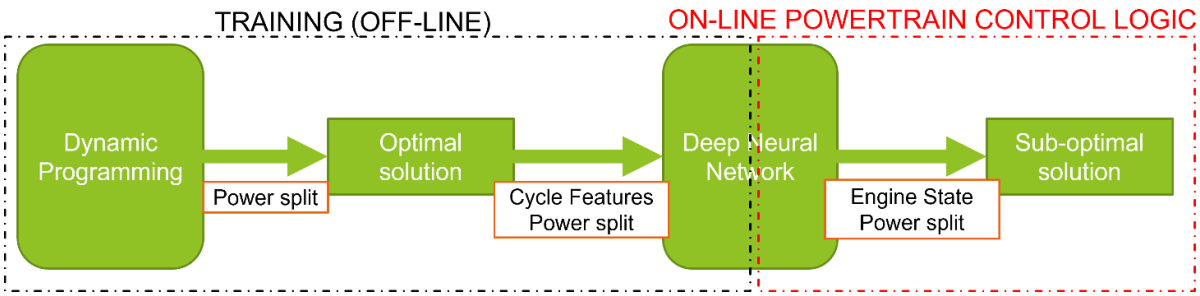

Figure 4.8: Supervised learning algorithm offline training process from DP and online implementation

4.6.1 Dynamic Programming

Dynamic Programming (DP) control algorithm can provide the optimal solution of the control problem, in this case, it allows to achieve the lowest fuel consumption on a pre-defined driving cycle, with the constrain of the complete knowledge in advance of the driving cycle.

It is easy to understand that, concerning an online application of the purposed EMS, the complete knowledge of the driving cycle is impossible to achieve, and this is one of the issues that makes this control algorithm not directly suitable for this thesis purpose. The main limitation of DP is related to its high computational effort that takes a too long time to optimize the power split on the driving cycle on an online optical implementation.

Despite this hard limitation, DP can be used as a benchmark to be reached by an online implementable control algorithm, such as a supervised learning model.

The Dynamic Programming model has been exploited to obtain the actual engine state and the Brake Mean Effective Pressure (BMEP) of the internal combustion engine each second, quantities used as labels during the training phase of the supervised learning model.

The developed DP model is based on a backward kinematic approach, deeper analysed in section 4.4.2, and it uses one state variable, State of Charge (SOC) of the battery, and two control variables, engine state and electric machine power, with a discretization time of 1 second.

Gear shift profile has been imposed using experimental values, for those cycles with this information available, and a biLSTM Neural Network model, trained with the experimental gear profile, for those cycles without the experimental gear profile.

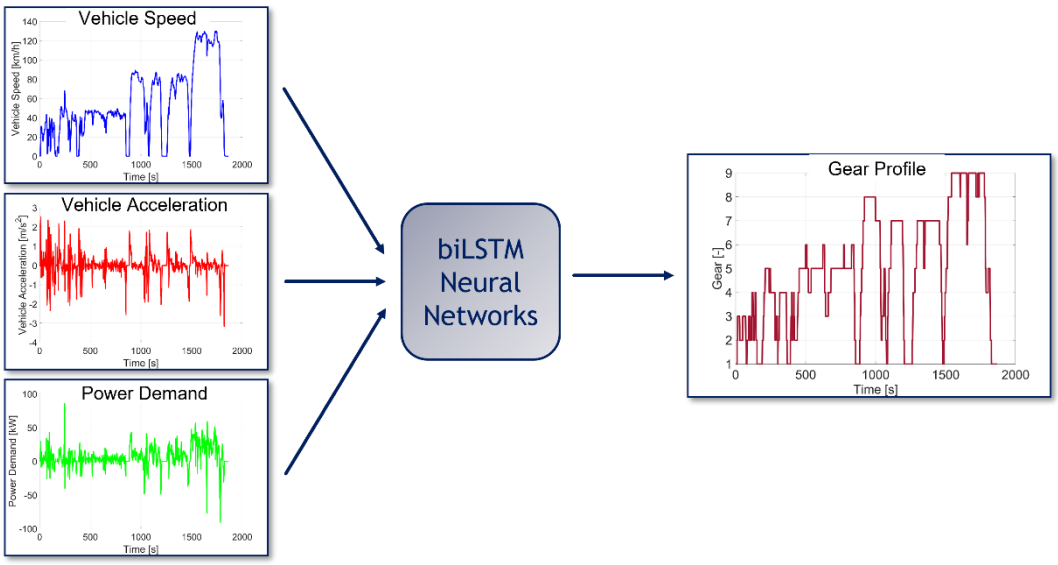

Figure 4.9: Gear profile biLSTM Neural Network model. Vehicle speed, vehicle acceleration and vehicle power demand are the selected features

It is important to choose the right control and state variables considering a trade-off between computational time and accuracy of the simulation since the computational effort increases exponentially with the number of control and state variables used. Of course, considering other state variables, such as the engine state, allows the development of a more accurate model which should make the DP model more representative of the real vehicle model. However, since a huge quantity of simulation has been performed, the best solution considers just the above-mentioned variables.

The cost function to be minimized consider just the overall fuel consumption and it is the following.

$$J(t, e(t), P_{EM}(t)) = \sum_{t=0}^{T} \dot{m}_f(t, e(t), P_{EM}(t)) \qquad 4.12$$

Where:

- $e(t)$ is the engine state.
- $P_{EM}(t)$ is the electric machine power.
- $\dot{m}_f$ is the engine fuel rate.
- T is the final simulation time.

Since this model is not continuous, due to computational effort reasons, the cost function J is not characterized by the presence of integration, such as in Equation 2.1, but of a sum operation.

The simulations have been performed both in charge sustaining operation, considering an initial SOC of 0.2, and charge depleting, considering different SOC initial values with SOC final value set at 0.2. In the following figures is possible to see some results of DP simulation on an RDE cycle.

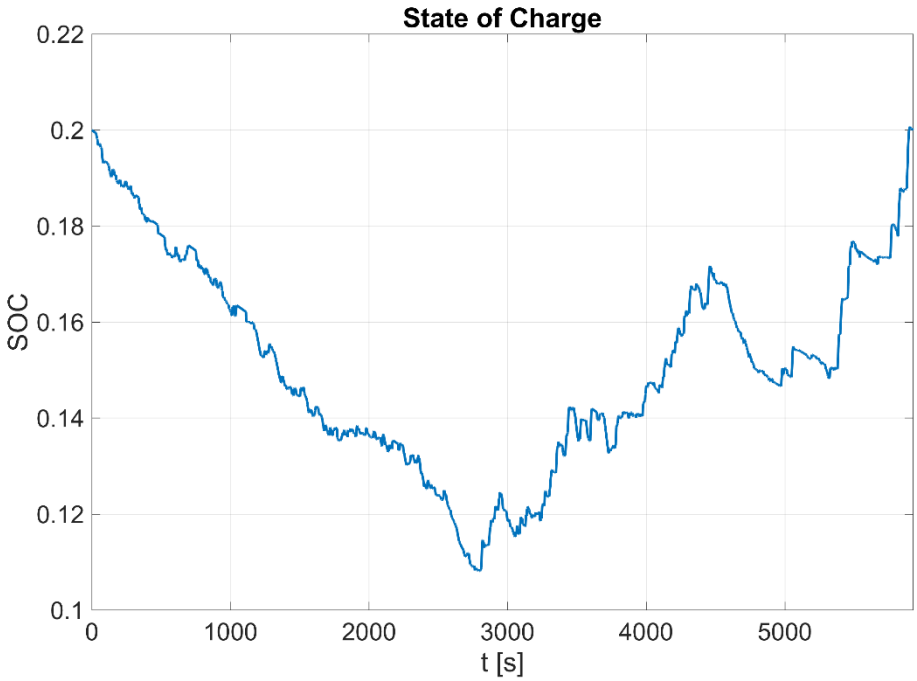

Figure 4.10: State of Charge in charge sustaining mode

From Figure 4.10 it is possible to note the perfect behaviour of DP optimization in terms of charge sustainability. The main goal of this battery management strategy is to obtain the initial value of SOC of the battery at the end of the driving cycle.

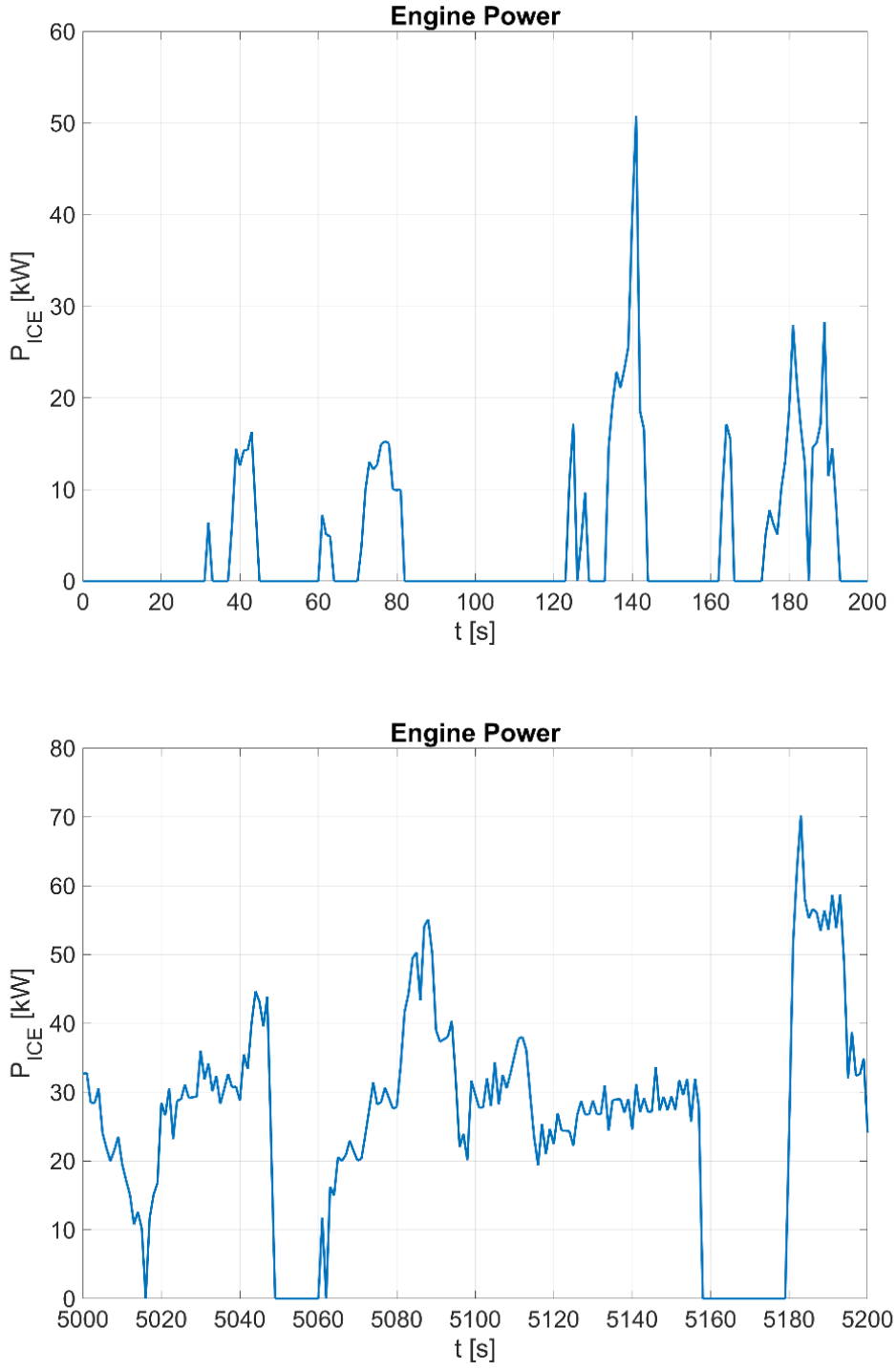

Figure 4.11: Different ICE behaviour between urban pattern (above) of the cycle and highway pattern (below)

Another interesting consideration comes from Figure 4.11, which plots internal combustion engine power in two different driving conditions. The first figure is referred to an urban driving condition, and the engine is exploited only during peaks of power demand, while the major part of the urban part is run in electric mode, an aspect confirmed from the SOC trend that, in the first part, tends to

decrease, discharging the battery. The second figure presents a different behaviour, with major exploitation of the engine in pure thermal mode or parallel mode. This is due to the higher power demand in the highway part due to the higher value of vehicle speed. Also for the highway part of the cycle is possible to confirm this behaviour from the SOC trend that, in this case, is rising, charging the battery to achieve the desired final value of SOC.

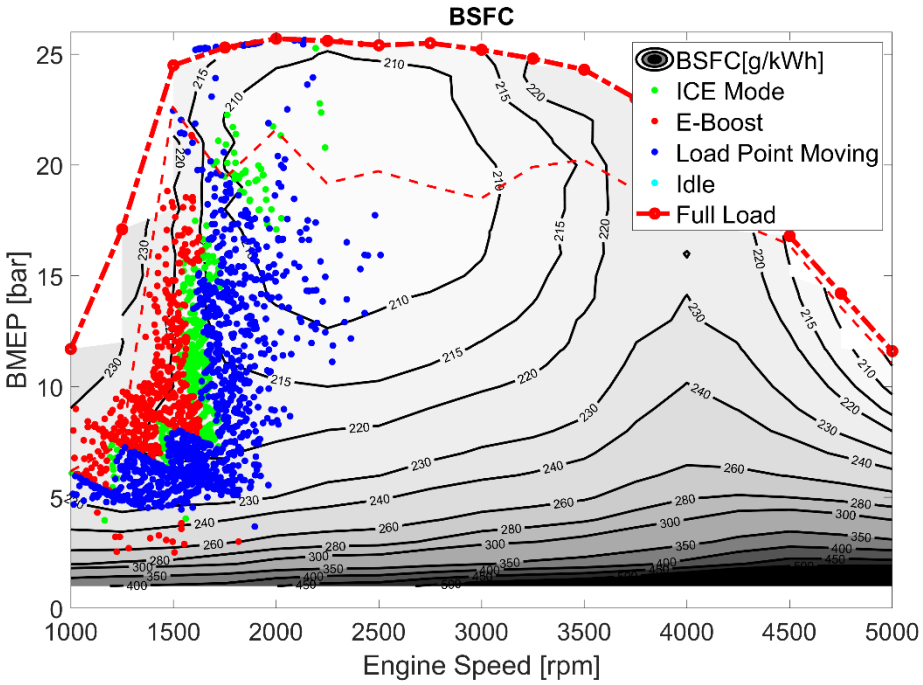

Figure 4.12: Engine operating points on BSFC map

One of the goals of a hybrid powertrain is the possibility of moving the engine operating point on more efficient zones, exploiting the electric machine, aspect that is shown in Figure 4.12, where the engine operating points are plots on the engine Brake-Specific Fuel Consumption (BSFC) map. The operating points tend to be as close as possible to Optimal Operating Line (dashed red line), which represents the most efficient operating point at fixed BMEP and engine speed.

This is done because one of the problems of the internal combustion engine is the low efficiency, especially at low load operation, and, thanks to a hybrid powertrain architecture, is possible to better exploit the internal combustion engine, making it more efficient.

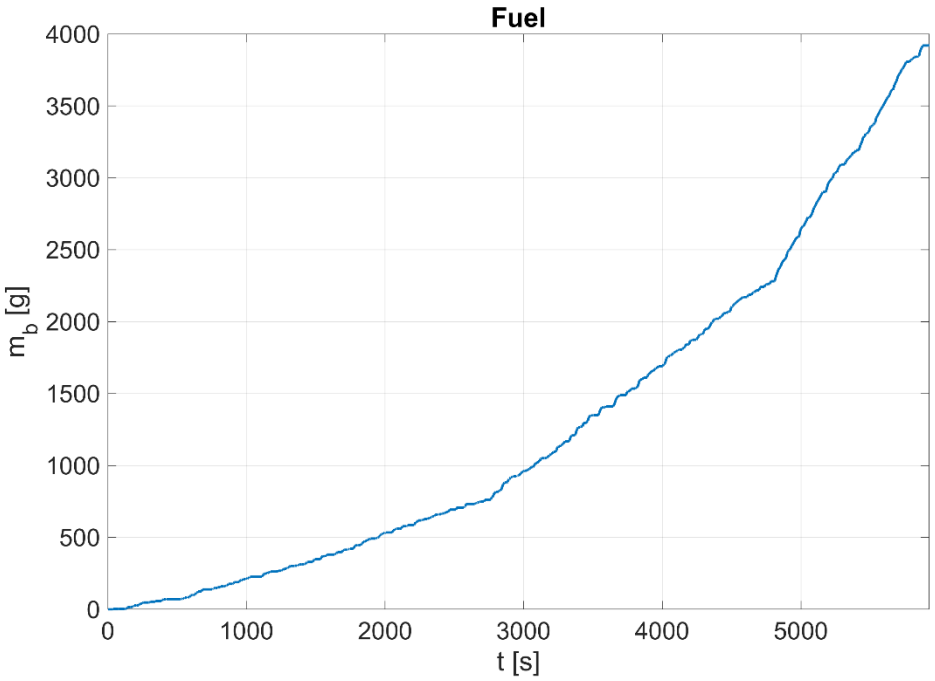

Figure 4.13: Overall fuel consumption

In a charge depleting strategy, the State of Charge of the battery is the most relevant quantity. In fact, it is possible to prove that the optimal solution in terms of fuel consumption is obtained when the SOC, in a SOC-distance plot, almost linearly decreases from the initial value to the final value.

