

POLITECNICO DI TORINO

Master's Degree Course in Mechatronic Engineering

Master's Degree Thesis



Politecnico
di Torino



Model based and AI range estimation for small EV vehicles

Supervisor:

Dott. Prof. Carlo Novara

Candidate:

Francesco Grillo

Company Supervisor:

Ing. Claudio Russo

Academic Year 2020/2021

ABSTRACT

The excessive growth of greenhouse gases, among the most dangerous to the Earth and human health itself, has forced the development of green technologies with the aim to solve this issue. Since one of the most negative factors, which has a big impact to the increasing of the pollution is the transportation field, both public and private, the carmakers have decided to introduce, in their production lines, electric vehicles (EV), both hybrid (HEV) and fully electric (BEV). Despite the goal is to reduce the greenhouses gases, the introduction of zero emission vehicles is not the solution to the problem, but only a starting point. Since the battery is one of the two sources of energy in HEV or the unique in BEV, it is necessary to have a good estimation of the state of charge (SOC). This is a fundamental parameter used in the battery management system (BMS) of the EV's battery; in fact, it reflects a lot of battery performances, can guarantee battery protection, prevents overcharging and overdischarging, improves and extends the battery life, but it can be used also to implement control strategies. Moreover, one of the most important problems associated to the limited spread of electric vehicles is the range anxiety. Among the possible resolutions of this problem, there is the development of a system able to compute an estimation, accurate and robust, of the SOC level of consumption of the battery system. Nevertheless, finding a good estimation of the SOC is a still open challenge, due to the complexity of the problem. The aim of this thesis work, carried out in collaboration with Bylogix S.r.l, is to develop two different types of algorithms, both written in Python language, that provide an estimation of the SOC that the vehicle will be expected to consume. In the design process of the two algorithms and, mainly, in the second one, real data acquired by a Tesla Model 3 with a 75 kWh battery pack were used; the data were provided by the company. The first algorithm was designed on a model-based approach; therefore, it is based on the physical knowledge of the problem and on the equations that allow to calculate the power consumption delivered by the battery pack for the vehicle motion, starting from the longitudinal vehicle dynamics equation. Then, the energy and the associated SOC consumption, needed to complete the travel, are derived. Velocity, altitude profile, and weather information are the input data used by the algorithm. The algorithm is then tested with respect to real drive cycles, extracted from the database given by the company. The second algorithm, on the other hand, was designed following a data-driven approach, including an artificial neural network model. The network, based on a feed-forward architecture, computes an estimation of the power consumption delivered by the battery pack, using velocity, acceleration, and slope road angle as inputs. After, the power quantity returned by the neural network is integrated, obtaining the energy, and thus, the SOC estimation is finally computed. The training, validation, and testing processes of this algorithm are performed using the same database provided. Moreover, both the algorithms were designed to work offline, so are

able to provide an estimation of the SOC consumption before the trip begin. To achieve this purpose, an algorithm that predicts the velocity profile was implemented, starting from road information and weather data. These data are obtained from APIs services and online databases, such as OpenStreetMap.

*A te,
che ci sei sempre stato e sempre ci sarai.
Per sempre, Papi, Ti amo.*

TABLE OF CONTENTS

LIST OF FIGURES	x
LIST OF TABLES	xvi
ACRONYMS	xviii
THESIS OVERVIEW	xx
Chapter 1	2
INTRODUCTION	2
1.1 A BIT OF HISTORY ABOUT ELECTRIC VEHICLES	2
1.2 PROS AND CONS OF ELECTRIC VEHICLES	5
1.3 MOTIVATIONS.....	7
1.4 THESIS OBJECTIVE AND CONTRIBUTIONS.....	10
Chapter 2	14
SOC MEASUREMENTS AND ESTIMATIONS TECHNIQUES	14
2.1 SOC DEFINITION.....	15
2.2 DIRECT MEASUREMENT METHODS	16
2.2.1 COULOMB COUNTING METHOD	17
2.2.2 OPEN CIRCUIT VOLTAGE METHOD	18
2.3 MODEL-BASED METHODS	20
2.3.1 EQUIVALENT CIRCUIT MODEL FOR BATTERY	20
2.3.2 ESTIMATION FILTER.....	23
2.4 DATA-DRIVEN METHODS.....	28
Chapter 3	30
A NOVEL MODEL-BASED APPROACH FOR OFFLINE SOC ESTIMATION	30
3.1 DESIGN OF THE ALGORITHM	31
3.2 INPUT DATA OF THE ALGORITHM	32

3.2.1 WAYPOINTS COORDINATES.....	33
3.2.2 OPEN STREET MAP	35
3.2.3 ROAD INFORMATION EXTRACTION.....	38
3.2.3.1 EXTRACTION TECHNIQUE 1	39
3.2.3.2 EXTRACTION TECHNIQUE 2	43
3.2.3.3 ROAD STREET IDENTIFICATION.....	44
3.2.4 VELOCITY PROFILE PREDICTION	45
3.2.5 DISTANCE PROFILE.....	53
3.2.6 ELEVATION PROFILE.....	55
3.2.7 WHEATER DATA.....	58
3.3 VEHICLE LONGITUDINAL DYNAMICS.....	59
3.3.1 TOTAL TRACTION FORCE.....	60
3.3.2 TOTAL TRACTION POWER AT WHEELS LEVEL	62
3.3.3 POWER CONVERSION FROM WHEELS TO BATTERY SIDE.....	64
3.4 ENERGY CONSUMPTION	68
3.5 SOC ESTIMATION	70
3.6 PARAMETERS IDENTIFICATION.....	72
3.6.1 AIR DENSITY	72
3.6.2 RELATIVE WIND VELOCITY	75
3.6.3 ROAD SLOPE	79
3.6.4 ROLLING RESISTANCE COEFFICIENT	80
3.7 VALIDATION OF THE ALGORITHM.....	81
3.7.1 Drive cycle, DC1	82
3.7.2 ANALYSIS ON A TEST SET.....	84
3.8 TEST OF THE MODEL-BASED ALGORITHM IN A REAL SCENARIO	87
3.9 CONCLUSIONS.....	98
Chapter 4.....	100
A MACHINE LEARNING TECNIQUE FOR SOC ESTIMATION.....	100
4.1 MACHINE LEARNING, A NEW FRONTIER FOT THE ESTIMATION PROBLEM.....	101
4.2 THEORY ABOUT NEURAL NETWORKS.....	104
4.2.1 THE NEURON.....	104
4.2.2 NEURAL NETWORK ARCHITECTURE	106
4.2.3 TRAINING WORKFLOW	107
4.2.4 FEED-FORWARD STEP	108