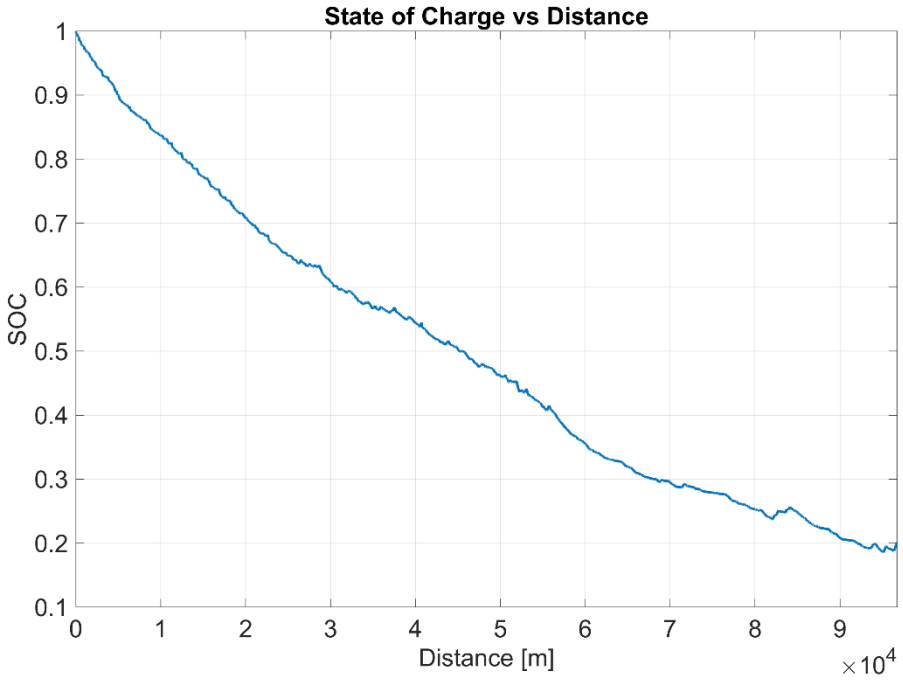

Figure 4.14: State of Charge vs Distance in a charge depleting strategy

From Figure 4.14 this trend is almost reached. The discordance from a perfect linear trend is due to the not continuous model but discretized with a 1 s time step, and to some missing information that makes this DP solution sub-optimal. However, the distance between the optimal solution and the obtained sub-optimal one is very low and it is necessary due to computational limits.

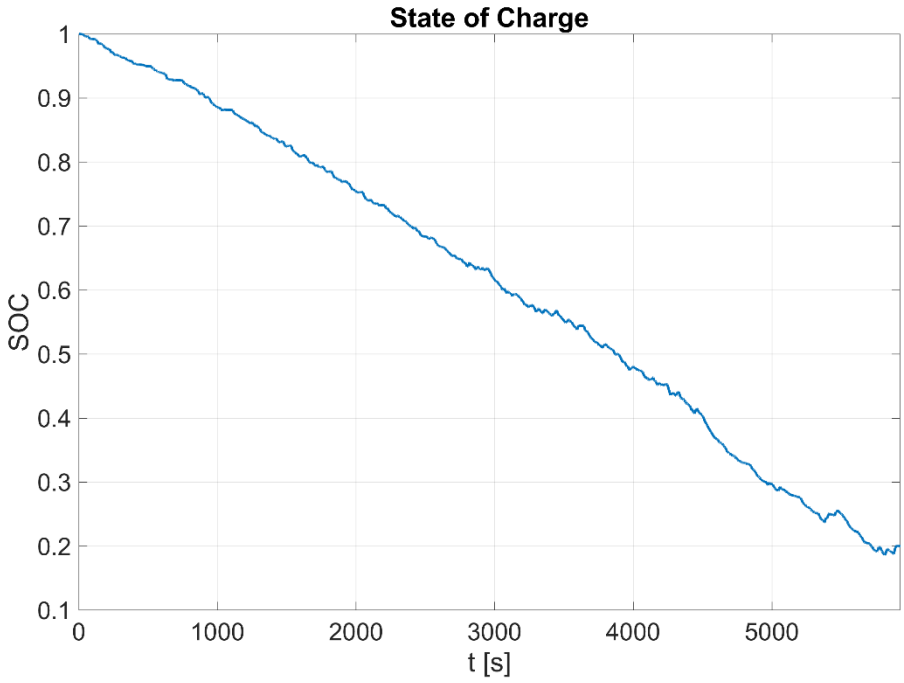

Figure 4.15: State of Charge vs Time in charge depleting

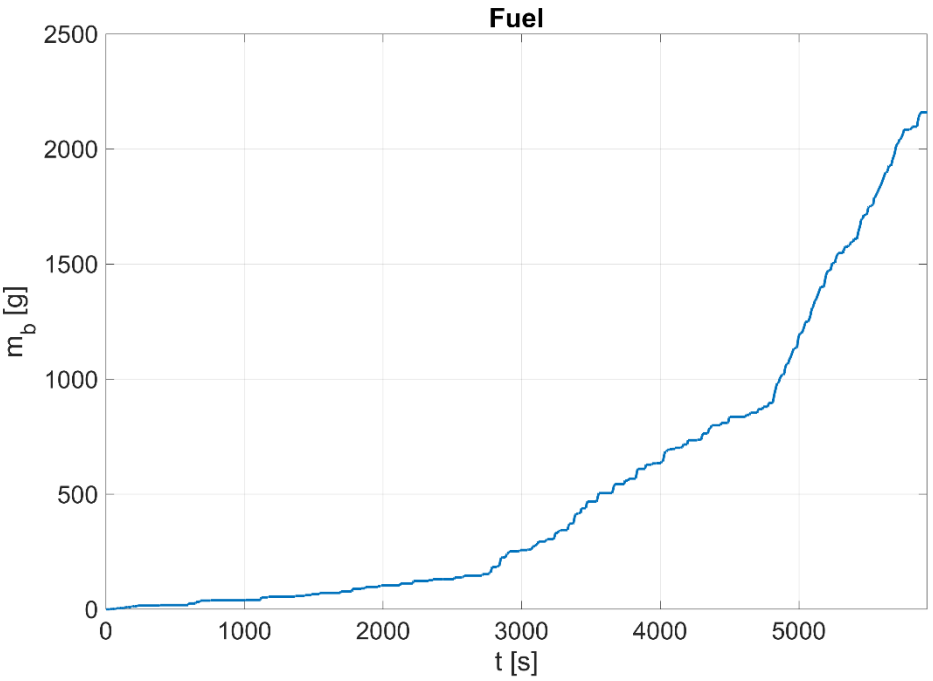

Figure 4.16: Fuel consumption in charge depleting

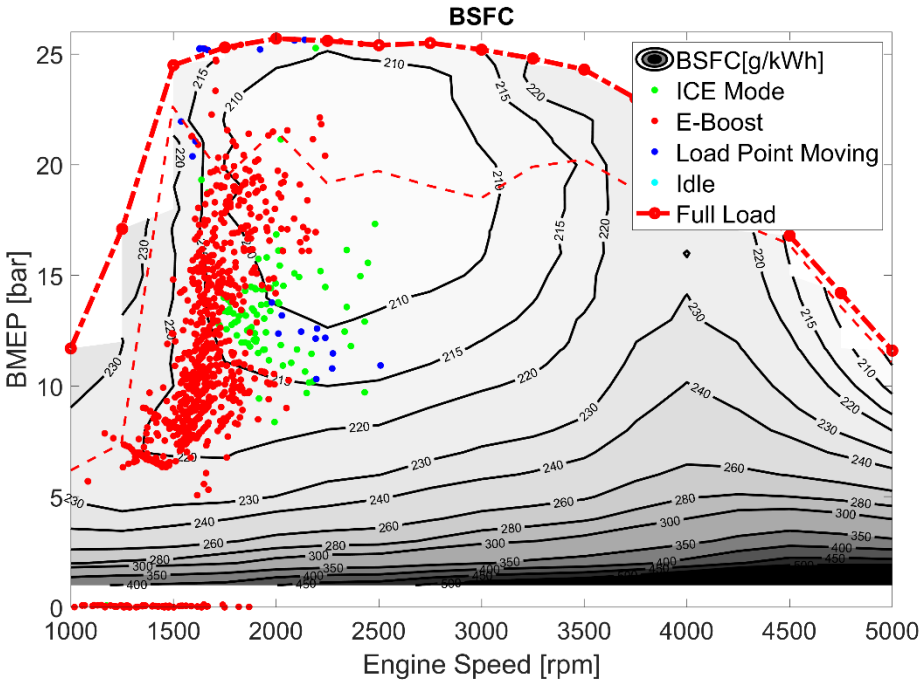

Figure 4.17: Internal combustion engine operating points on BSFC map in charge depleting

4.6.2 Deep Learning-based EMS

The online implemented Energy Management System (EMS) has been designed using two different deep neural networks to control a charge sustaining mode and two deep neural networks to control the charge depleting mode because the management of the battery is quite different between the two

strategies and a single deep neural network could not be able to provide the expected results on both the optimization strategies.

A driving cycle can be seen as a quantity (i.e., vehicle speed) function of time, thus, to make the best decision in terms of power split optimization, some past information are needed, concerning the State of Charge trend or vehicle speed trend to identify the driving pattern (urban, rural or highway) to decides between a battery charge or discharging strategy. The most suitable deep neural network architecture is a Recurrent Neural Network (RNN), with Long Short-Term Memory (LSTM) layers, described in deep in section 3.6.

In deep learning projects, the most important part is related to data pre-processing because, without appropriate pre-processed data, it is impossible to achieve good results from the model. Possible problems are related to the over or underfitting of the model.

The overfitting problem is detectable by analysing the model performance on the test and train dataset. If test performance is not comparable to train one, there is an overfitting problem. This is due to a high adaptation of the model on train data, without a generalization capacity, underlined by the poor performance on test data, which are data that the model has not seen and are not used during the training phase. In fact, during the training, if data have not been correctly pre-processed or the training dataset is not wide enough to catch all possible situations, the model can learn only the noise present in the data and not the general trend of the solution.

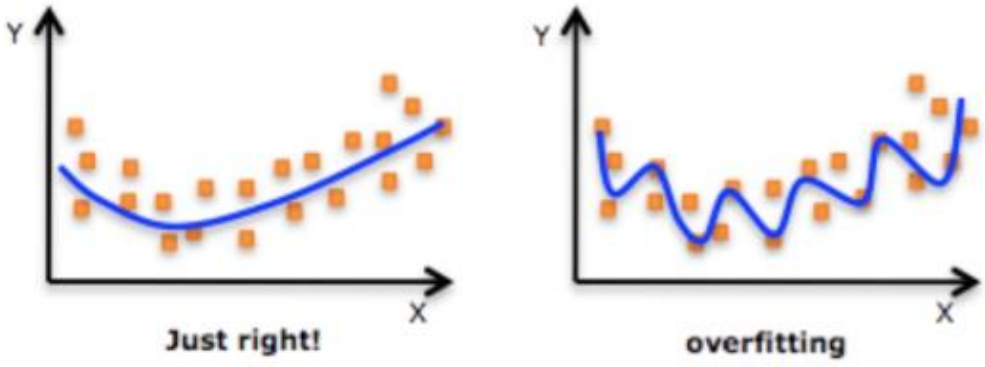

Figure 4.18: Difference between an overfitted model and a correct model[45]

The underfitting problem is due to the wrong selection of the features that describe the problem. In fact, if not all the needed features are selected, during the training phase, the performance is very bad, because the model does not have the right information to relate the output value to the input quantities.

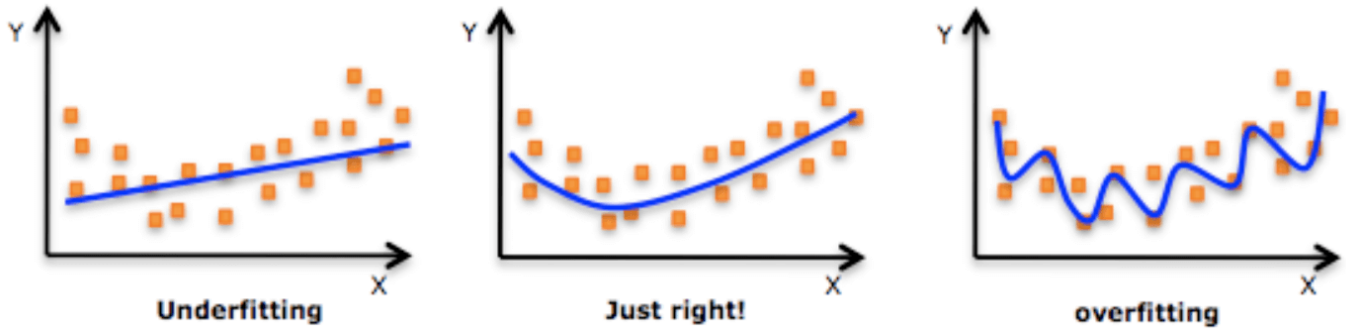

Figure 4.19: Difference between an underfitted model, an overfitted model and a correct model[45]

Concerning how the EMS has been designed, the first LSTM neural networks predict the engine state at each time step. This output is then passed as input to the second LSTM neural network, used to predict the BMEP value of the internal combustion engine at each time step, as shown in Figure 4.20. The features selected to represents the relevant quantities for the engine state prediction in charge sustaining mode are the following:

- **Vehicle speed**: mainly used as passed information, to recognize on which driving condition the vehicle is operating at each time step.
- **Vehicle acceleration**: has the same task as the vehicle speed.
- **Powertrain power demand**: is the power signal at the inlet of the gearbox, that is the power that, coming from the wheel, the powertrain must satisfy. It is probably the most relevant feature because, depending on its value, a direct decision on the engine state could be taken.
- **State of Charge**: the network can take decisions based on the level of SOC because, especially in charge sustaining operation, values too high or too low of SOC force the internal combustion engine to be turned off or turned on to discharge or recharge the battery. It should represent a sort of internal state of the model.

The features used as input for the BMEP neural network in both charge sustaining and charge depleting mode are the following.

- **Vehicle speed**: same considerations did for the first neural network.
- **Vehicle acceleration**: same considerations did for the first neural network.
- **Powertrain power demand**: same considerations did for the first neural network.
- **Gearbox speed**: this feature has the task to virtually fix one quantity on the BSFC plot. This is useful because one of the targets of the parallel mode is to move the engine operating point as close as possible to the optimal operating line and, if the engine speed is foxed, the only variable is the BMEP.

- **Engine State**: used to recognize if the engine is turned on or off and outputs a null value if the engine is not working.

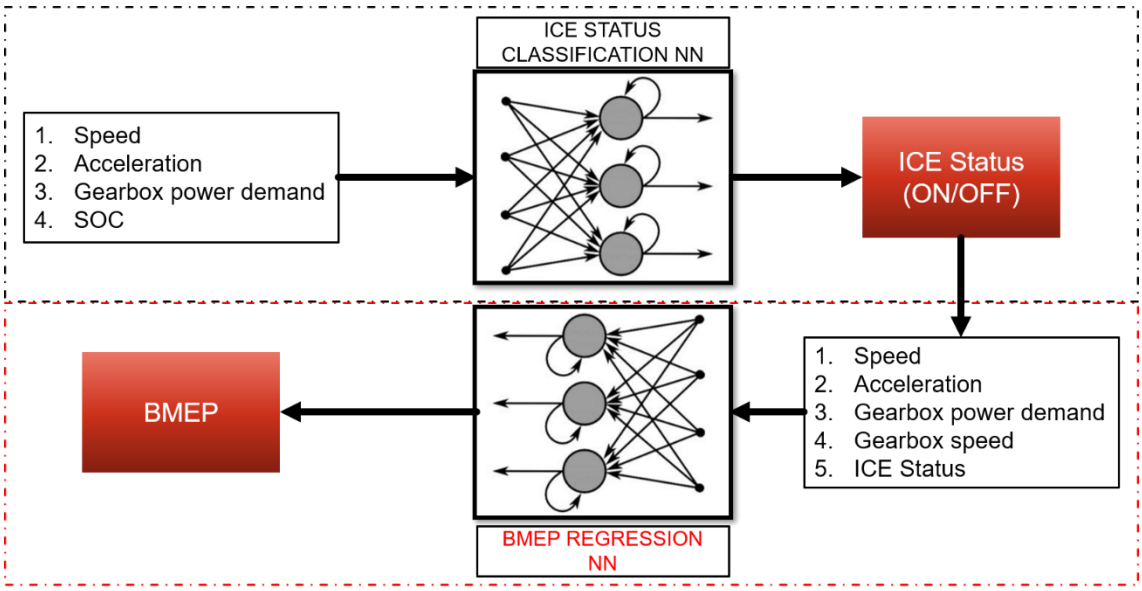

Figure 4.20: Energy Management System based on two LSTM NN architecture representation

The engine state prediction neural network used in charge depleting exploit some future information, theoretically available through V2X connectivity, such as the estimated driving cycle time and distance, necessary to obtain the correct SOC trend, having the following features:

- **Time percentage**: $\frac{Time}{Time_{TOT}}$ to provide time-based information about how long the driving cycle last.
- **Distance**: travelled distance from the vehicle at each time step.
- **Distance percentage**: $\frac{Distance}{Distance_{TOT}}$ to provide distance-based information about how many meters the vehicle must run until the end of the cycle.
- **Vehicle speed**: same considerations did for the charge sustaining neural network.
- **Vehicle acceleration**: same considerations did for the charge sustaining neural network.
- **Powertrain power demand**: same considerations did for the charge sustaining neural network.
- **State of Charge**: same considerations did for the charge sustaining neural network.

During the training phase, to maximize the performance of the network both on train and test datasets, some additional considerations must be taken.

First of all, all the input features to the network must have comparable magnitudes. In this case, without any operation, this condition is not respected because features magnitude space from 10^{-1} of SOC to 10^{4} of power demand. One of the possible operations that can be done is called

standardization of the data. It consists of rescaling each sequence of features with zero mean and unitary standard deviation, standardizing the data in the following way:

$$\boldsymbol{y} = \frac{\boldsymbol{x} - \frac{\sum_{i=1}^{N} x_i}{N}}{\sigma(\boldsymbol{x})} \tag{4.13}$$

Where:

- $\boldsymbol{y}$ is the rescaled data vector.
- $\boldsymbol{x}$ is the original data vector.
- N is the number of data per each feature, considering all the available driving cycles.
- $\sigma(\boldsymbol{x})$ is the standard deviation of the not scaled data.

Data standardization has been applied to each input feature for both the used neural network.
Concerning the output of the regressive neural network, it is important to consider the output label in a narrow range that can space from 0 to 1 or -1 to 1. All these data operations have been made to avoid obtaining too many high weights during the training phase, an aspect that can lead to overfitting the model.
BMEP output values have been rescaled in the following way, to space from 0 to 1:

$$\boldsymbol{BMEP}_{scaled} = \frac{\boldsymbol{BMEP} - \min \boldsymbol{BMEP}}{\max \boldsymbol{BMEP} - \min \boldsymbol{BMEP}} \tag{4.14}$$

A maximum value of 25 bar has been used while 0 bar is the minimum value.

4.6.3 MATLAB Neural Networks Models Implementation and Experiment Manager Optimization

MATLAB environment allows managing deep learning problems, creating deep neural networks exploiting some predefined functions, without explicating manually writing the code to create the network.
This feature allows to speed up and hugely simplify each kind of artificial intelligence model.
When any artificial intelligence problem has to be managed, the first step is to divide the available dataset into three smaller datasets, used respectively for training, validating and testing the model.
This split operation is necessary to verify the generalization capacity of the model and detect if the model overfits.

Since the available dataset is composed of 73 driving cycles, it has been split into 57 test driving cycles, 10 validation driving cycles, used to test the model while the training phase is in progress and 6 test driving cycles to evaluate the overall performance of the proposed neural network-based EMS. An additional split of the above-mentioned driving cycles database has been performed, split them into sequence long 120 s. This operation has been done because LSTM neural network can have some problems managing too long sequences and, since 120 s is enough to recognize the actual driving conditions, these sequences have been used to train and validate the model.

A future sensitivity analysis on this parameter can be done to evaluate its influence on the performance of the model.

Regarding the architecture of the two RNN is structured as follow:

1. **Engine State classification LSTM neural network in charge sustaining**:
    - Sequence Input Layer: a layer that manages inputs in the form of sequences of values function of time (4 nodes).
    - LSTM Layers: recurrent layer, described in section 3.6.2. The concatenation of more than one LSTM layer defines the deep of the network (1 layer of 48 nodes).
    - Dropout layer: a layer that is useful to limit the overfitting problem.
    - Fully Connected Layer: characterized by having just 2 nodes, necessary to pass the information to the next layer that has to manage the probability of the input of belonging to each class. The activation function is *sigmoid*.
    - Softmax Layer: compute the probability of the input values belonging to each class of the classification problem (in this case engine turned on/off).
    - Classification Output Layer: outputs the predicted class.

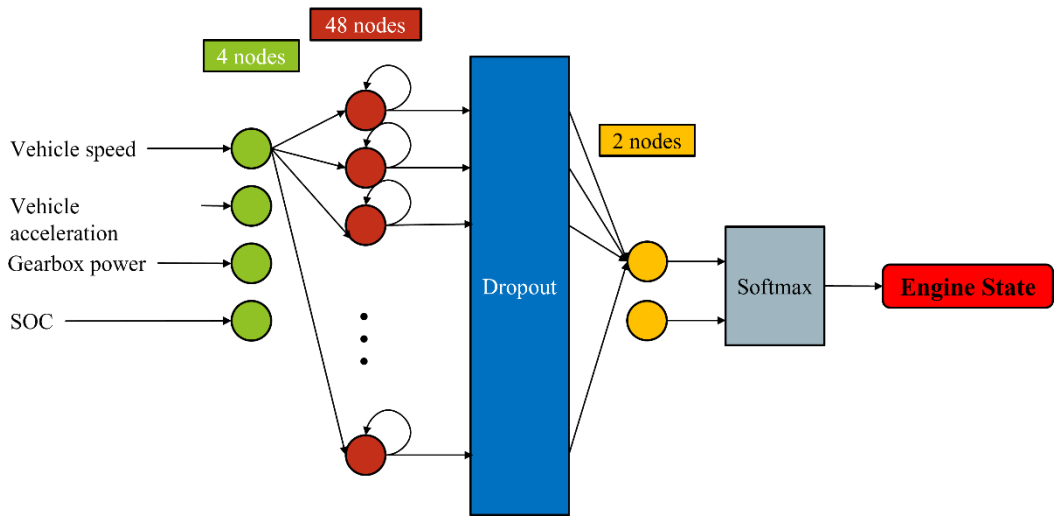

Figure 4.21: Engine state LSTM neural network in charge sustaining representation

2. **BMEP regression LSTM neural network in charge sustaining**:
    - Sequence Input Layer: a layer that manages inputs in the form of sequences of values function of time (5 nodes).
    - Fully Connected Layer: layer with a certain number of nodes, connected with each node of the previous and the next layers, with a *tanh* activation function (54 nodes).
    - LSTM Layers: recurrent layer, described in section 3.6.2. The concatenation of more than one LSTM layer defines the deep of the network (2 layers of 50 and 24 nodes respectively).
    - Dropout layer: a layer that is useful to limit the overfitting problem.
    - Fully Connected Layer: characterized by having just 1 node, necessary to pass the information to the next output layer. The activation function is *sigmoid*.
    - Regression Output Layer: outputs the predicted BMEP value.

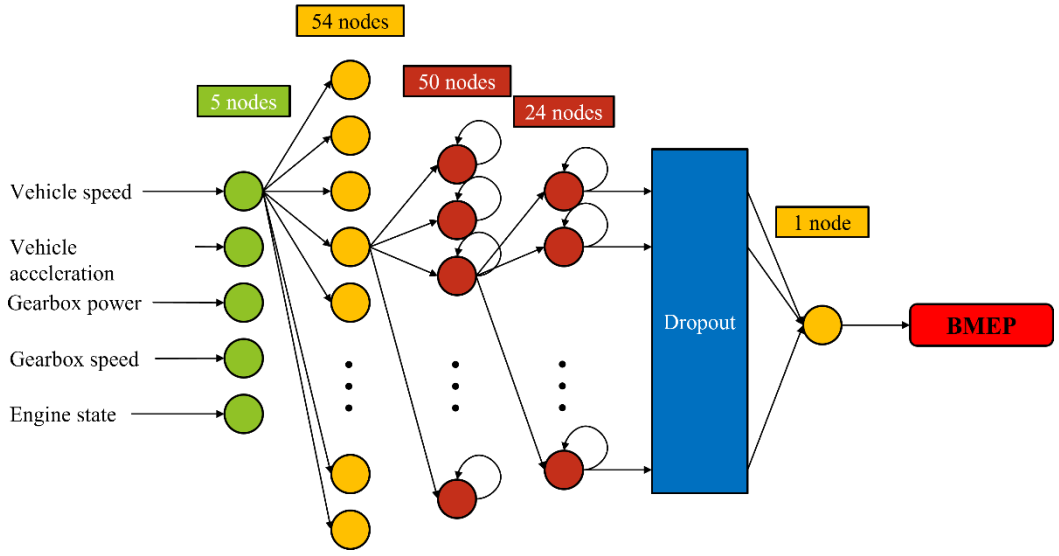

Figure 4.22: BMEP LSTM neural network in charge sustaining representation

3. **Engine State classification LSTM neural network in charge depleting**:
    - Sequence Input Layer: a layer that manages inputs in the form of sequences of values function of time (7 nodes).
    - Fully Connected Layer: layer with a certain number of nodes, connected with each node of the previous and the next layers, with a *tanh* activation function (63 nodes).
    - LSTM Layers: recurrent layer, described in section 3.6.2. The concatenation of more than one LSTM layer defines the deep of the network (1 layer of 37 nodes).

- Dropout layer: a layer that is useful to limit the overfitting problem.
- Fully Connected Layer: characterized by having just 2 nodes, necessary to pass the information to the next layer that has to manage the probability of the input of belonging to each class that characterized the classification problem. The activation function is *sigmoid*.
- Softmax Layer: compute the probability of the input values belonging to each class of the classification problem (in this case engine turned on/off).
- Classification Output Layer: outputs the predicted class.

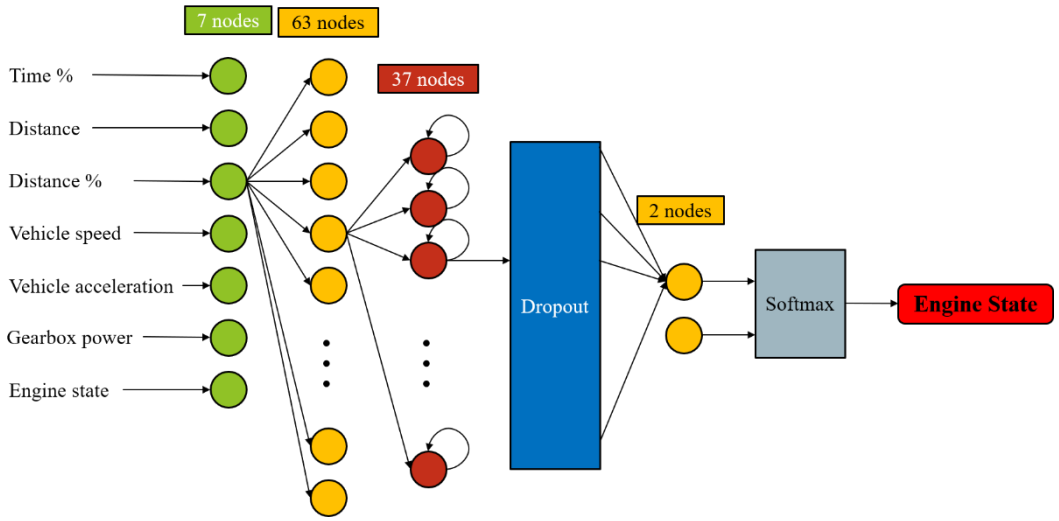

Figure 4.23: Engine state LSTM neural network in charge depleting representations

4. **BMEP regression LSTM neural network in charge sustaining**:
   - Sequence Input Layer: a layer that manages inputs in the form of sequences of values function of time (5 nodes).
   - Fully Connected Layer: layer with a certain number of nodes, connected with each node of the previous and the next layers, with a *tanh* activation function (59 nodes).
   - LSTM Layers: recurrent layer, described in section 3.6.2. The concatenation of more than one LSTM layer defines the deep of the network (1 layer of 75).
   - Dropout layer: a layer that is useful to limit the overfitting problem.
   - Fully Connected Layer: characterized by having just 1 node, necessary to pass the information to the next output layer. The activation function is *sigmoid*.
   - Regression Output Layer: outputs the predicted BMEP value.

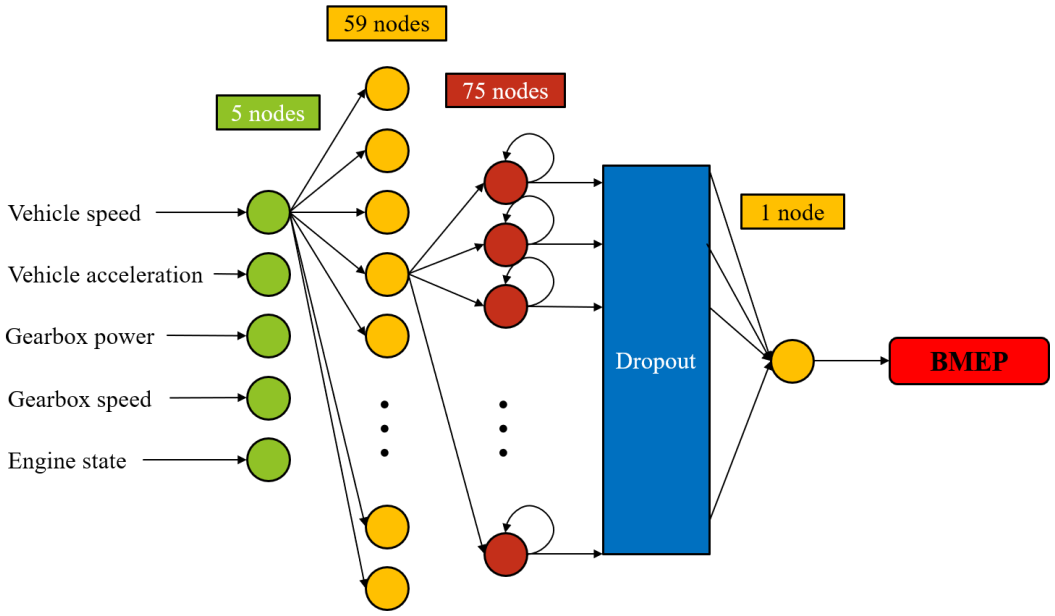

Figure 4.24: BMEP LSTM neural network in charge depleting representation

Another very useful MATLAB tool, called *Experiment Manager*,[46] has been used to perform the hyperparameters optimization. It allows to sweep in a set range each defined hyperparameter (*sweep optimization*) or to perform the *Bayesian optimization*,[19][29][46] a probabilistic based optimization method that allows to speed up the optimization process compared to sweep optimization.
Among the hyperparameters, the most important are the deep of the neural network and the number of nodes of each layer, optimized employing the Bayesian optimization procedure.

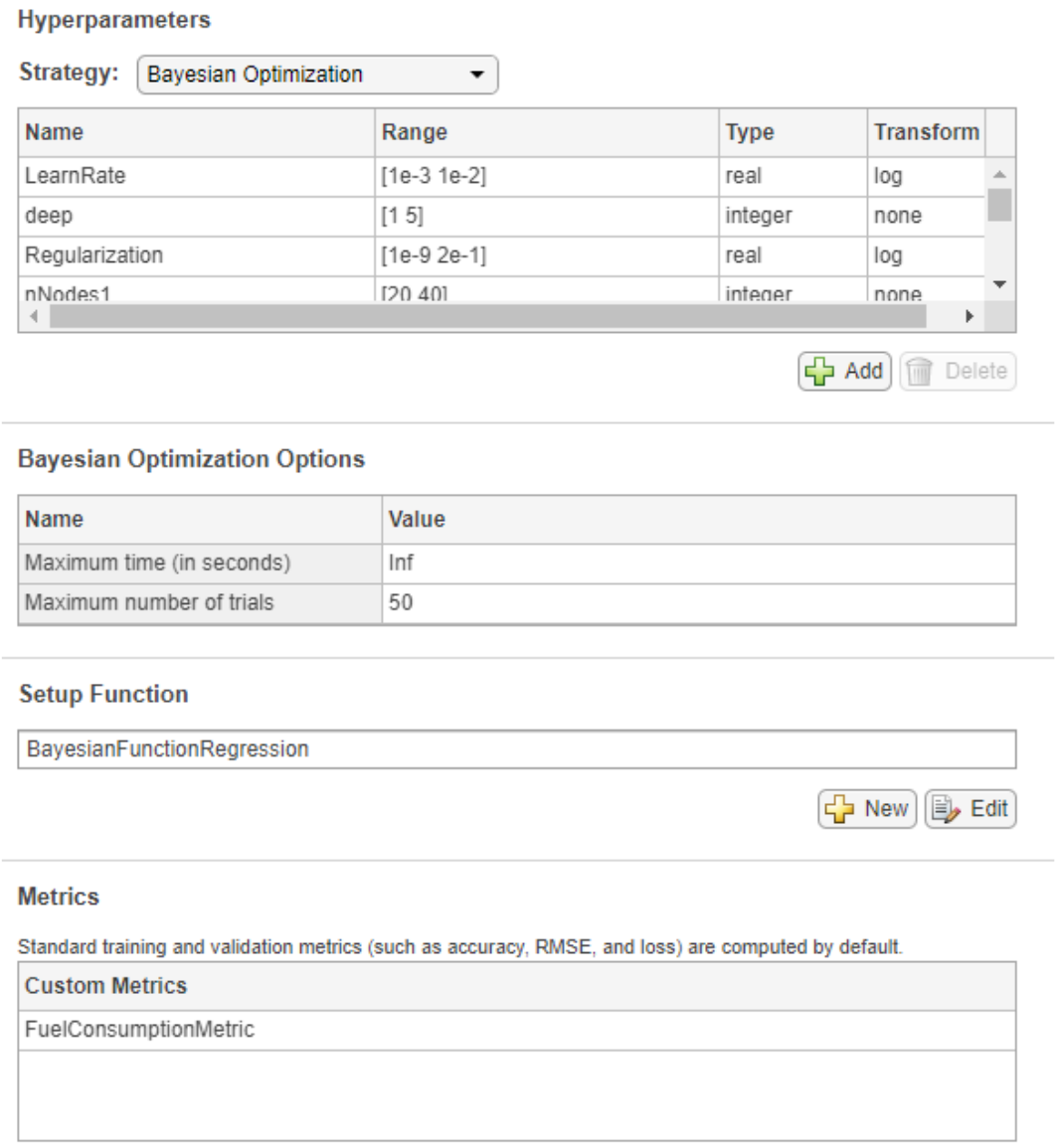

Figure 4.25: MATLAB Experiment Manager graphical user interface

4.6.4 Neural Network-based Energy Management System Performance Evaluation

The most important phase in the development of an artificial intelligence model is to test it with data not used during the training phase, to evaluate the generalization capacity of the model.

The EMS has been tested considering how well perform in terms of fuel consumption and charge sustainability capacity, rather than using standard performance indexes such as Root Mean Square Error (RMSE), using a user-defined cost function.

$$Cost = \frac{1}{n}\sum_{i=1}^{n} \frac{\int_{t_0}^{t_f}\left(\left|\dot{m}_{b\,DP} - \dot{m_b}_{NN}\right|dt\right)}{\int_{t_0}^{t_f} \dot{m}_b dt} + c_1 \frac{\left|SOC_{DP}(t_f) - SOC_{NN}(t_f)\right|}{SOC_{DP}(t_f)} \quad 4.15$$

Where:

- $\dot{m_b}_{DP}$ is the fuel rate coming from dynamic programming optimization.
- $\dot{m_b}_{NN}$ is the fuel rate coming from neural network-based EMS simulation.
- $SOC_{DP}(t_f)$ is the battery State of Charge coming from dynamic programming optimization at the final time step.
- $SOC_{NN}(t_f)$ is the battery State of Charge coming from neural network-based EMS simulation at the final time step.
- n is the number of test driving cycles.
- c_1 is a scale factor to make the fuel consumption evaluation part and charge sustainability part comparable.

It evaluates a sort of relative error that considers both the variables of interest, which are the overall fuel consumption and the network capacity to reach to final SOC target at the end of the driving cycle. Another aspect to consider is that some input features (SOC and engine state) depend on the output of the previous time step. Since they influence the output of both the networks, the error between the correct results coming from DP and the predicted one, will be higher in the test phase than in the train or validation phase that inputs these features as results from the DP simulation, without any error affection from previous time step output.

To speed up the test phase, a backward kinematic vehicle model has been developed, very similar to the dynamic programming vehicle model, deeply analysed in section 4.6.1, developed in Simulink environment to consider the dynamic behaviour of battery SOC during the simulation.