4.2.5 LEARNING PROCESS.....	111
4.2.6 PROBLEMS DURING THE TRAINING PROCESS	115
4.2.7 EVALUATION METRICS.....	119
4.2.8 ACTIVATION FUNCTIONS.....	121
4.3 DESIGN OF A NEURAL NETWORK FOR THE SOC ESTIMATION PROBLEM.....	127
4.3.1 DATA SELECTION.....	129
4.3.2 MODELS EVALUATION.....	131
4.3.3 AN OVERVIEW OF THE DEVELOPED NEURAL NETWORK MODEL.....	139
4.4 ANALYSIS OF THE RESULTS USING ANN ALGORITHM.....	140
4.4.1 Drive cycle, DC1	141
4.4.2 Drive cycle, DC2	144
4.4.3 Drive cycle, DC3	147
4.4.4 STATISTICAL ANALYSIS.....	150
4.5 CONCLUSIONS	153
Chapter 5.....	154
COMPARISON OF THE OBTAINED RESULTS AND CONCLUSIONS.....	154
5.1 DRIVE CYCLE 1	154
5.2 COMPARISON ON 22 DRIVE CYCLES' SET.....	156
5.3 THESIS CONCLUSIONS AND FUTURE DEVELOPMENTS.....	158
REFERENCES	160

LIST OF FIGURES

Figure 1. Electric car produced by GM in 1973. [4]	3
Figure 2. Toyota Prius, the first hybrid vehicle. [5].....	4
Figure 3. Fiat 500e.	5
Figure 4. Tesla model X. [6]	5
Figure 5. Diagram with sources of CO2. [8].....	6
Figure 6. Sold of EV per country, in Europe, between 2013 and 2017. [11].....	7
Figure 7. Tesla Supercharger. [14]	8
Figure 8. Enel X JuiceBox. [15].....	9
Figure 9. State of charge representation. [9].....	15
Figure 10. Example of Coulomb Counter sensor. [10]	18
Figure 11. Comparison of discharge curve between a Lead-Acid and lithium-Ion. [10]	18
Figure 12. State of Charge versus OCV in Lead-Acid battery. [20]	19
Figure 13. State of Charge versus OCV in Lead-Acid battery. [20]	19
Figure 14. State of Charge versus OCV in Lead-Acid battery. [20]	21
Figure 15. Stages of battery integration on an electric vehicle. [9].....	23
Figure 16. Block diagram for state estimation problem, where \mathbf{u} is the input data vector, \mathbf{X} is the true state vector of the system, \mathbf{Z} is the measurement vector, \mathbf{v} is the observation vector, $\hat{\mathbf{X}}$ is the estimated state vector. [21]	24
Figure 17. Block diagram of the Predictor/Corrector form of the Kalman filter algorithm. [24] .	26
Figure 18. General review on Predictor/Corrector form of the KF. [13]	26
Figure 19. Working flow of the model-based algorithm for SOC estimation, in real application scenario.....	32
Figure 20. Example of GPX file containing the waypoint coordinates associated to a route.	34
Figure 21. Example of the web interface of OSRM route map service. [30]	34
Figure 22. Example of node information in XML format. Note the presence of node id, latitude, longitude, and tag. [32]	36
Figure 23. Example of way information in XML format. Note the presence of way id, list of node ids and tags. [33].....	36
Figure 24. Example of query with node.....	37
Figure 25. Example of query with way.....	37
Figure 26. Example of query with way.....	37
Figure 27. Example of data parsing for extraction of node information on a Python dictionary, named node.	38
Figure 28. Example of data parsing for extraction of way information on a Python dictionary, named way.....	38
Figure 29. Operation flow followed to extract the road data from OSM, Technique 1.	39
Figure 30. Query to download road information, written Overpass QL language.	39
Figure 31. Example of complete request to download road data, in Overpass QL language, using technique 1.	40
Figure 32. Example of OSM nodes presented in the selected area.	41
Figure 33. Example of application of technique 1, with 5 OSM's nodes and 3 waypoints.	42