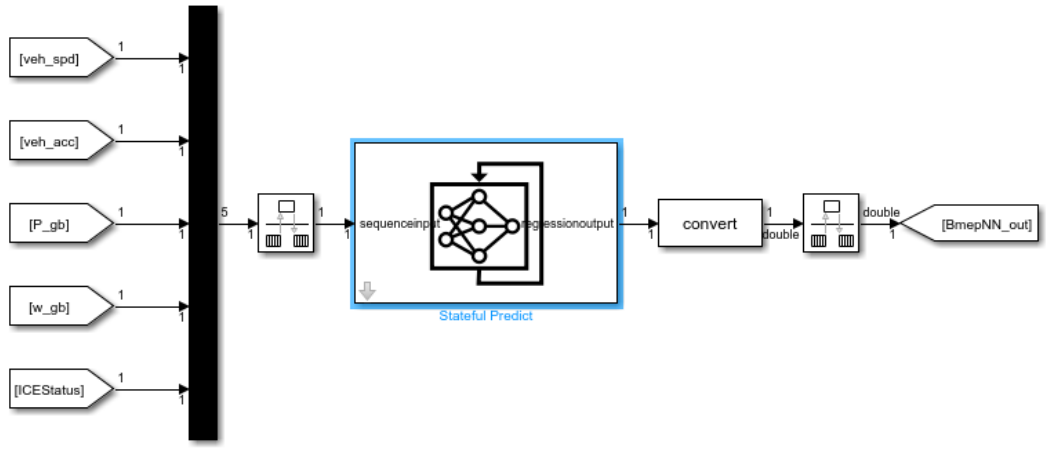

Figure 4.26: BMEP Neural Network implementation in backward kinematic Simulink model to test the pre-trained network. The block S*tateful Predict* loads a MATLAB data file (.mat) with the pre-trained neural network and predicts the output at each time step updating the internal state of the network.

The best NN-based EMS has the following results during the kinematic test phase:

Table 4.3: EMS performance indexes on test driving cycle dataset

Cycle	Accuracy	RMSE	Cost function
Test1	97.69%	0.045	1.91
Test2	95.75%	0.054	6.73
Test3	96.27%	0.050	4.01
Test4	95.31%	0.050	4.20
Test5	95.36%	0.061	2.60
Test6	97.48%	0.044	1.54
Training	97.69%	0.017	-
Validation	96.42%	0.021	-

From the comparison of training performance indexes and test ones, the engine state NN is not affected by any overfitting problem and has a great capacity of predicting the correct engine state at each time step reaching very high accuracy values both in the test and training phase. From the confusion chart of Figure 4.27, on one of the test driving cycles, the great prediction capacity is highlighted. This chart is characterized by the right predictions on the main diagonal, while the non-diagonal terms represent the wrong predictions of the model.

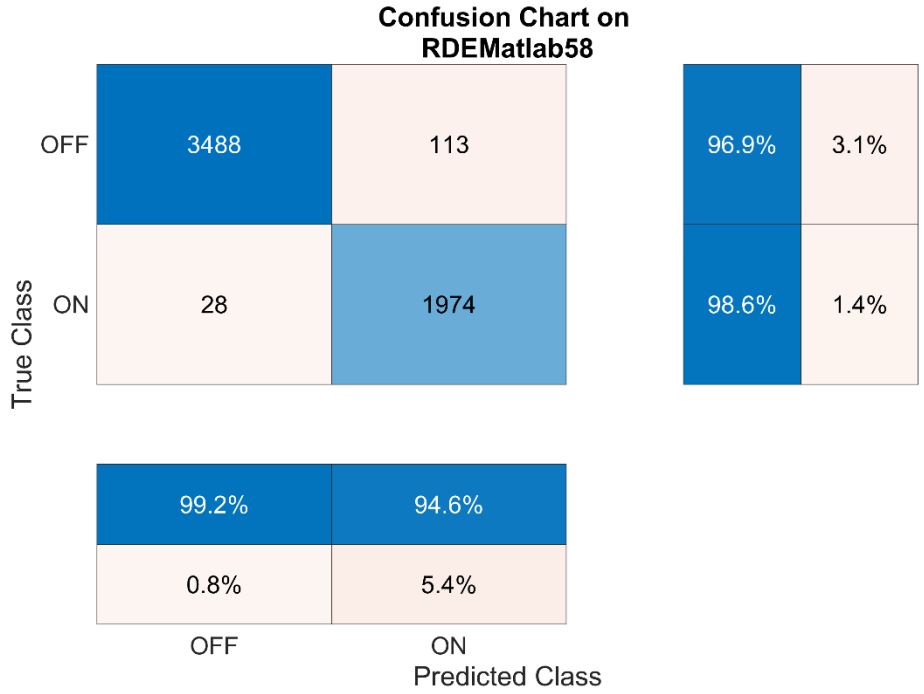

Figure 4.27: Confusion chart on one of the test cycles

RMSE in the test phase is higher than training or validation RMSE but this behaviour is not due to overfitting of the model, since validation and training values are quite close, but to the error chain that starts from a wrong prediction of the first neural network and flows to the output of the second network.

However, using just one neural network (i.e., just BMEP network) instead of the combination of the purposed EMS, leads to more bad results, since the knowledge of the actual engine state is a key feature for the prediction of the BMEP value.

Since the output value of BMEP influences the engine and electric machine power demand, a discrepancy between the correct value of BMEP and the predicted one will lead to an error also in the SOC value that will flow through the entire driving cycle. Since the actual SOC is computed from the previous time step value, a wrong value in the previous time step will lead to a wrong value also in the right time step. This aspect can cause a divergence between the DP SOC value and the NN-based EMS value. However, this aspect is mitigated by the capacity of the engine state neural network to take decisions also from SOC values.

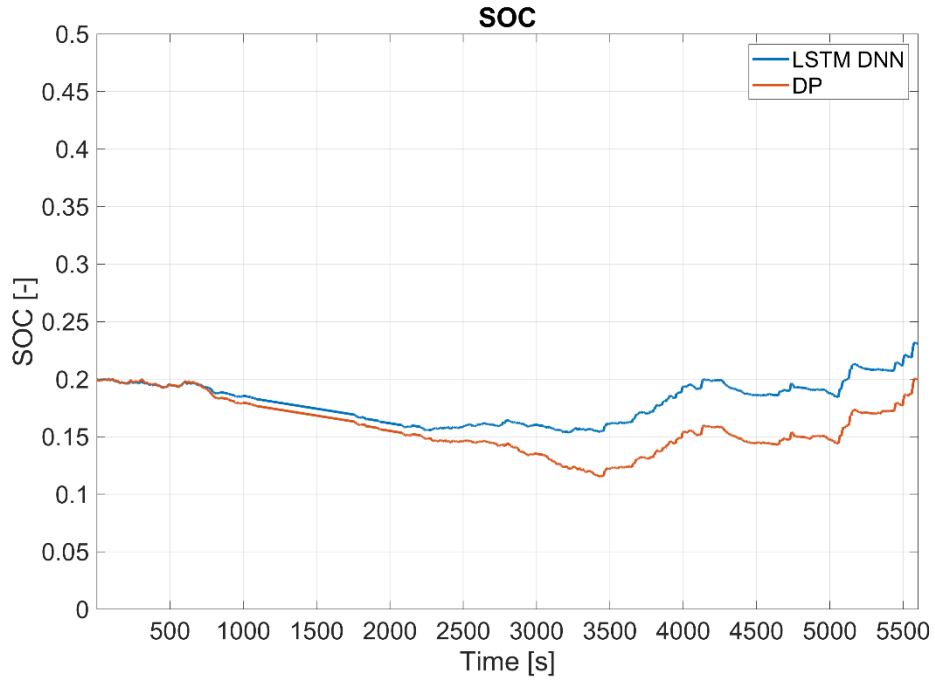

Figure 4.28: State of Charge comparison between DP (correct) and NN-based EMS (predicted) on the test cycle

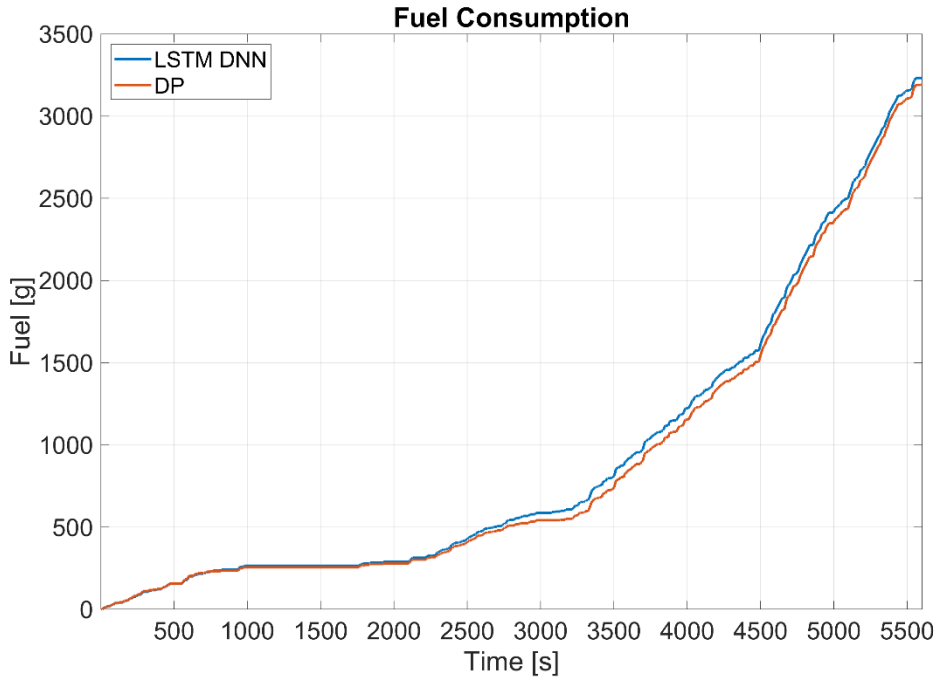

Figure 4.29: Fuel consumption comparison between DP (optimal) and NN-based EMS (sub-optimal)

Figure 4.28 and Figure 4.29 highlights the great behaviour of the NN-based EMS on one of the used test-driving cycles, in terms of both charge sustainability and fuel consumption on the kinematic vehicle model.

The same consideration can be appliable also to the charge depleting trained neural networks.

4.6.5 Dynamic Model

Vehicle fuel consumption is affected by phenomena that cannot be represented in a simple kinematic model. To have more realistic values of fuel consumption and to effectively evaluate the purposed Energy Management System, the NN-based EMS, developed in Simulink, has been integrated on a dynamic vehicle model developed in GT-SUITE environment.

This model has a map-based powertrain integrated into a quasi-static backward dynamic vehicle model, presented in section 4.4.2. It has been validated with a huge quantity of experimental data, with a methodology described in [42], representing a plug-in Hybrid Electric Vehicle with a P2 hybrid powertrain architecture available on the market.

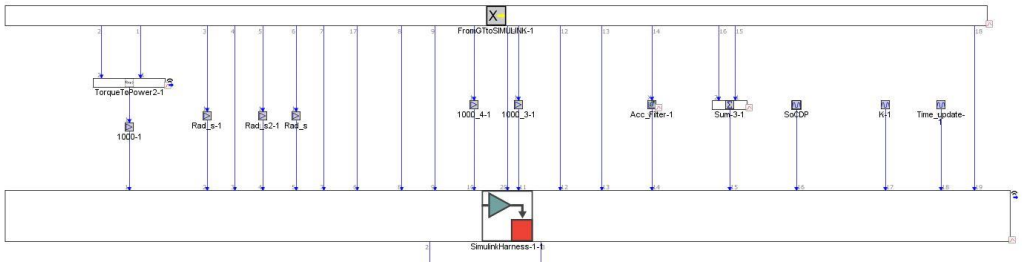

Figure 4.30: Simulink Harness in GT-Suite environment

Concerning the modelling approach, the communication between the two software is possible utilizing *Simulink Harness* block in GT-SUITE (Figure 4.30) and *GT-SUITE Model* block in Simulink library (Figure 4.31). To achieve correct communication, GT-SUITE has been set as slave, with the simulation that runs in Simulink environment.

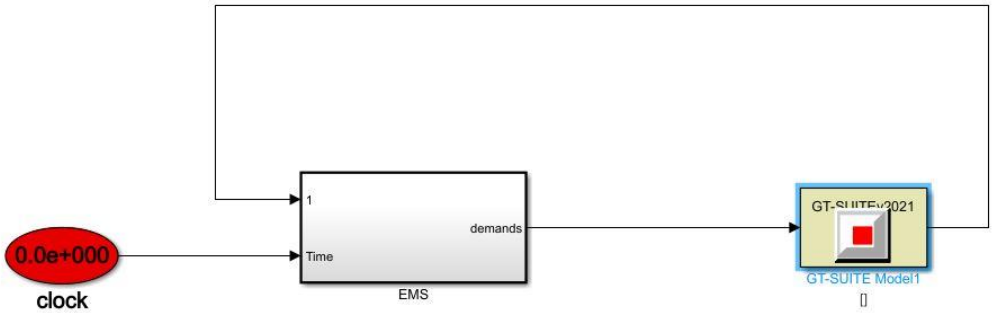

Figure 4.31: GT-SUITE Model block in Simulink environment

5 Results

The proposed Neural Network-based Energy Management System was tested on the test cycles dataset and the results obtained on two of them are presented in this chapter.

Dynamic Programming (DP) optimized model is considered as a benchmark because, as already seen in section 2.2, this optimization method allows to obtain the optimal solution, given *a-priori* known driving cycles. However, due mainly to computational effort reasons, the DP model optimizes the power split considering a kinematic model of the vehicle, while the dynamic model considers all the dynamic behaviour of the vehicle and the powertrain in terms of mechanical response, introducing some aspect that DP model cannot consider.

The consequence is that there is a relevant gap between the benchmark and the effective model in terms of fuel consumption that makes this comparison quite unfair.

A fairer fuel consumption evaluation can be done on the same dynamic vehicle model, equipped with a different power-split control strategy: a Rule-Based strategy, and an ECMS strategy.

The ECMS fuel consumption is a new benchmark to be as close as possible, since, even if it is not an optimal solution, it is an offline optimization procedure.

Concerning the SOC trend, the DP signal can be used to evaluate the battery charging/discharging strategy.

The simulation results will be presented for both charge sustaining and charge depleting operations.

5.1 Charge Sustaining

Charge sustaining results are presented on two RDE compliant driving cycles, used for testing the model.

The goal is to minimize fuel consumption by obtaining a charge sustainability strategy, that consists of having the final SOC value as close as possible to the initial value.

The initial value of SOC has been set to 0.2.

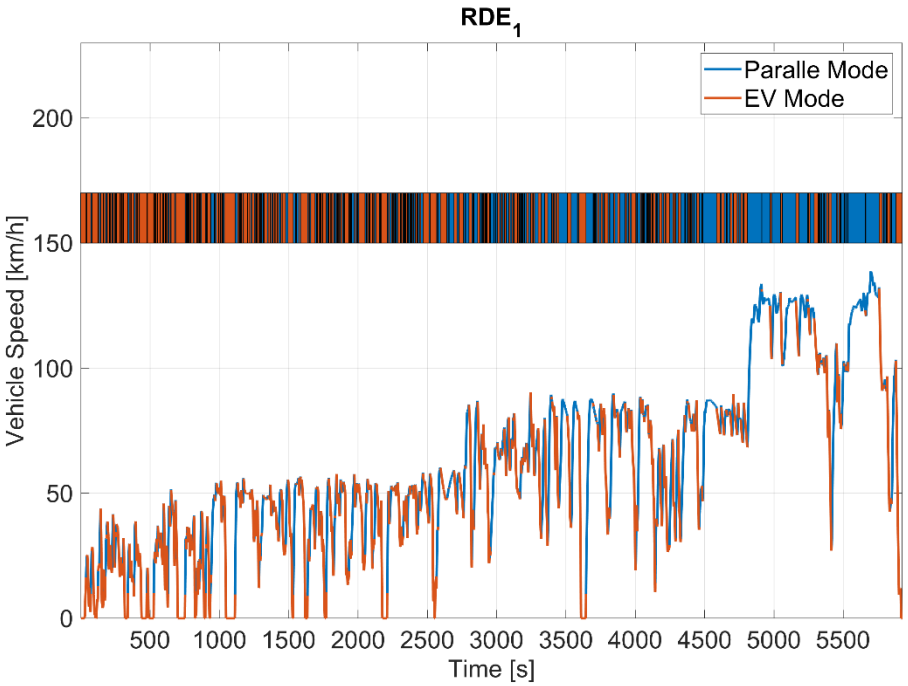

Figure 5.1: Vehicle speed and relative engine state on the first RDE test driving cycle

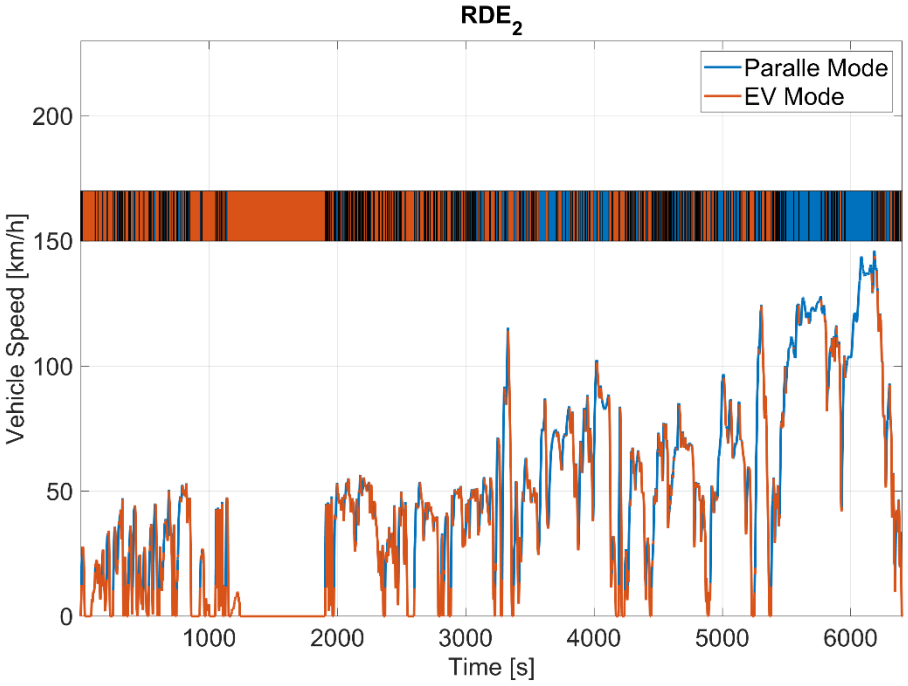

Figure 5.2: Vehicle speed and relative engine state on the second RDE test driving cycle

Figure 5.1 and Figure 5.2 show the powertrain operating modes at each time step. The urban part, which in both cycles is present at the initial parts of the cycle, is performed mainly in pure electric mode, an aspect confirmed from the SOC trend that, in this part is decreasing as shown in Figure 5.3 and Figure 5.4.

The last part of both the cycles represents a highway driving condition, performed mainly in parallel mode, with the goal of recharge the battery to obtain a charge sustaining operation.

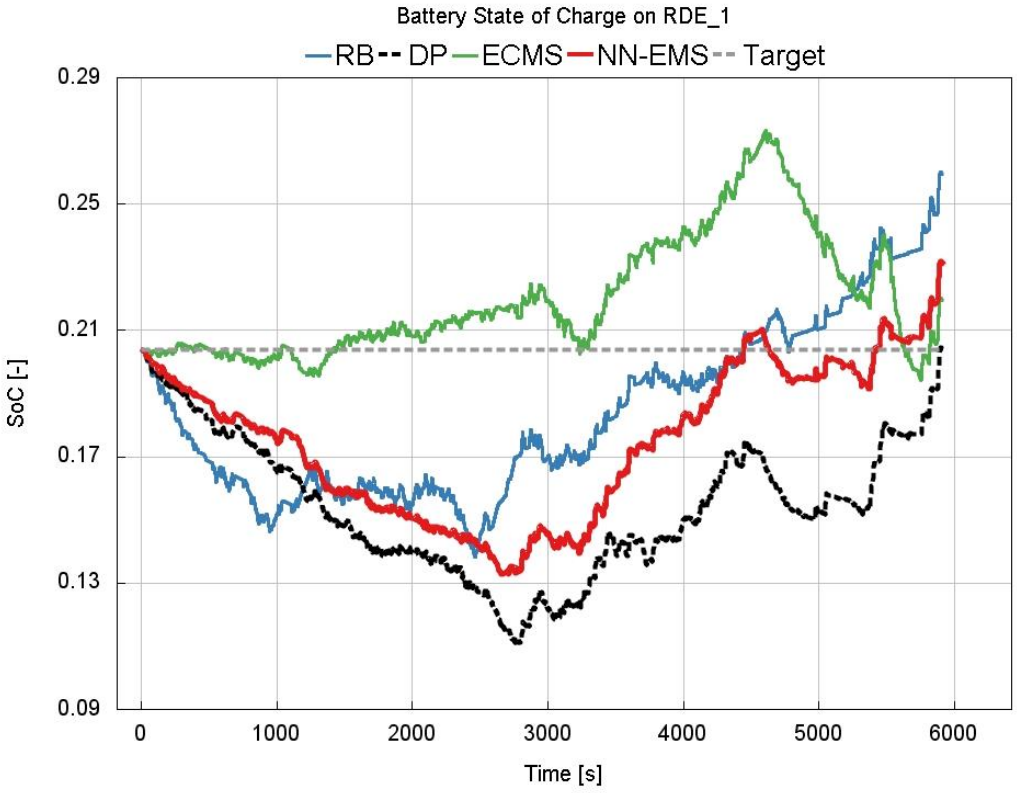

Figure 5.3: SOC comparison between DP, RB, ECMS and NN-based EMS on the first RDE test driving cycle

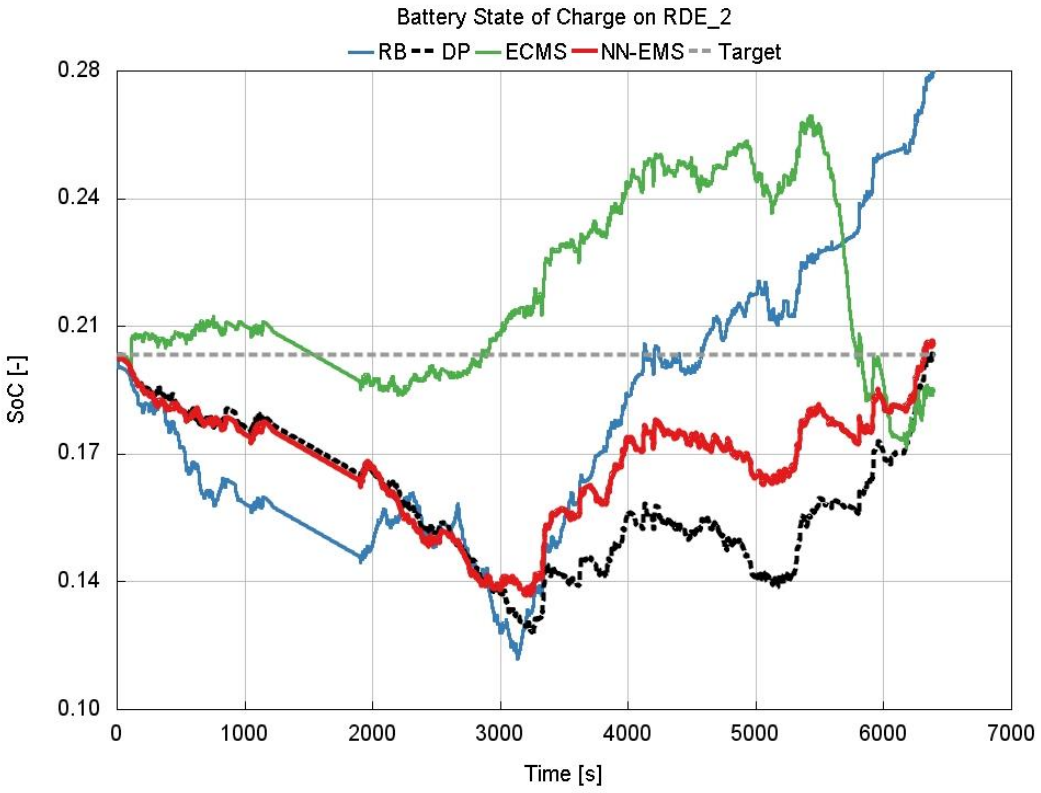

Figure 5.4: SOC comparison between DP, RB, ECMS and NN-based EMS on the second RDE test driving cycle

Figure 5.3 and Figure 5.4 plot the State of Charge trend in both test cycles. The final SOC is very close to the initial value and it has a comparable evolution during the cycle with the DP SOC signal, an aspect that confirms a very good behaviour of the neural network strategy on either the first and the second driving cycle.

Neural Network-based EMS compared with Rule-Based presents a better capacity of charge sustainability, since the final SOC value is closer to the target value. The higher SOC final value of RB-EMS implies higher fuel consumption at the end of the cycle (Figure 5.5).

The comparison with ECMS-EMS presents a completely different evolution of the state of charge with a higher value of NN-based EMS than the ECMS-EMS one. However, as shown in Figure 5.5 and Figure 5.6, despite a higher final SOC value, NN-based EMS reaches lower fuel consumption than ECMS-EMS. This behaviour underlines more efficient exploitation of the internal combustion engine.

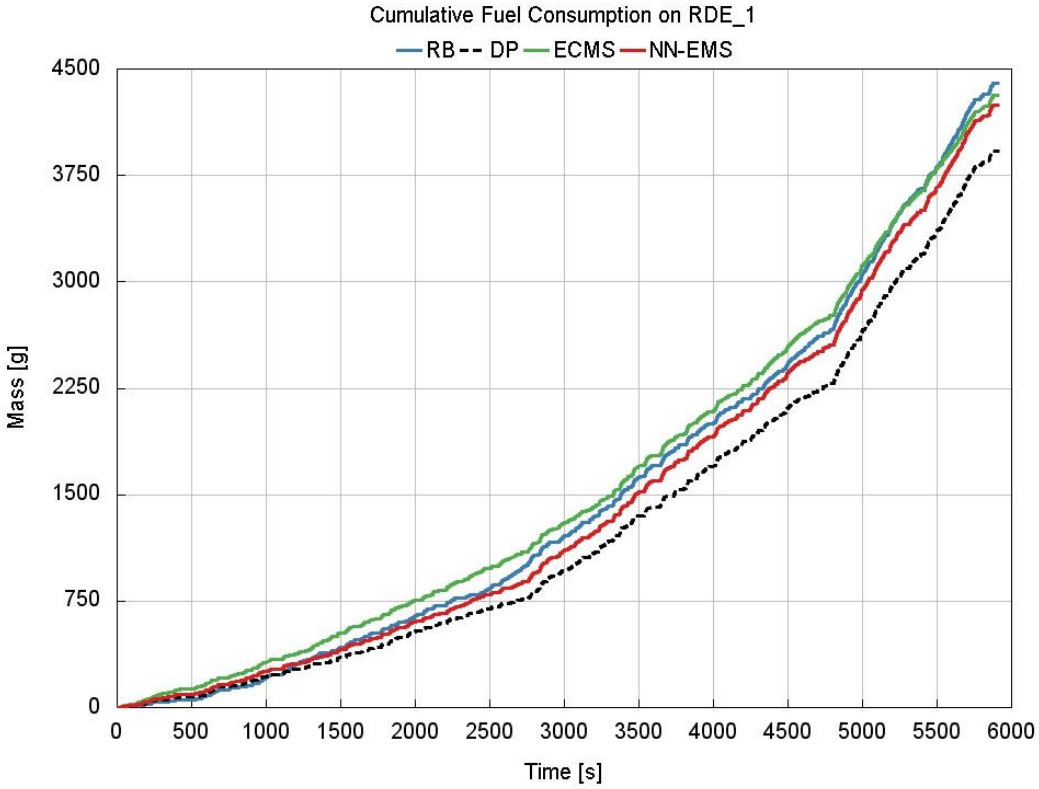

Figure 5.5: Fuel consumption comparison between RB, ECMS, DP and NN-based EMS on the first RDE test driving cycle

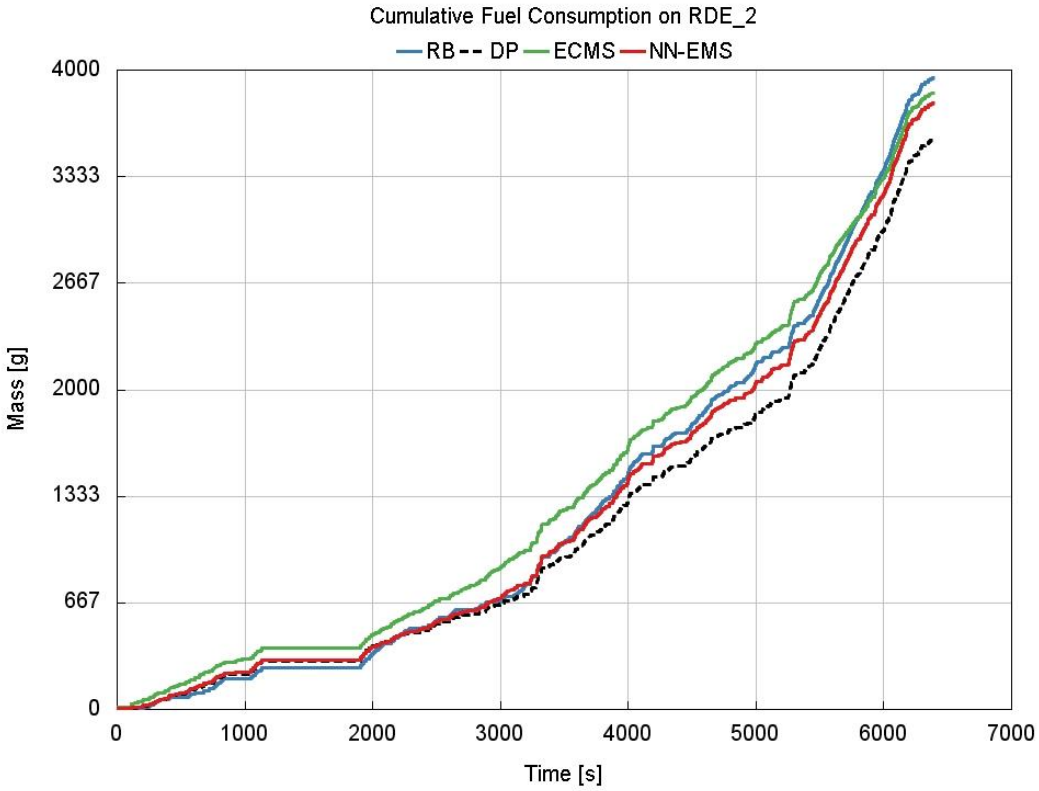

Figure 5.6: Fuel consumption comparison between RB, ECMS, DP and NN-based EMS on the second RDE test driving cycle

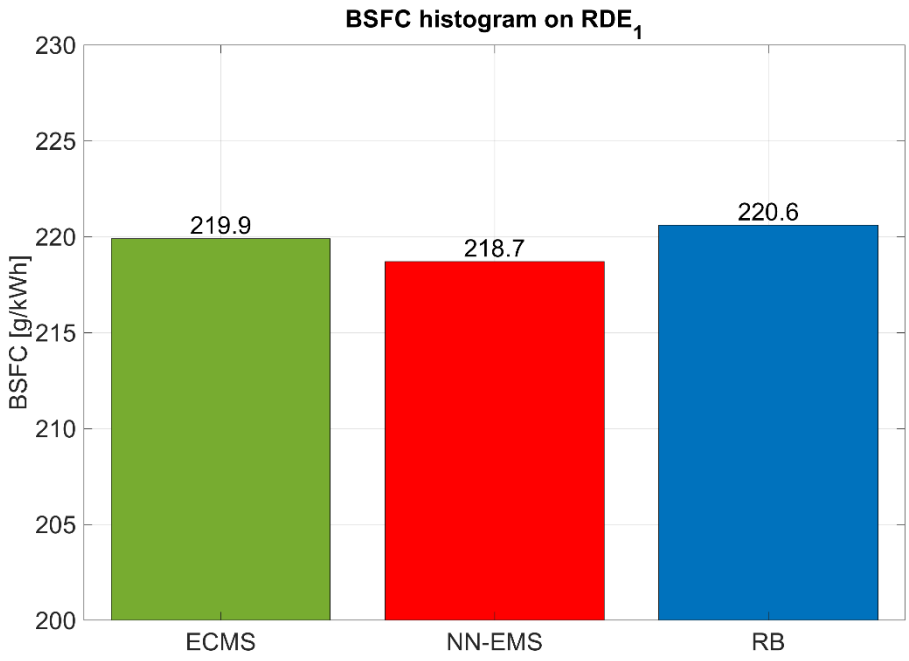

Figure 5.7: BSFC comparison between the three selected EMS on the first test driving cycle

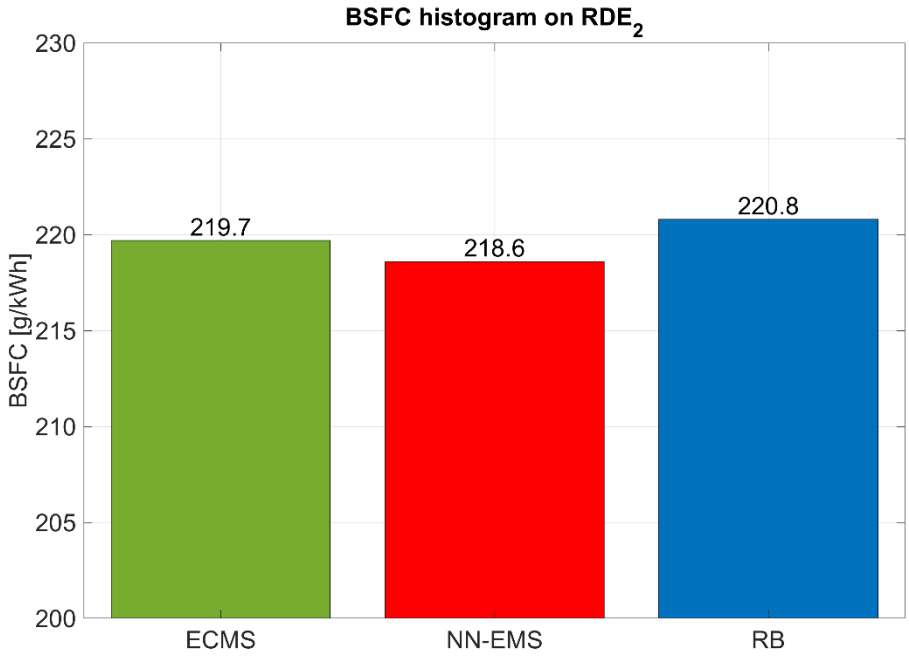

Figure 5.8: BSFC comparison between the three selected EMS on the second test driving cycle

From Figure 5.7 and Figure 5.8 is possible to point out that NN-based EMS is the most efficient solution, from BSFC, a parameter linked to engine efficiency. Smaller BSFC higher the efficiency. In both the figure NN-based EMS has the lowest BSFC.