Figure 34. Example of complete request to download road data, in Overpass QL language, using technique 2.	43
Figure 35. Algorithm for road data extraction.	45
Figure 36. Legal speed limit profile obtained by the algorithm, according to both road signs and road classification.	47
Figure 37. Implementation of the backward algorithm.	47
Figure 38. Velocity-acceleration profile according to real collected data. In red the deceleration trend, in black the acceleration one as function of the velocity.	48
Figure 39. Legal speed limit profile. In red the velocity profile considering the acceleration, in black the basic case without acceleration.	49
Figure 40. Legal speed limit profile. In red the velocity profile considering the deceleration, in black the basic case without deceleration.	49
Figure 41. Example of velocity profile obtained.	50
Figure 42. Example of velocity profile obtained by the algorithm.	51
Figure 43. Example of velocity profile obtained by the algorithm, with a zoom on the first 3 Km.	51
Figure 44. Example of velocity profile obtained by the algorithm, with a zoom on range 2-16 Km.	52
Figure 45. Example of velocity profile obtained by the algorithm, with a zoom on the last 7 Km.	52
Figure 46. Flow chart for the legal speed limit profile generation.	53
Figure 47. Example of route made by 5 nodes and 4 ways. The total length, in meters is $\ell_{TOT} = \ell_1 + \ell_2 + \ell_3 + \ell_4$	53
Figure 48. Elevation profile obtained from the algorithm.	56
Figure 49. Elevation profile, focused on the firsts 2.5 Km.	56
Figure 50. Implementation of the Butterworth digital filter in Python.	56
Figure 51. Elevation profile obtained from the algorithm. In red the filtered profile, in black the original one.	57
Figure 52. Elevation profile, focused on the firsts 2.5 Km. In red the filtered profile, in black the original one.	57
Figure 53. Total forces acting on a BEV [44].	59
Figure 54. Comparison of all the forces acting a vehicle according to the longitudinal dynamic equation.	61
Figure 55. Comparison of all the power acting a vehicle according to the longitudinal dynamic equation.	62
Figure 56. Main components of a total electric vehicle. [44].	64
Figure 57. Pre-flight briefing of Volkswagen VW ID.3.	64
Figure 58. Comparison between the power consumption at wheels level (back) and power consumption at electric motor level (red), associated to a trip.	67
Figure 59. Comparison among the power consumption at wheels level (black), power consumption at electric motor level (red) and power consumption at battery level (blue), associated to a trip.	67
Figure 60. RMSE and MAE values associated to the accuracy computed from data in Figure 61, plot (a).	71
Figure 61. Comparison between the energy computed from measured battery power (eq. 3.35) and energy from measured SOC (eq. 3.36) (a). Absolute error between the two quantities (b). .	71
Figure 62. Air density dependency on ambient temperature and air pressure, fixed the humidity level.	73
Figure 63. Example of main wind direction in a compass diagram. [52]	76

Figure 64. Diagram representing a wind blowing from NE (30°). The black arrow is the vehicle longitudinal direction, the blue one is the wind direction, and the purple is the new wind direction.....	77
Figure 65. Diagram with the main vectors to be considered in relative wind speed computation.	77
Figure 66. Example of road with a slope. Δd = distance from A to B, Δh = altitude level from A to B, l = total length, θ = slope angle.	79
Figure 67. Example of the slope profile, expressed in percentage.	80
Figure 68. Some data about the drive cycle 1.....	82
Figure 69. Comparison between real and estimated power consumption associate to DC1.....	82
Figure 70. Comparison between real and estimated energy consumption (a) and the associated absolute error (b), relative to DC1.	83
Figure 71. Comparison between real and estimated SOC consumption (a) and the associated absolute error (b), relative to DC1.	83
Figure 72. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC1.....	84
Figure 73. Metrics performances for estimation of SOC consumption, related to all the 22 drive cycles, used for testing.	85
Figure 74. Absolute error relatives to the SOC computation over the time, for all the 22 drive cycles used for testing.	85
Figure 75. Plot with mean (blue) and standard deviation (gray) values relative to the absolute SOC error over time, computed considering all the 22 routes for testing.	86
Figure 76. Mean value with standard deviation, related to the final and maximum absolute error value, associated to the SOC quantity, computed considering the 22 routes for testing.	87
Figure 77. Route selected for testing test, from Bylogix S.r.l HQ, Grugliasco (A) to via Corrado Corradini, Torino (B).	88
Figure 78. Road classification extracted from OpenStreetMap, relative to Trip A. [4].....	89
Figure 79. Elevation profile of the selected route, Trip A.	89
Figure 80. Slope profile of the selected route, in [%], relatives to Trip A.	89
Figure 81. Velocity profile predicted by the algorithm with BVF=0.90 (red), compared with the real velocity profile measured (blue).	90
Figure 82. Acceleration profile derived from the predicted velocity profile.....	91
Figure 83. Power consumption at different vehicle's systems.	91
Figure 84. Energy consumption measured (red) and energy consumption estimated (blue) from battery power, relative to the trip A. BVF = 0.90.	92
Figure 85. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. SOC(t_0) = 90, BVF = 0.90.	92
Figure 86. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. SOC(t_0) = 89.40, BVF = 0.90.	93
Figure 87. Velocity profile predicted by the algorithm with BVF=1.02.....	94
Figure 88. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. SOC(t_0) = 90, BVF = 1.02.	94
Figure 89. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. SOC(t_0) = 89.40, BVF = 1.02.	95
Figure 90. Comparison of different velocity profiles, according to different BVF, with respect the measured velocity (red line), relative to trip A.....	96
Figure 91. Comparison of different energy consumption trend, associated to different BVF, with respect the measured one (red line), relative to trip A.....	96