Table 5.1: BSFC comparison on the two-test driving cycle on the different control strategy presented

Control strategy	BSFC reduction on RDE_1	BSFC reduction on RDE_2
RB	0 %	0 %
ECMS-EMS	-0.32 %	-0.5 %
NN-EMS	-0.87 %	-1.01 %

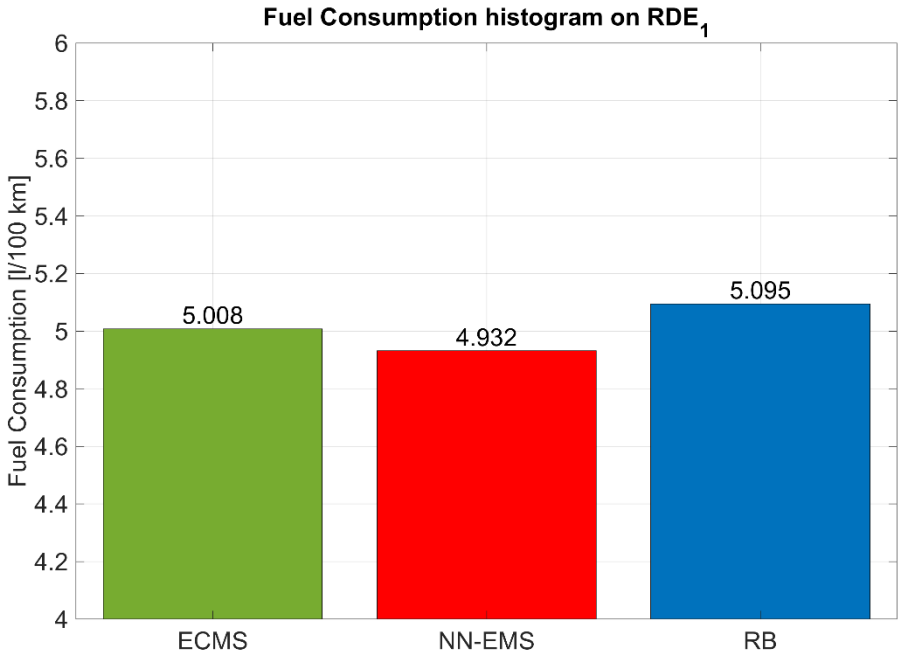

Figure 5.9: Fuel consumption on the three selected EMS on the first test driving cycle

Table 5.2: Fuel consumption reduction comparison on the two-test driving cycle on the different control strategy presented

Control strategy	Fuel consumption reduction on RDE_1	Fuel consumption reduction on RDE_2
RB	0 %	0 %
ECMS-EMS	-1.74 %	-2.37 %
NN-EMS	-3.31 %	-3.95 %

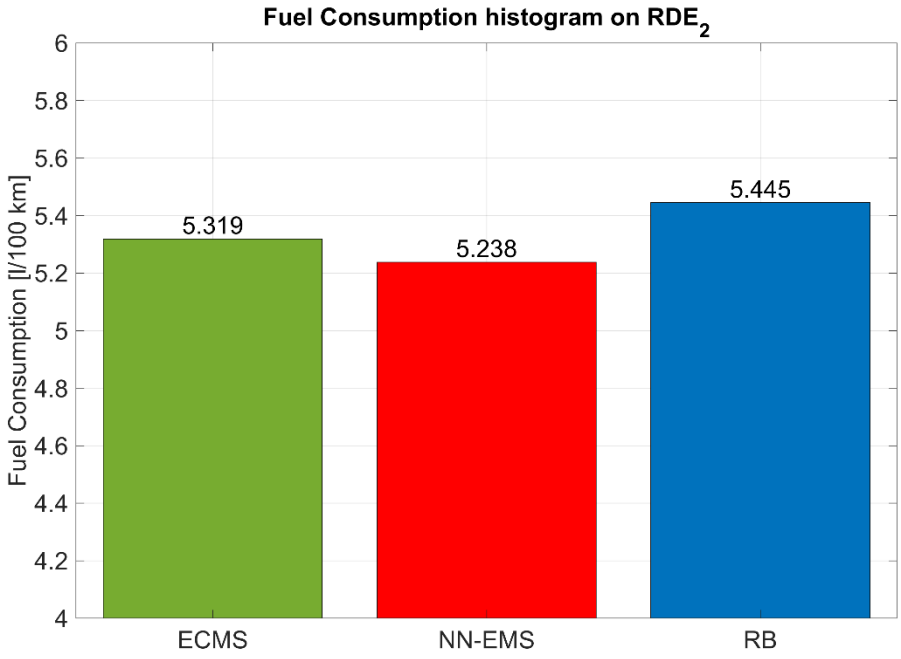

Figure 5.10: Fuel consumption on the three selected EMS on the second test driving cycle

Figure 5.9 and Figure 5.10 represent how the fuel consumption is remarkably decreased in the proposed EMS compared to both the other optimization strategies.

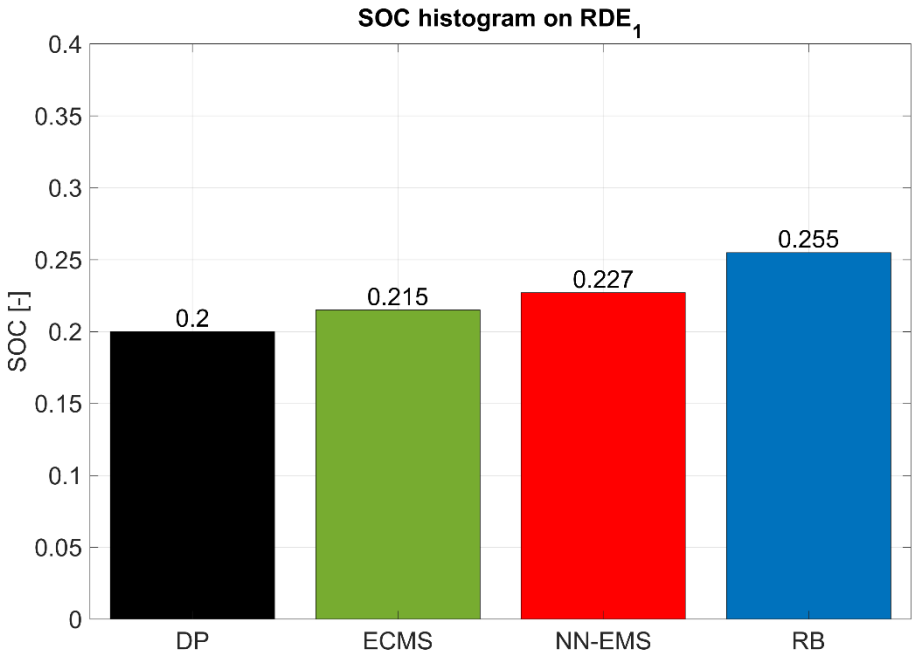

Figure 5.11: SOC comparison with the target value on the first test driving cycle

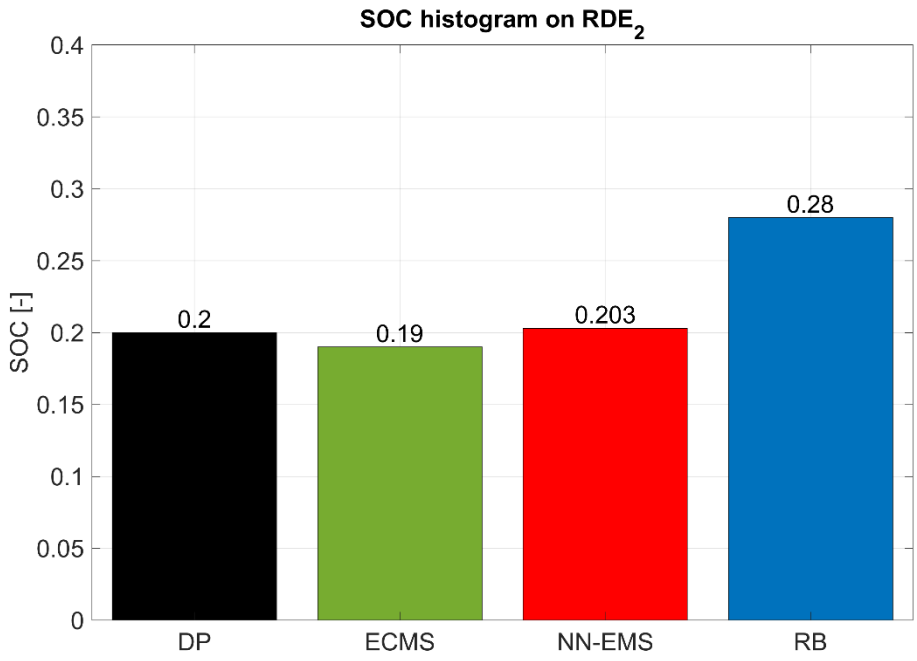

Figure 5.12: SOC comparison with the target value on the second test driving cycle

Table 5.3: Final SOC comparison between the presented control strategy and the target value

Control strategy	SOC distance from the target on RDE_1	SOC distance from the target on RDE_2
DP	0 %	0 %
RB	+21.6 %	+32 %
ECMS-EMS	+6.98 %	-5.26 %
NN-EMS	+9.75 %	+1.1 %

ECMS strategy reaches the closest value to the target in terms of SOC. This result is expectable because it is the only strategy that is pure optimization, while both Rule-Based and NN-based EMS are models calibrated on experimental data the former and on an optimal control strategy, such as the dynamic programming, the latter.

However, NN-EMS can obtain remarkable achievements also in terms of charge sustainability capacity, being very close to the target SOC value.

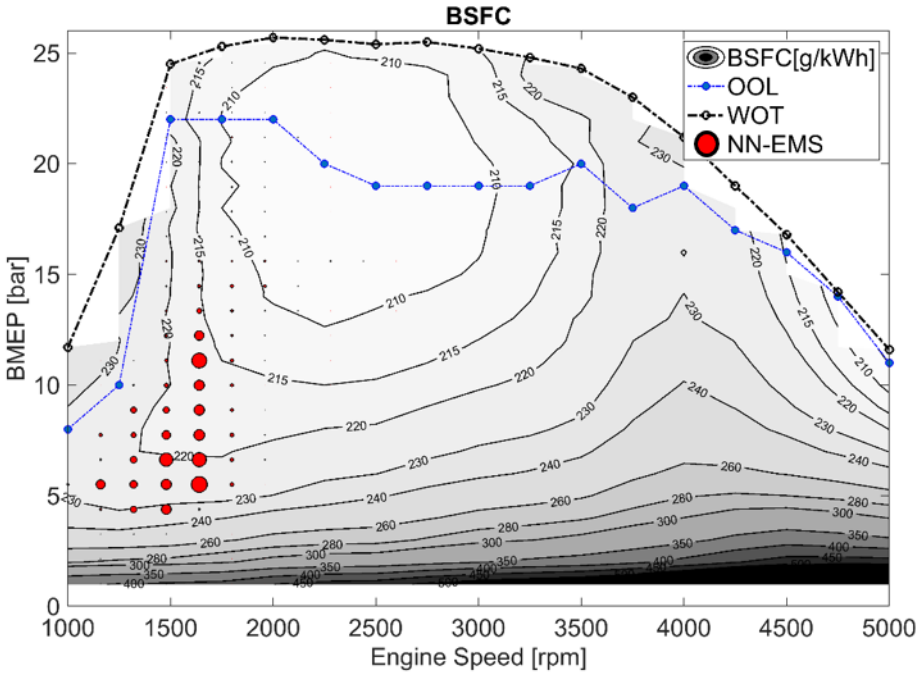

Figure 5.13: Time distribution of the engine operating points of NN-based EMS on the first test driving cycle

In Figure 5.13 and Figure 5.14, the time distribution of engine operating points controlled by NN-based EMS is plot, while in Figure 5.15 and Figure 5.16 the operating points are referred to EMS-ECMS. Bigger the dot, more time the engine has spent at that efficiency region.

The comparison between Figure 5.13 and Figure 5.15 highlights a different time distribution, with the NN-based EMS that presents a more uniform distribution on the BSFC map. This allows exploiting the engine at higher load operating points, characterized by a lower BSFC and so a greater efficiency.

In ECMS-EMS the operating points are mainly focused in the zone that ranges from 5 to 10 bar of BMEP, so at low load.

This is the main reason for the smaller BSFC of NN-based EMS compared to ECMS-EMS shown in Figure 5.7 and Figure 5.8.

The same consideration can be done on the second test driving cycle.

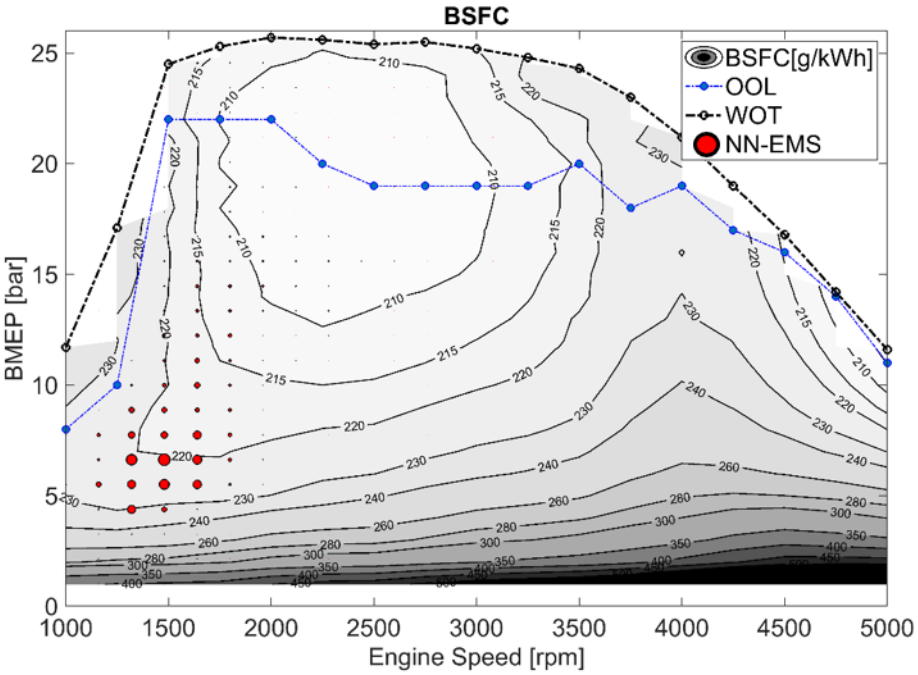

Figure 5.14: Time distribution of the engine operating points of NN-based EMS on the second test driving cycle

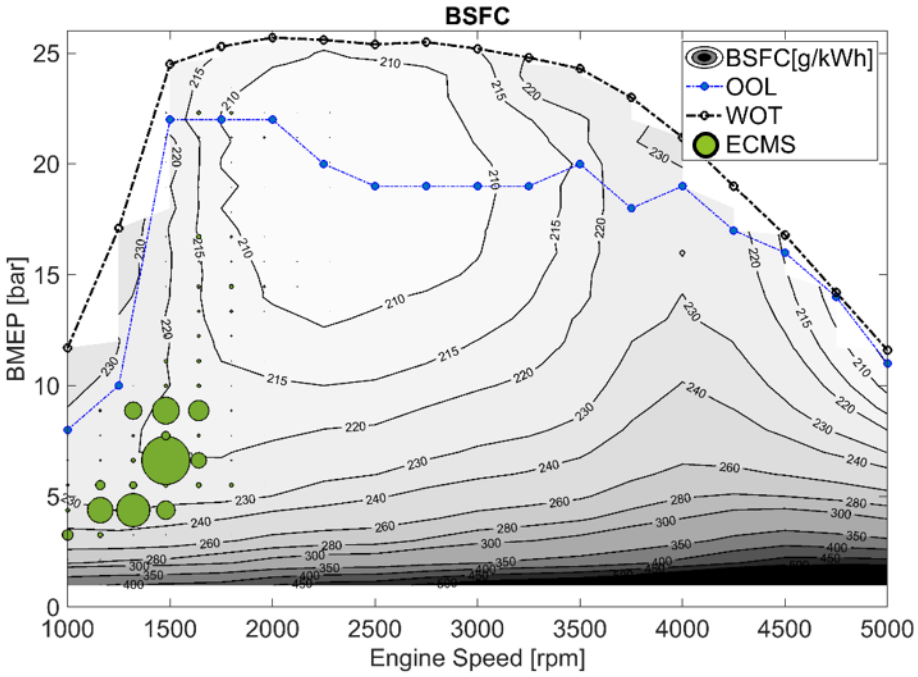

Figure 5.15: Time distribution of the engine operating points EMCS-EMS on the first test driving cycle

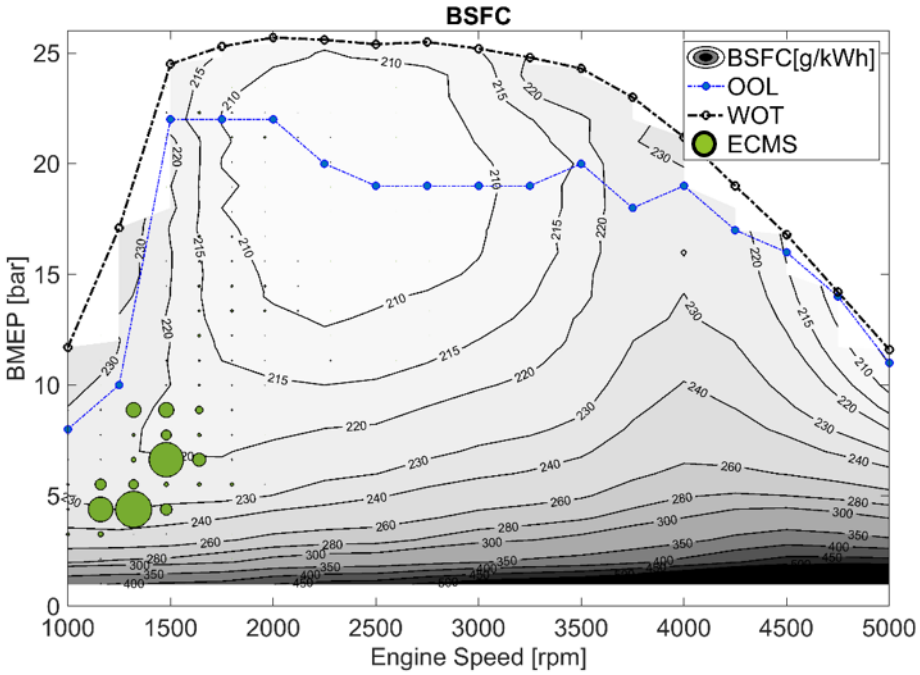

Figure 5.16: Time distribution of the engine operating points EMCS-EMS on the second test driving cycle

5.2 Charge Depleting

A plug-in Hybrid Electric Vehicle is better exploited in a charge depleting mode since its battery has a capacity that allows a remarkable full-electric range.

Ideally, the driving cycle should start with a fully charged (100% SOC) battery and stop with an empty battery (0% SOC). However, the Energy Management System has been designed to reach a final SOC as close as possible to 0.2, to eventually switch to charge sustaining operating mode.