Figure 92. Comparison of different SOC consumption levels, associated to different BVF, with respect the measured one (red line), relative to trip A.	97
Figure 93. Design schema of the ANN algorithm to SOC estimation.	101
Figure 94. Black-box model. [56].....	102
Figure 95. Machine learning and classical programming differences. [57].....	103
Figure 96. Function implemented in a single artificial neuron. [60].....	105
Figure 97. Recurrent Neural Network and Feed-Forward Neural Network.	107
Figure 98. Workflow of the main step characterizing the training process in a neural network.	108
Figure 99. Example of ANN with 2 layers used to estimate the SOC. [12]	109
Figure 100. Working principle of the DNN. [54]	112
Figure 101. $fx = x^2$	113
Figure 102. Example of GD algorithm with different learning rate.	113
Figure 103. Gradient Descent process.	114
Figure 104. Example of model with high accuracy. [63]	115
Figure 105. Example of model with underfitting (left), overfitting (right), and optimal model (center).	117
Figure 106. Example of underfitting, overfitting and optimal model in learning curves.	118
Figure 107. Example of possible learning curves scenario.....	118
Figure 108. Binary Step function.....	122
Figure 109. Linear function.....	123
Figure 110. Sigmoid function.	124
Figure 111. Hyperbolic tangent function.....	125
Figure 112. ReLU function.	126
Figure 113. Leaky ReLU function.	126
Figure 114. Parametrized Leaky ReLU function, with several parameters' values.	127
Figure 115. Example of data used in the neural network model design.	128
Figure 116. Workflow for neural network model design.....	129
Figure 117. Example of data contained in a driven cycle, used a test set for the neural network. In all the subplot, the x-axis indicates the travel time duration [s].	133
Figure 118. Learning curves for training and validation set associated to NN1.	134
Figure 119. Learning curves for training and validation set associated to NN2.	134
Figure 120. Learning curves for training and validation set associated to NN3.	135
Figure 121. Learning curves for training and validation set associated to NN4.	136
Figure 122. Learning curves for training and validation set associated to NN5.	136
Figure 123. MAE and RMSE on the training dataset, for the networks developed.	137
Figure 124. Evaluation of the MAE error on the battery power consumption, for the developed networks, in the three test sets, DC1, DC2, DC3.	138
Figure 125. Evaluation of the MAE error on the battery power consumption, for the developed networks, in the three test sets, DC1, DC2, DC3.	138
Figure 126. Architecture of the neural network NN5. In the hidden layer the node's activation functions are ReLu, instead, in the output layer is linear.	140
Figure 127. Drive cycle 1 data.	142
Figure 128. Comparison between real and estimated power consumption associate to DC1....	142
Figure 129. Comparison between real and estimated energy consumption (a) and the associated absolute error (b), relative to DC1.	143
Figure 130. Comparison between real and estimated SOC consumption (a) and the associated absolute error (b), relative to DC1.	143

Figure 131. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC1.....	144
Figure 132. Drive cycle 2 data.	145
Figure 133. Power consumption associate to DC2.	145
Figure 134. Energy consumption (a) and the associated absolute error (b), relative to DC2.....	146
Figure 135. SOC consumption (a) and the associated absolute error (b), relative to DC2.	146
Figure 136. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC2.....	147
Figure 137. Drive cycle 3 data.	147
Figure 138. Power consumption associate to DC3.	148
Figure 139. Energy consumption (a) and the associated absolute error (b), relative to DC3.....	148
Figure 140. SOC consumption (a) and the associated absolute error (b), relative to DC3.	149
Figure 141. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC3.....	149
Figure 142. Metrics performances for estimation of SOC consumption, related to all the 22 drive cycles, used for testing.	150
Figure 143. Absolute error relatives to the SOC computation over the time, for all the 22 drive cycles used for testing.	151
Figure 144. Plot with mean (blue) and standard deviation (gray) values relative to the absolute SOC error over time, computed considering all the 22 routes for testing.	152
Figure 145. Mean value with standard deviation, related to the final and maximum absolute error value, associated to the SOC quantity, computed considering the 22 routes for testing.	152
Figure 146. Main information about the DC1.	155

LIST OF TABLES

Table 1. Classification of state of charge estimation techniques.....	16
Table 2. Speed limit according to road classification. [36]	46
Table 3. Butterworth filter parameters.	57
Table 4. Energy consumptions values relative to the quantity obtained using the equation 3.36 and the 3.35.....	70
Table 5. Several reference values of rolling resistance coefficient. [45].....	80
Table 6. Numerical information about the energy and consumption associated to DC1.	83
Table 7. Resume of mean and standard deviation values relative to final and maximum absolute error value for SOC, computed considering the 22 routes for testing.....	86
Table 8. Recap of the energy and SOC consumption values associated to the trip A. $SOC(t_0) = 90$	95
Table 9. Table of comparison of the consumption cases, associated to different BVF values, relative to trip A. Column named ‘FEC’ means Final Energy consumption value. $SOC(t_0) = 90$	97
Table 10. Table to check underfitting and overfitting of the model by the error values.	118
Table 11. Dataset division.	130
Table 12. NN1, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).	133
Table 13. NN2, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).	134
Table 14. NN3, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).	135
Table 15. NN4, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).	135
Table 16. NN5, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).	136
Table 17. Sum-up of the RMSE and MAE values, relative to the output of the neural network, computed in the 3 test sets. In the column ‘Layers’, the letters I, H, O indicate Input, Hidden, and Output, respectively.	137
Table 18. Resume of the hyperparameters setting in the developed neural network NN5.	140
Table 19. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC1.....	143
Table 20. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC2.....	146
Table 21. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC3.....	149
Table 22. Resume of mean and standard deviation values relative to final and maximum absolute error value for SOC, computed considering the 22 routes for testing.....	152

Table 23. Comparison of the real and estimated values, obtained by the two algorithms, in terms of energy and SOC consumption, for DC1.....	156
Table 24. Final and maximum absolute error, in terms of mean and standard deviation, computed for SOC estimation by both the algorithms on 22 drive cycles.	157