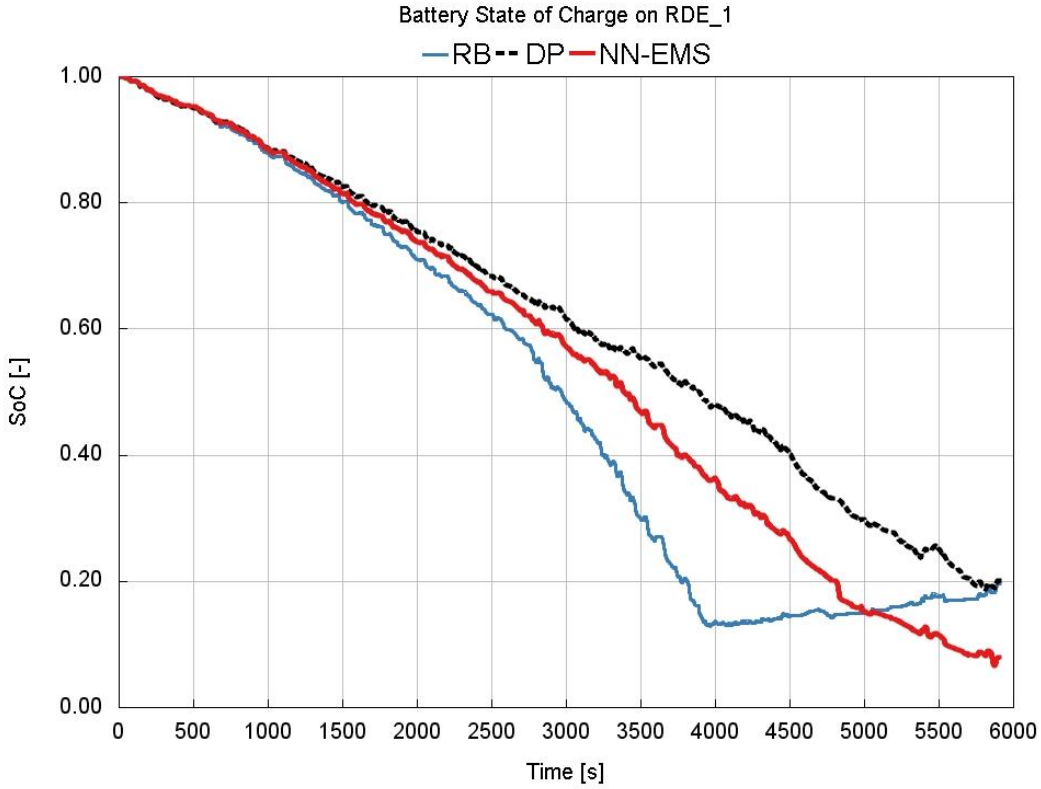

Figure 5.17: Comparison between SOC of NN-based EMS, DP and RB on the first test driving cycle

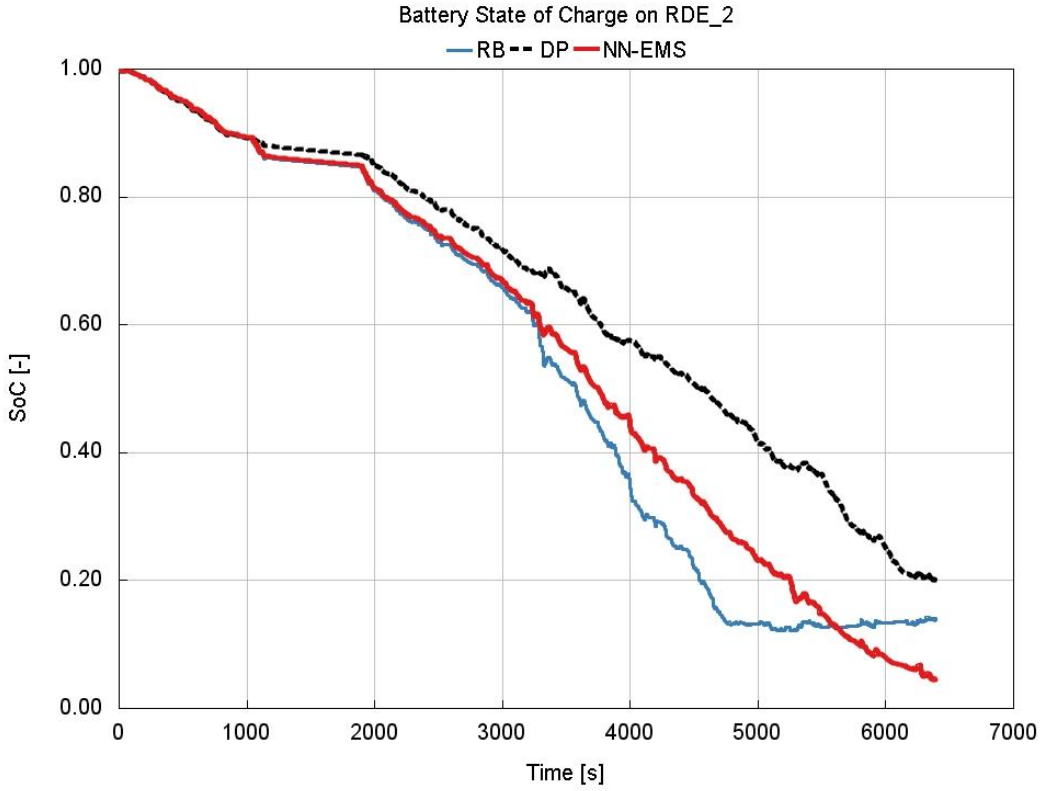

Figure 5.18: Comparison between SOC of NN-based EMS, DP and RB on the second test driving cycle

The SOC of the NN-based EMS has a comparable trend with DP SOC signal, with a higher propensity to discharge the battery, as underlined by the confusion chart (Figure 5.20), because the model is mainly wrong when the engine should be turned on and the prediction is to do not exploit it (2nd row with 80.3% of overall accuracy).

A comparison with how the RB model manages the discharge of the battery (Figure 5.19) highlights a more intelligent way of exploiting the battery energy since NN-based EMS do not distinguish between the highway and urban part of the cycle but try to obtain a SOC trend as much linear as possible, as DP does, with a clear advantage in terms of fuel consumption (Figure 5.21 and Figure 5.22).

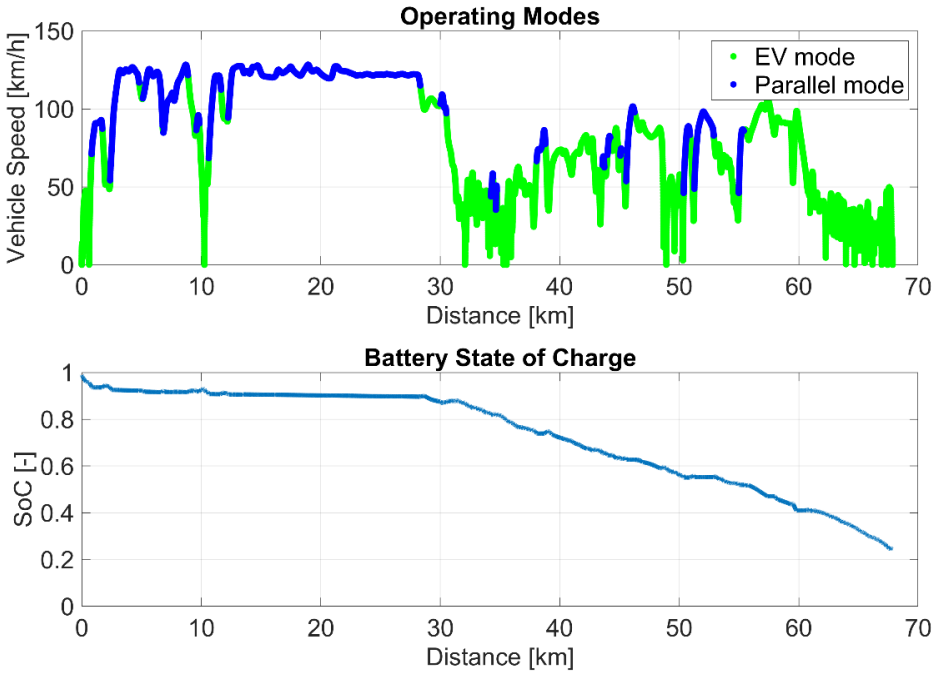

Figure 5.19: RB battery management strategy on RDE driving cycle[42]

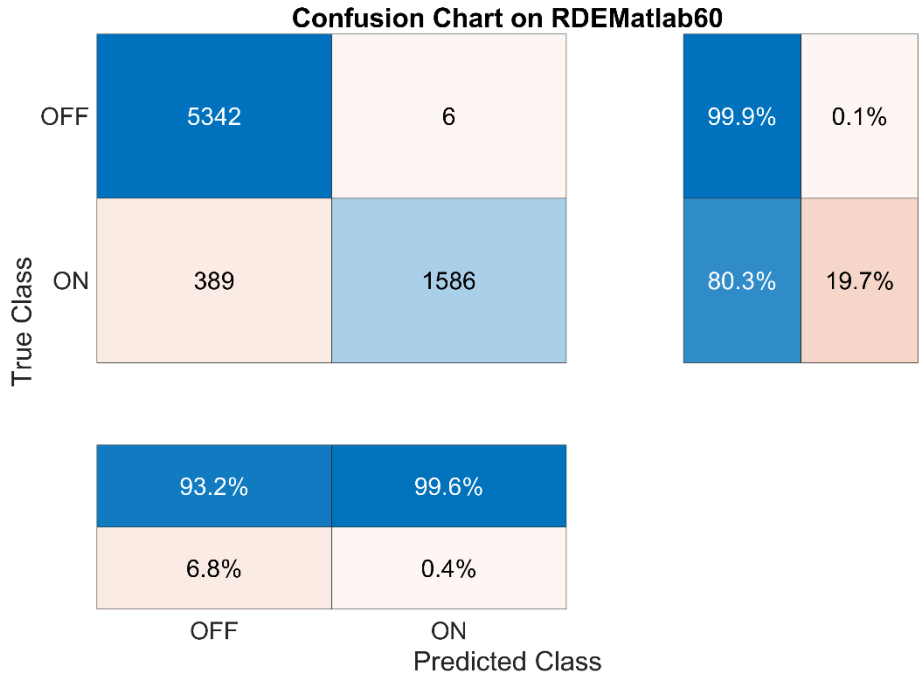

Figure 5.20: Confusion chart on one of the test driving cycles of engine status optimized neural network

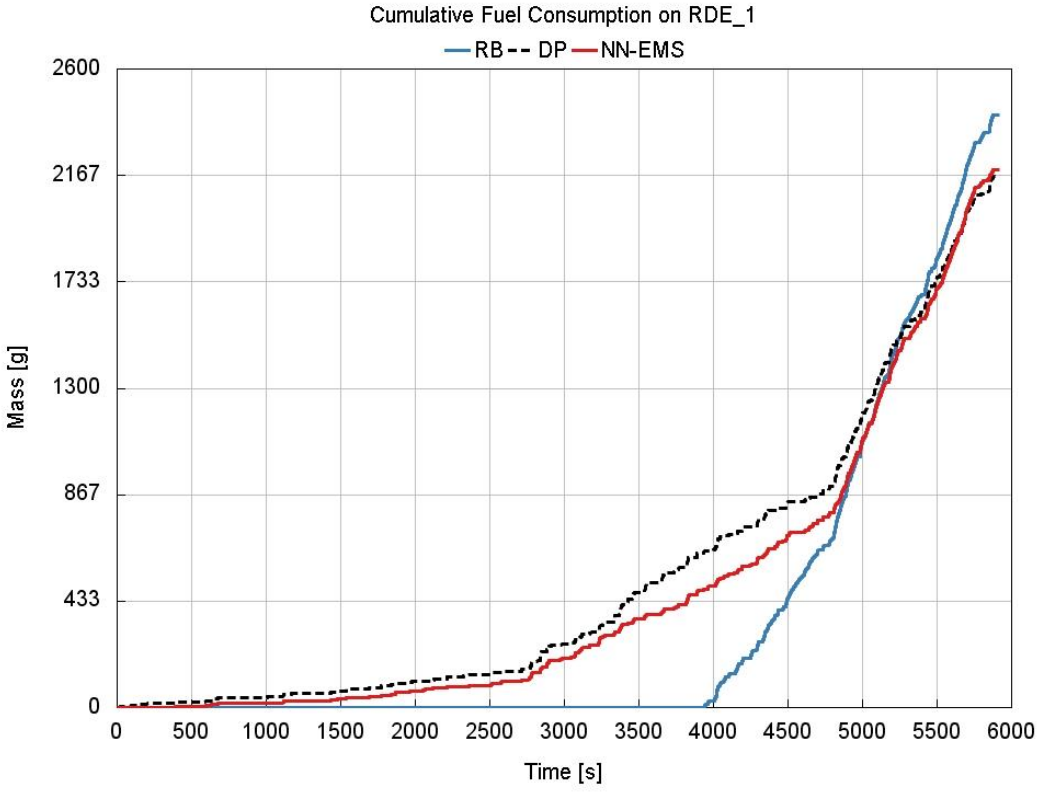

Figure 5.21: Fuel consumption comparison between DP and NN-based EMS on the first test driving cycle

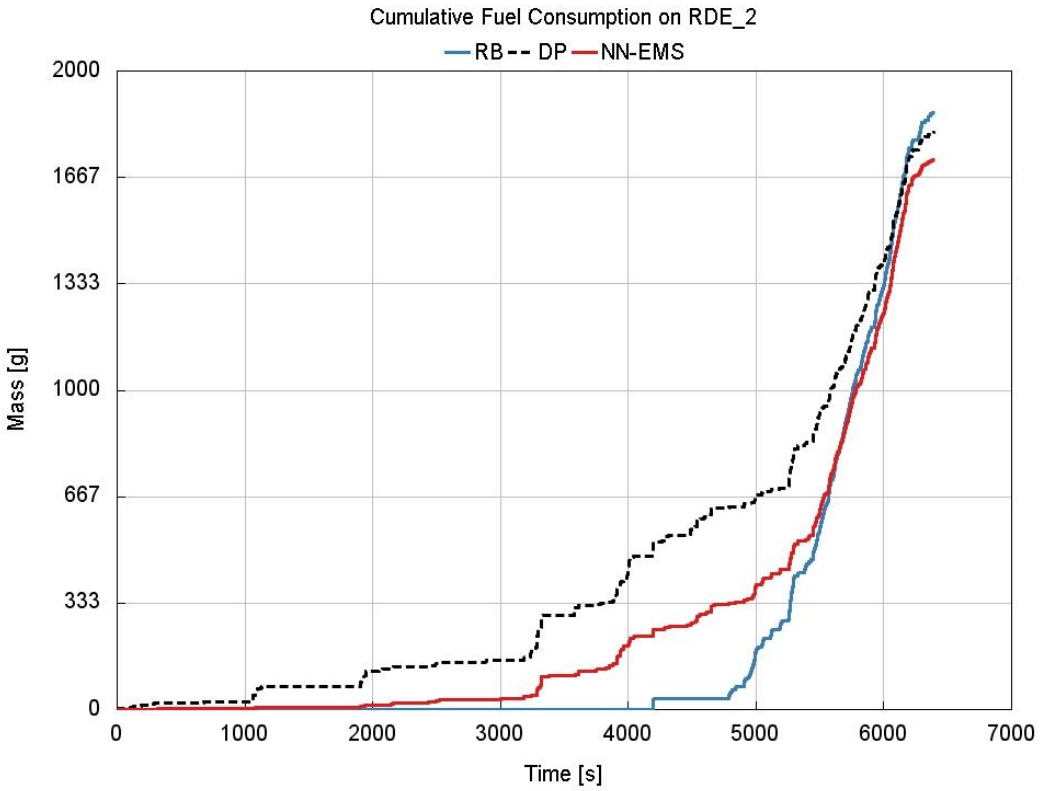

Figure 5.22: Fuel consumption comparison between DP and NN-based EMS on the second test driving cycle

In Figure 5.21 and Figure 5.22, DP, RB and NN-based EMS fuel consumption are compared. RB strategy exploit in the first part of both the cycles a pure electric driving mode, since the fuel consumption starts to increase after a considerable amount of time, in which a charge sustaining mode is enabled. This strategy leads to higher fuel consumption than NN-based EMS that adopt a pure charge depleting strategy, even if the final SOC of the latter strategy is lower.

However, despite the lower SOC in NN-based EMS, the fuel consumption reduction is remarkable, as shown in Table 5.5, highlighting the more efficient strategy proposed by this method.

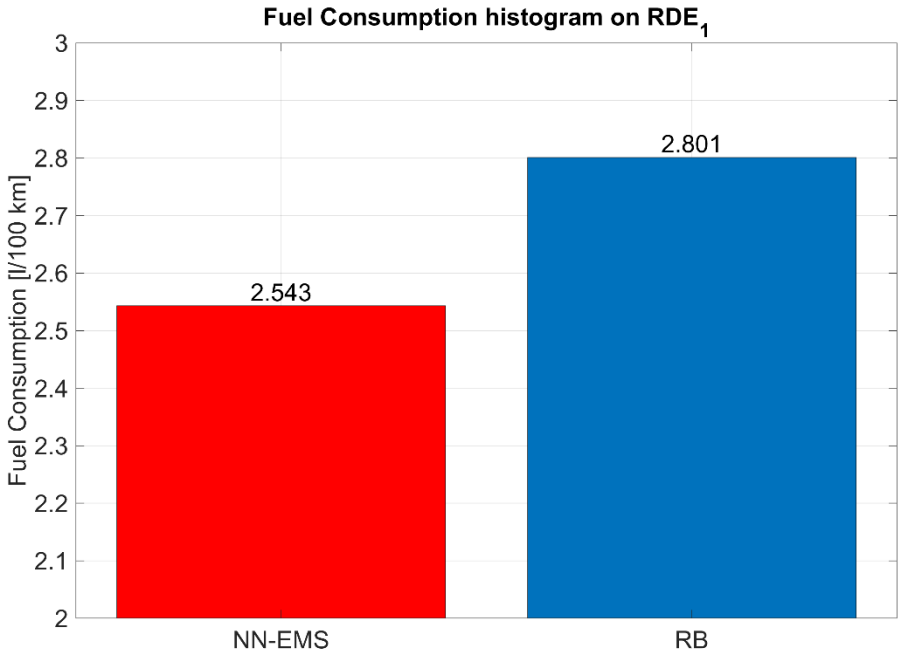

Figure 5.23: Fuel consumption histogram with NN-based EMS and DP optimized value on first test driving cycle