ACRONYMS

ANN	Artificial Neural Network
BP	Back Propagation
BEV	Battery Electric Vehicle
BMS	Battery Management System
BVF	Bias Velocity Factor
CO ₂	Carbon dioxide
CAN	Controlled Area Network
DNN	Deep Neural Network
DOD	Depth of Discharge
EKF	Extended Kalman filter
EV	Electric Vehicle
FFNN	Feed-Forward Neural Network
GPS	Global Positioning System
GUI	Graphical User Interface
HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
KF	Kalman filter
LTI	Linear Time Invariant
LSTM	Long Short-Term Memory
ML	Machine Learning
MAE	Mean Absolute Error
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive model with eXogenous input
NN	Neural Network
OCV	Open-circuit voltage
OSRM	OpenSourceRoutingMachine
OSM	OpenStreetMap
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
SRTM	Shuttle Radar Topography Mission
SOH	State of Health
SOC	State of Charge
WN	White Noise

THESIS OVERVIEW

The thesis work is structured in five chapters. The first chapter contains a general introduction to the electric vehicle world, showing the good and bad points. Then, the motivations that encouraged to proceed with the analysis of the SOC estimation problem are described, with a focus on the objectives and contributions given.

In the second chapter, the main techniques present in literature and employed for solving the problem are illustrated. Among these, there are: model-based approaches, direct measurements, and data-driven models.

The chapter three outlines a characterization of the model-based algorithm proposed and developed. The chapter starts with a description of its workflow process, where are explained the idea that have contributed to its development. Since this model works offline, it is able to estimate the SOC associated to the selected route, before the trip begins, but it was necessary to understand how to obtain the required input data. For this reason, the second paragraph explains all the processes, functions and services used to extract these information. After, there are three paragraphs where all the formulations that allow to compute the SOC, starting from the longitudinal dynamics equation of the vehicle, are illustrated. In the sixth paragraph, the functions used to identify some parameters, needed by the algorithm to compute the output are shown. The last three parts of the chapter are dedicated: to the description of the obtained results; to the algorithm testing process in a simulated real scenario; and to the final considerations.

The thesis continuous with the fourth chapter. Here, the idea behind the development of the second algorithm are expounded. The chapter starts with an introduction to the machine learning approach for solving estimation problems. Follow a theoretical description about the neural networks and their mathematical formulation. In the third paragraph, the workflow designed to select the proper neural network model is showed; and to conclude, the analysis of the obtained results is presented in chapter four. The chapter ends with the relative conclusions.

In the fifth and last chapter, the results' comparison obtained by the two algorithms are illustrated: in particular, the description is referred to the achievements related both to a single drive cycle and to the set of 22 drive cycles. The chapter ends with the work conclusions and an overlooking of future developments associated.

INTRODUCTION

1.1 A BIT OF HISTORY ABOUT ELECTRIC VEHICLES

Although it cannot be established an exact year in which electric vehicles have been created and realized, it is possible to identify their born in the thirties of 1800, or in other words, in the first part of the 19th century. The major architects of this innovation were mainly Hungarians, as Anyos Jedlik, Scots, as Robert Anderson and Americans, as William Morrison, that, however, did not immediately come to the production of a fully electric vehicle, but they began to theorize a medium that had a battery through a series of breakthroughs. [1], [2]

The main problem of these first battery was in their inability to be recharged, which only came in the second half of the 19th century, thanks to the work of the French physicist, Gaston Planté, considered as the father of the lead-acid rechargeable battery. [1], [3]

Morrison, in particular, worked intensively on an electric vehicle model that is still a benchmark for modern electric vehicles today and, around 1890, he made it a widely sold product in America, even beating the competition of diesel cars, since, for example, the latter had a nauseating smell of fuel, hardly tolerated especially by the high society of the time, that is the main buyer. The production of electric cars reached its peak around 1912, remaining stable until 1920. In this juncture of time, electric cars were used in many areas, primarily as cars, but also as taxis and transport vehicles for goods. [2]

Subsequently, it suffered a decline caused by the lower price of diesel, thanks to the discoveries of new crude oil fields in Texas, with petrol stations that were popping up across the country and, moreover, to the concomitant diffusion of Ford's diesel cars, which were competitively priced and, above all, could travel greater distances than electric vehicle. On the other hand, the only real problem with electric vehicles was the short distance they could travel without having to be recharged, a problem that persists today. [2]

So, with the improvement of the roads and the widening of the latter, together with them there was the need, because, on the other hand, owning a car was no longer a luxury

item, to drive longer-range vehicles and so, already in the second half of 1930, electric cars disappeared from both the US and Europe to make way for the increasingly widespread diesel cars. However, in doing so, the levels of air pollution began to rise, with the consequences that, in the long run, have led to the condition we live in today. Around 1970, the increase in the price of fuel, which manifested itself mostly during the Arab oil embargo, defined the return to the need for the usage of electric machines, both in order not to depend on oil sources, and to escape the harmful effects which, over the years, they would arise due to pollution. [2]

By means of this, for example, General Motors developed, in 1973, a modern prototype of electric vehicle which could be more compatible with the urban scenario, Figure 1. [2], [4]



Figure 1. Electric car produced by GM in 1973. [4]

However, not only carmakers had produced electric vehicles, but also the NASA had contributed to the development of this vehicles; in particular, its electric Lunar rover was the first manned vehicle to steer on the moon in 1971, and that was used by the crews of Apollo 15, 16, and 17 to move along the surface of the moon. [2]

Nevertheless, the electric vehicles market didn't dominate over gasoline vehicles one, because of their low speed, usually 40-45 miles per hour, although many attempts were made in order to commercialize them as much as possible, for example, the "Citicar" launched by manufacturer Sebring-Vanguard became the sixth largest in America from the sales. [2]

It was only in the 90' that the electric cars had their real success, especially in America, probably due to the passage of the 1990 Clean Air Act Amendment and the 1992 Energy Policy Act, which were determined in wake up the population towards the importance

of a change in the way of life. In fact, both in 1996 and 1997 two cars had a huge popularity: the first car was the EV1 vehicle, manufactured by General Motors. This last vehicle had a range of about 80 miles and was able to accelerate from 0 to 50 mph in just seven seconds; the negative aspect was the high production costs, so the production was interrupted in 2001 and this car only remained a cult among the fans. [2]

The real approach towards electric cars was thus supported in 1997 with the Toyota Prius, Figure 2, which was the first hybrid car to be mass produced, owned, and so, indirectly sponsored by important celebrity figures, such as Leonardo DiCaprio and Harrison Ford. Toyota used a nickel metal hybrid battery, a technology that was supported by the Energy Department's research. [2], [5]



Figure 2. Toyota Prius, the first hybrid vehicle. [5]

The real relaunch of electric cars, between ups and downs, will only take place from the second half of the 2000s and, in particular, in 2006, when a relatively small Silicon Valley's start-up company echoed the media for its intention to produce electric cars of luxury that they could travel more than 200 miles on a single charge: this becomes reality in 2008 with the Roadster. Thus, in 2010, Tesla Motors received a 465 millions of dollars grant from the Department of Energy's Loan Programs Office, in order to establish a manufacturing facility in California, where Tesla became the biggest and most popular auto industry employer. [2]

Subsequently, also other automakers started to produce their own electric vehicles to launch on the market: in 2010, Nissan Leaf and Chevy Volt were released. The first one is an all-electric vehicle, instead the second one is the first commercially available plug-in hybrid. Therefore, today it is possible to find many affordable cars, proposed by the main automakers, both totally electric and hybrid: Tesla, for example, with his 4 models, Tesla Model 3, Tesla Model S, Tesla Model Y, Tesla Model X, is the host in the world of electricity. This is followed by the manufacturers as Audi, with its e-tron models, or

BMW, Volkswagen, up to Fiat, which launched its electric car, the Fiat 500e in the last year. [2], [6]



Figure 3. Fiat 500e.



Figure 4. Tesla model X. [6]

1.2 PROS AND CONS OF ELECTRIC VEHICLES

In the last years, particularly in the last two decades, the growing of the pollution has spread and reached high levels rapidly, up to urgently push to find solutions, in order to bring and maintain the values for polluting gases below the nominal threshold. This important issue is taken under consideration by the vast majority of developed countries and international companies, of any type and dimension. Since a big piece of contamination derives from the excessive usage of ICE (Internal Combustion Engine) cars, just think that only in Italy there are about 60 millions on people and 50 millions of cars, or by transport vehicles, that are equipped by a heat engine too, it was the automotive industry to engage, with firmly, this fight, with the attempts to reduce and to limit the greenhouse gas emission. To give a bit of statistical data, the transport sector

predominantly depends on fossil fuels and it is responsible for the 16% of greenhouse gases. [7]

In Figure 5, it is possible to have an overview of the main sources of Carbon dioxide and this demonstrates that the transport is the second major field for generation of CO₂. [8]

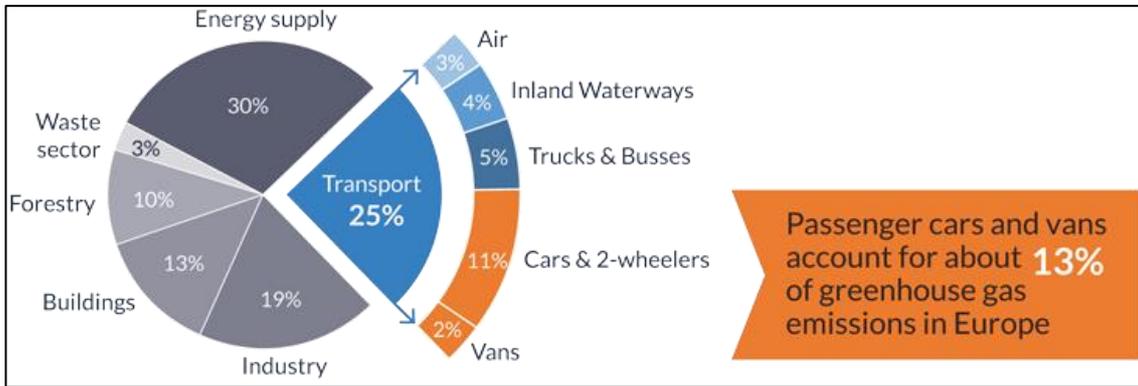


Figure 5. Diagram with sources of CO₂. [8]

As direct consequence, the major car makers, in the world and not, has decided to introduce, in their production lines, cars supplied by batteries, both of hybrid type and totally electric, also called Hybrid Electric Vehicle (HEV) and Battery Electric Vehicle (BEV), respectively. From a lot of tests conducted, it was demonstrated that the usage of electric vehicles, having no polluting emissions or over the edge, as in HEV case, influences with almost no impact in the contribution of the pollution dependent on greenhouse gases, mainly referring to Carbon dioxide, but not only. Therefore, it can be deduced that this is a valid and really feasible approach as a small, but important solution to the problem. However, it is still defined as an approach and not a solution since, despite these positive aspects, several negative points are present, one of these is the way in which the energy of the vehicle's batteries are regenerated; this last is a process characterized by a series of working mode and operations that leads to an indirect increase of the CO₂ production level. This one described is not the only negative point around the spread of the electric vehicle, nevertheless it is enough to give the idea that the conversion process, towards a totally, or also partial, green world is long and tough; moreover, it is necessary making more and more studies and research, to reach a solution with zero greenhouse emission and completely green. Besides these serious problems, the diffusion, and the approval of the electric vehicle in the daily routing is hampered by other minor issues, that however have a huge influence on the final user, the consumer. Among these minor factors, like the purchase price, the maintenance costs, the reduction of the polluting gases and the range anxiety level, the sine qua non is the last term, that according to the people ideas, is the essential factor that, today, has a major influence on the chosen of purchase or not an electric vehicle. [9]