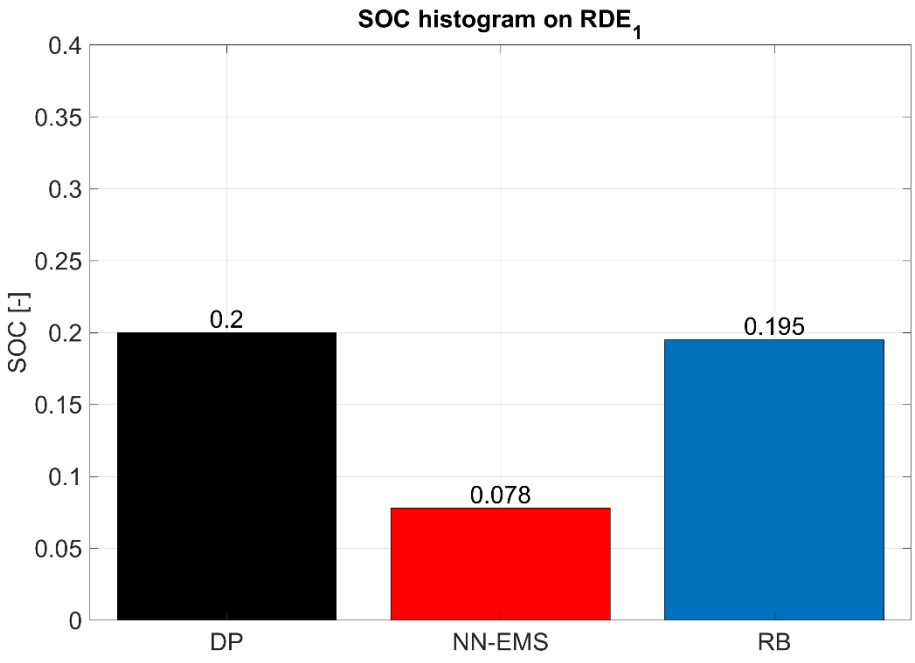

Figure 5.24: State of Charge histogram on first test driving cycle

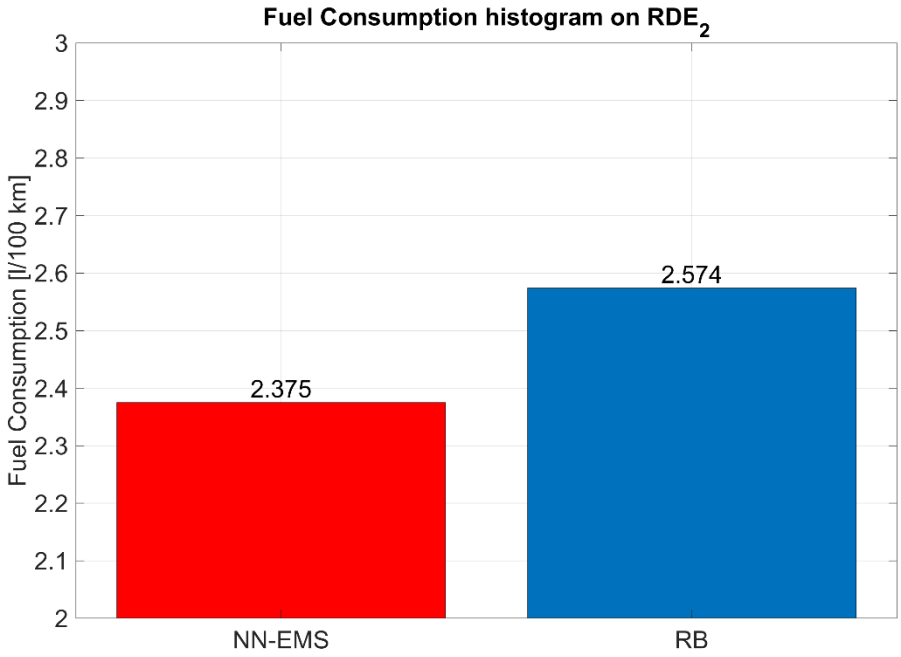

Figure 5.25: Fuel consumption histogram with NN-based EMS and DP optimized value on second test driving cycle

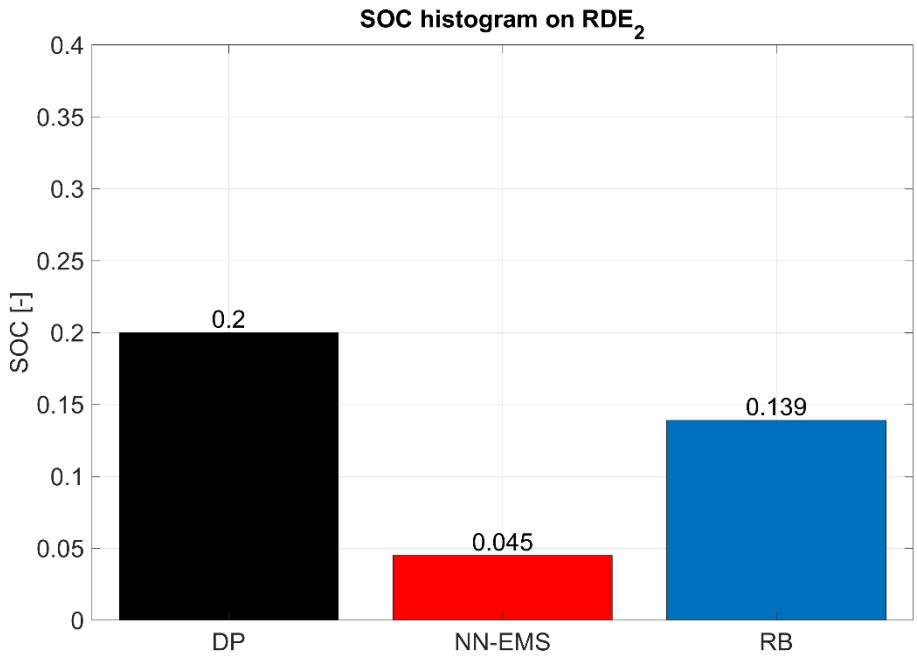

Figure 5.26: State of Charge histogram on second test driving cycle

Table 5.4: SOC in charge depleting comparison with the target on the two test driving cycles

Control strategy	SOC distance from the target on RDE_1	SOC distance from the target on RDE_2
DP	0 %	0 %
NN-EMS	-61 %	-77.5 %
RB	-2.56 %	-43.9 %

Table 5.5: Fuel consumption reduction in charge depleting comparison with the two purposed strategies on the two test driving cycles

Control strategy	SOC distance from the target on RDE_1	SOC distance from the target on RDE_2
RB	0 %	0 %
NN-EMS	-10.2 %	-8.38 %

6 Conclusion

The increasingly demanding worldwide trend concerning the reduction in terms of CO_2 and pollutant emissions is leading all the automotive field to be rethought.

The regulator-imposed CO_2 emissions targets are increasingly strict, leading car manufacturers to search new solutions to increase the fuel efficiency with the goal of saving as much fuel as possible, reducing the CO_2 emissions.

Among the solutions provided by the literature, the introduction of hybrid powertrain to propel the vehicle is one of the most promising, thanks to the possibility of combining the advantages of traditional internal combustion engines and full electric vehicle.

However, the introduction of an ancillary source of energy with its associated actuator, introduces an additional degree of freedom, that is the power split among the available actuators.

This is to be managed by the Energy Management System of the powertrain controller, to fully exploit all the benefits that an electrified powertrain architecture can provide in terms of fuel saving and fuel efficiency.

This dissertation was mainly focused on the methodology used to design a deep learning-based Energy Management System, with the exploitation of deep Recurrent Neural Networks, an Artificial Intelligence model able to use past information coming from the driving cycle in order to decide to decide the optimal power split.

Different neural networks solutions and architectures have been tested and the most promising one is based on two different networks that decide respectively the engine state and the engine Brake Mean Effective Pressure (BMEP) at each time step, trained with the optimal solution provided by Dynamic Programming optimization algorithm.

A plug-in Hybrid Electric Vehicle with a P2 powertrain architecture available on the European market was used to test the proposed Energy Management System. The proposed approach was tested, by means of numerical simulation, on a validated model of the case study implemented in the GT-SUITE environment.

The results have been compared with two different powertrain control strategy based on a Rule-Based strategy, excreted from the experimental campaign carried out on the real vehicle, and an Equivalent Consumption Minimization Strategy (ECMS) optimization.

Remarkable achievement in terms of fuel economy were achieved on both charge sustaining and charge depleting strategy. In charge sustaining, there is a fuel consumption reduction of about 4 %

respect to the Rule Based strategy, as shown in Table 5.2, with a remarkable charge sustaining capacity of the Energy Management System. The engine exploitation is more efficient thanks to the lowest BSFC among the compared control strategies (Table 5.1).

In charge depleting, the EMS is able to provide a full charge depleting strategy on the entire driving cycle (Figure 5.24 and Figure 5.26), reaching a fuel consumption reduction of about 10 % respect to Rule Based EMS (Table 5.5).

A remarkable improvement of the Neural Network-based EMS can be obtained exploiting V2X connectivity, with the possibility of considering future information about the predicted vehicle speed, road infrastructure and traffic to improve the capacity of the EMS of reaching the SOC target value and further increases the engine efficiency.

As final consideration, even if this thesis work presents remarkable results in terms of fuel saving with the exploitation of a supervised learning model, such as a deep learning model, future development of EMS based on Artificial Intelligence should exploit the Reinforcement Learning (RL) algorithms. RL is more suitable to complex control problems that act in highly dynamic environment, such as the design of the Energy Management System of a Hybrid Electric Vehicle.

Acknowledgement

Con questo ultimo pensiero si chiude un altro importantissimo capitolo della mia vita. È stato un percorso sicuramente pieno di ostacoli e difficoltà, ma allo stesso tempo molto stimolante, che mi ha accresciuto prima come uomo e poi come studente e dato la possibilità di conoscere persone splendide provenienti da ogni parte di Italia ed Europa.

Non posso far altro che ringraziare in primis la mia famiglia per tutti i sacrifici che ha fatto per permettermi di studiare in una delle più prestigiose università italiane, per il continuo e costante supporto che ha saputo darmi nei momenti di difficoltà, per aver condiviso con altrettanta gioia, la mia felicità quando sono riuscito a realizzare gli obiettivi che mi ero prefissato.

Non smetterò mai di ringraziare i miei genitori per il modo in cui mi hanno fatto crescere, l'educazione che mi hanno fornito e i valori che mi hanno trasmesso, sperando che questo mio traguardo possa ripagarli in parte di tutto quello che hanno fatto per me.

Un enorme ringraziamento va fatto a Celeste, la mia ragazza, che ha saputo condividere e superare con me le difficoltà che la distanza impone ad una relazione. Ha saputo comprendermi, supportarmi e, soprattutto, sopportarmi in ogni momento durante questi cinque anni universitari, non facendomi pesare che ci fossero 1000 km tra di noi, ma rendendo ogni singola videochiamata e ogni singolo incontro, anche se per pochi giorni, una situazione di assoluta normalità come se questa distanza fosse annullata.

Un ringraziamento speciale va al Prof. Federico Millo e al Prof. Luciano Rolando che mi hanno dato la possibilità di potermi cimentare in un così appassionante e stimolante campo tecnologico, quale quello dei veicoli ibridi. Non posso non citare Luca, per la sua enorme disponibilità nel fornirmi aiuto e nel darmi le più disparate spiegazioni durante lo sviluppo della mia tesi. Penso che una guida migliore non potesse capitarmi.

Infine, ci tengo a ringraziare tutti i miei amici, dagli amici di "giù" a tutte le fantastiche persone che hanno reso la mia esperienza torinese il più piacevole possibile, a partire dalle più classiche birre al Sir Francis Drake per finire alle serate più distruttive. Un grazie va a "Gravina", nella quale e con i quali ho i ricordi più belli di questa esperienza universitaria. Grazie anche a tutte le persone che ho conosciuto durante l'Erasmus in Danimarca, con le quali ho condiviso uno dei periodi più belli della mia vita e che più mi ha accresciuto come uomo.

Grazie a Torino, sperando che questo sia il punto di partenza per nuove affascinanti avventure.

Ad Maiora, Gigi.

Bibliography

[1] R. Lindsey, ‘Climate Change: Atmospheric Carbon Dioxide’, 2020. https://www.climate.gov/print/8431 (accessed Sep. 13, 2021).

[2] L. Rolando, ‘An Innovative Methodology for the Developmenet of HEVs Ehenrgy Management System’, Doctoral Thesis, Politecnico di Torino, 2012.

[3] European Commission, ‘Reducing CO2 emissions from passengers cars’. https://ec.europa.eu/clima/policies/transport/vehicles/cars_en (accessed Sep. 13, 2021).

[4] E. Spessa, ‘Controllo delle emissioni di inquinanti’. Politecnico di Torino.

[5] S. Onori, L. Serrao, and G. Rizzoni, *Hybrid electric vehicles: Energy management strategies*, no. 9781447167792. Springer Publishing Company, 2016.

[6] V. Silvio and R. Luciano, ‘Hybrid propulsion system’. Politecnico di Torino.

[7] F. Millo, L. Rolando, and E. Servetto, ‘Development of a Control Strategy for Complex Light-Duty Diesel-Hybrid Powertrains’, *SAE Tech. Pap.*, Sep. 2011, doi: 10.4271/2011-24-0076.

[8] J. M. Morbitzer, ‘High-level modeling, supervisory control strategy development, and validation for a proposed power-split hybrid-electric vehicle design’. 2005.

[9] C. C. Lin, H. Peng, J. W. Grizzle, and J. M. Kang, ‘Power management strategy for a parallel hybrid electric truck’, *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 6, pp. 839–849, Nov. 2003, doi: 10.1109/TCST.2003.815606.

[10] G. Paganelli, ‘Conception et commande d’une chaîne de traction pour véhicule hybride parallèle thermique et électrique’, *undefined*, 1999.

[11] G. Paganelli, T. M. Guerra, S. Delprat, J. J. Santin, M. Delhom, and E. Combes, ‘Simulation and assessment of power control strategies for a parallel hybrid car’, *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 214, no. 7, pp. 705–717, 2000, doi: 10.1243/0954407001527583.

[12] D. P. Bertsekas, *Dynamic programming and optimal control*. .

[13] O. Sundström and L. Guzzella, ‘A Generic Dynamic Programming Matlab Function’, 2009.

[14] E. Gross, ‘On the Bellman’s principle of optimality’, *Phys. A Stat. Mech. its Appl.*, vol. 462, pp. 217–221, Nov. 2016, doi: 10.1016/J.PHYSA.2016.06.083.

[15] D. D. Tran, M. Vafaeipour, M. El Baghdadi, R. Barrero, J. Van Mierlo, and O. Hegazy, ‘Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated energy management strategies’, *Renewable and Sustainable Energy Reviews*, vol. 119. Elsevier Ltd, Mar. 01, 2020, doi: 10.1016/j.rser.2019.109596.

[16] MathWorks, ‘Rete neurale convoluzionale’. https://it.mathworks.com/discovery/convolutional-neural-network-matlab.html (accessed Oct. 07, 2021).

[17] ‘Il Reinforcement Learning nel Retail’. https://wwwakite.azurewebsites.net/it/news/il-reinforcement-learning-nel-retail (accessed Oct. 07, 2021).

[18] S. Russel and P. Norvig, *Artificial intelligence: a modern approach*. 2012.

[19] M. A. Nielsen, *Neural Networks and Deep Learning*. 2015.

[20] M. Tom, *Machine learning*. 2019.

[21] MathWorks, ‘Machine Learning with MATLAB’. .

[22] MathWorks, ‘Reinforcement learning with MATLAB’. .

[23] MathWorks, ‘Machine Learning Course’. .

[24] D. Arthur and S. Vassilvitskii, ‘K-means++: The advantages of careful seeding’, *Proc. Annu. ACM-SIAM Symp. Discret. Algorithms*, vol. 07-09-Janu, pp. 1027–1035, 2007.

[25] T. Evgeniou and M. Pontil, ‘Support vector machines: Theory and applications’, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2049 LNAI, pp. 249–257, 2001, doi: 10.1007/3-540-44673-7_12.

[26] Wikipedia, ‘Support-Vector Machine’. https://en.wikipedia.org/wiki/Support-vector_machine (accessed Oct. 07, 2021).

[27] R. Irina, ‘An Empirical Study of the Naïve Bayes Classifier’, 2001.

[28] M. RH, ‘Neuronal cell types’, *Curr. Biol.*, vol. 14, no. 13, 2004, doi: 10.1016/J.CUB.2004.06.035.

[29] K. L. Du and M. N. S. Swamy, *Neural networks design*. 2006.

[30] MathWorks, ‘MATLAB trainingOptions -’. https://it.mathworks.com/help/deeplearning/ref/trainingoptions.html?searchHighlight=trainingoptions&s_tid=srchtitle (accessed Sep. 14, 2021).

[31] MathWorks, ‘Softmax layer - MATLAB’. https://it.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html (accessed Sep. 14, 2021).

[32] MathWorks, ‘Cross-entropy loss for classification tasks - MATLAB’. https://it.mathworks.com/help/deeplearning/ref/dlarray.crossentropy.html (accessed Sep. 14, 2021).

[33] Wikipedia, ‘Gradient descent’. https://en.wikipedia.org/wiki/Gradient_descent (accessed Oct. 05, 2021).

[34] A. Animesh, ‘The Problem of Vanishing Gradients’. https://towardsdatascience.com/the-

problem-of-vanishing-gradients-68cea05e2625 (accessed Oct. 07, 2021).

[35] J. Bronwlee, ‘How to Fix the Vanishing Gradients Problem Using the ReLU’. https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/ (accessed Oct. 07, 2021).

[36] D. P. Kingma and J. L. Ba, ‘Adam: A method for stochastic optimization’, *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, 2015.

[37] H. Kong, Y. Fang, L. Fan, H. Wang, X. Zhang, and J. Hu, ‘A Novel Torque Distribution Strategy Based on Deep Recurrent Neural Network for Parallel Hybrid Electric Vehicle’, *IEEE Access*, vol. 7, no. August, pp. 65174–65185, 2019, doi: 10.1109/ACCESS.2019.2917545.

[38] A. Sherstinsky, ‘Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network’, *Phys. D Nonlinear Phenom.*, vol. 404, Aug. 2018, doi: 10.1016/j.physd.2019.132306.

[39] MathWorks, ‘Gated recurrent unit (GRU) layer - MATLAB’. https://it.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.grulayer.html (accessed Sep. 14, 2021).

[40] S. Hochreiter and J. Schmidhuber, ‘Long Short-Term Memory’, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[41] MathWorks, ‘Long Short-Term Memory Networks - MATLAB’. https://it.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html (accessed Sep. 14, 2021).

[42] G. Dipierro, E. Galvagno, G. Mari, F. Millo, M. Velardocchia, and A. Perazzo, ‘A reverse-engineering method for powertrain parameters characterization applied to a P2 plug-in hybrid electric vehicle with automatic transmission’, *SAE Tech. Pap.*, vol. 2020-June, no. June, Jun. 2020, doi: 10.4271/2020-37-0021.

[43] R. Luciano, M. Federico, and P. Luca, ‘Energy Management System Optimization Based on V2X Connectivity’, pp. 13–17, 2021, doi: 10.46720/F202N-XXX-NNN.

[44] F. Millo, L. Rolando, and M. Andreat, ‘Numerical Simulation for Vehicle Powertrain Development’, *Numer. Anal. - Theory Appl.*, Sep. 2011, doi: 10.5772/24111.

[45] ‘Concetto di overfitting’. https://ichi.pro/it/concetto-di-overfitting-167655442738138 (accessed Oct. 07, 2021).

[46] MathWorks, ‘Experiment Manager - MATLAB’. https://it.mathworks.com/help/deeplearning/ref/experimentmanager-app.html (accessed Sep. 28, 2021).