However, it is also necessary to say how, although these drawbacks, the people are starting to feel the need for a change. This change of trend is observable going to analyze some data. Despite the world has fought versus the pandemic emergency, due to the Covid-19, in the last two years, and still today, the automotive market has experienced a deceleration in sales with a decreasing of the 27% of the sales, that corresponds to about 9.7 millions of units, the electric vehicles have been registered positive results, with a trend inversion. Figure 6 shows a diagram of the sold of EV per country in Europe in the range 2013, 2017. In Italy, in 2020, had been sold 32.358 electric vehicles, with an upgrade of the 753% with respect to the month of December in the past year, that in practical terms are about 60.000 vehicles, including both Battery Electric Vehicles and Hybrid Electric Vehicles. [10]

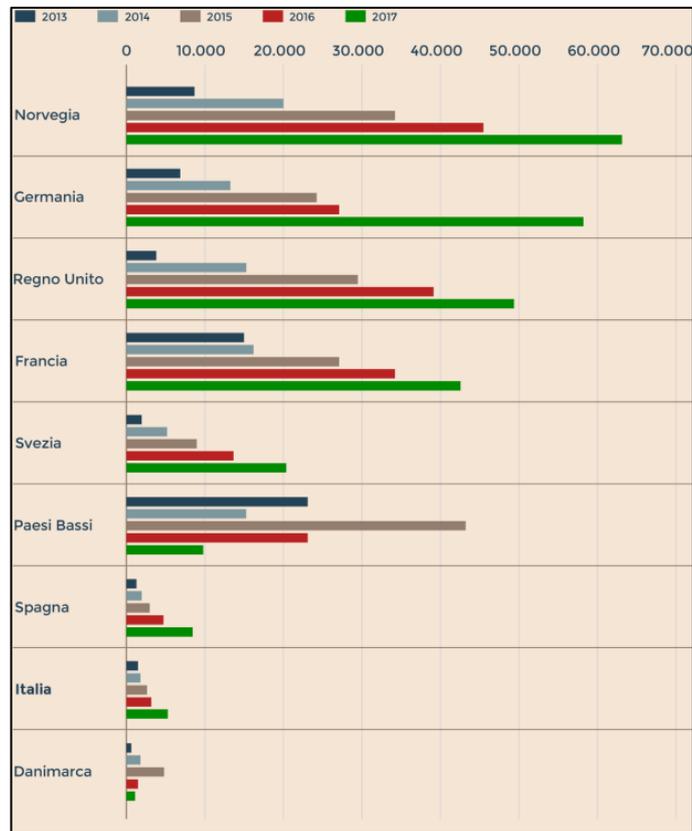


Figure 6. Sold of EV per country, in Europe, between 2013 and 2017. [11]

1.3 MOTIVATIONS

Everyday utilities, electric vehicle, industrial applications are examples of systems that use the batteries as the main source of energy. The battery is a very complex system,

characterized by a lot of non-linear behaviors, parametric uncertainties, that make challenging its modeling and analysis. One of the problems, still open today, is the estimation of the SOC, a fundamental parameter, that cannot be measured directly, and that reveals some battery performances. A good estimation can guarantee a battery protection, prevent over charge and discharge, but also improve and extend the battery life. [12], [13]

Moreover, it can be used to implement good and efficient control strategies to preserve the battery energy. [13]

For this reason, both the car makers and the battery manufacturers work severely to improve this function of the battery management system, that is a fundamental device installed, for example, in the electric vehicles. Other reasons that encourage to find an algorithm that implements an estimation, robust and accurate, of the SOC function are strictly related to the final user, the driver. One of his greatest fear is to be stranded, or in other words, to run out of the energy. A consequence of this last problem is the range anxiety. The range anxiety is a real mental state, that could hurt the electric vehicle driver, getting him a state of anxiety that borns from the fear of do not reach the final destination before the autonomy of the battery level is completely exhausted, causing the driver to perceive the driving range a lot less than it should be. [14]

As a direct consequence, the range anxiety is a real limit to the spread of the electric vehicles. This state of mind can be justified by the lack of a dense network of charging infrastructure, public and not, disseminated in the country. In the last years, several solutions are designed and evaluated to break down this phenomenon: for example, Tesla, leader company in the development of electric vehicles, has created the so called, Supercharger, Figure 7. These are charging stations, that guarantee a charging of the vehicle in shorter time. [15]

In Italy, the Enel X company, is among the main green farms that had decided to invest in this field and has introduced its own charging network stations, and moreover an own device, remotely controllable too, to be installed at home and called JuiceBox, Figure 8. [16]



Figure 7. Tesla Supercharger. [14]



Figure 8. Enel X JuiceBox. [15]

The previous one is becoming a true solution to the range anxiety problem, but obviously, is not easy to achieve in the short term. Associated with the range anxiety there is also the problem of the range estimation, that gives an idea of the driving range that a vehicle can cover with a certain charging level. In classical cars, where the heat engine is the propulsion unit, the driving range is not a problem, as in electric vehicle case, maybe since the people are able to predict the consumptions and know from what factors are influenced. For this reason, another possible solution to the range anxiety is associated to the improving of the range estimation system but it is not easy to reach. In the last years, a lot of research centers and not have invested in the development of algorithms able to give an estimation, as better as possible, of parameters that, directly or indirectly, give information about the residual level of battery state of charge (SOC). In the electric vehicle, the level of consumption can be seen from the dashboard, by referring to the state of charge, that is the equivalent indicator of petrol consumption in an ICE vehicle. So, having a good estimation of the SOC, the user will be conscious of the actual battery level and therefore, if it is necessary to plan a break to charge the vehicle, during the trip. However, to have a good, accurate and reliable range estimation system, it has been necessary to develop many models: such of those return, with a fine approximation and under certain conditions, a true energy consumption value or an approximation of the SOC of the electric vehicle, during the trip. Certainly, flesh out algorithms and models for the state of charge estimation is not easy, since the battery is a very complex system to model, and it is characterized by a lot of non-linear processes and influenced by a lot of systems present in the vehicle. So, to design an accurate model, must be considered a lot of factors, both external and internal to the vehicle and establish some working conditions. For example, have to be analyzed dynamic mechanical factors, electronic components that consume energy provided by the battery system. Therefore, it is necessary to consider the vehicle's construction features, the specification about the electric motor and all the subsystems that make an electric vehicle, but also the external temperature, the weather conditions, the typology of road, the driving style, and

many other external factors. In the next chapter, it will be discussed about what the SOC is and how could be measured.

1.4 THESIS OBJECTIVE AND CONTRIBUTIONS

As said before, the estimation of the SOC is a still open challenge, due to the complexity of the battery system, so this work is intended to give a contribute to this issue.

The aim of the thesis was to develop two offline algorithms to estimate the state of charge consumption, associated to a trip, trying so to improve the range estimation problem and provide a valid software to use in real applications. The two models were designed on two different estimation methods: the first one is developed on the physical knowledge of the BEV, so according to a model-based approach; the second one, on the availability of lot of data that ensure the usage of a data-driven technique, the Neural Network one. In both cases, however, the data used, to develop the models were the same. The data were supplied by the Bylogix S.r.l company, after being acquired from a Tesla Model 3 with a 75 kWh battery pack, using the CAN bus (Controller Area Network). The Tesla model 3, has a lithium-ion battery, with a nominal voltage of 360 V.

The CAN is the major standard network or bus used by automotive systems to exchange information and data, in robust way. The data were extracted with a frequency of 1 Hz, so the data was collected every 1 s. The total number of data was 120687, that corresponds about to three months of measurements, made between March and May of 2021, and contains the driving, the reverse, and the charging condition.

Although, for this thesis goal, it was decided to use only those information associated to the driving conditions of the vehicle, that meant to select only the cases associated to vehicle velocity greater than 0. A bit of engineering of data was performed in order to: parsing the data from the original json format, extrapolate the needed data, and found new useful information from the original ones.

The data, collected from the vehicle, were:

- Accelerator, that represents the percentage of force applied to the accelerator pedal.
- Altitude, that is the elevation values above the level of sea, in meters.
- Ambient temperature, in Celsius degree.
- Breaking, a binary values that is 1, when the brake pedal is pressed, 0 otherwise.
- Current, that is current supplied by the battery, in ampere.
- Direction, that represents the orientation of the vehicle, in compass reference system.

- GearSelected, that indicate the gear selected among Drive, Parking, Reverse and Neutral.
- GspAccuracy.
- Latitude.
- Longitude.
- SOC, that is the state of charge level read on the dashboard, in percentage.
- Timestamp, that indicates the acquiring time of the measurement.
- Uispeed, that is the velocity of the vehicle, in Km/h.
- Voltage, that is the voltage at battery level, measured in voltage.

Both the algorithms were written in Python language. The first algorithm, the model-based one, was designed in a different way with respect to the canonical one, that can be found in literature. In fact, instead of using an electrical equivalent circuit model of the battery, it was decided to develop a model of the battery electric vehicle. The main reason was that the collected data come from vehicle dashboard, and in general are associated to vehicle information and are not taken by making experiments of the battery pack, as instead it would have been necessary to perform to use the equivalent circuit battery model. To be clearer, the model takes into account all the physical equations that allow to compute the energy consumption associated to a trip; the starting point is the second Newton's Law, from which is derived the longitudinal vehicle dynamics equation. In particular, the equations implemented, starting from the power computation at wheel level, end to calculate the power exchanged by the battery. The power transition from wheels to the battery, passing through to electric motor, is performed using the efficiency conversion term η . Once the battery power was estimated, the consumed energy was derived and, to conclude, the associated state of charge is also computed. The input data needed by the algorithm are: the velocity profile, the altitude profile and the weather data. These data can be obtained, for a real application scenario of the algorithm, all from APIs and databases; for example, the velocity profile can be predicted using road information, that are obtained from OpenStreetMap service. However, the validation of the implemented equations is performed by using both real measured data, provided from Bylogix engineers, and other data obtained by APIs requests.

After checked that the implemented formulas worked fine and the error committed, in energy and state of charge consumption, was quite small, it was decided to test the algorithm in a simulated real scenario. Thus, the input data used were only the coordinates of the route planned, from which it was possible to obtain all the needed data to estimate the SOC, among which the predicted velocity profile. However, a more detailed description about the model-based estimation method developed, will be provided in chapter 3.

The second proposed algorithm was based on a data-driven approach and, in particular, it was designed using the artificial neural networks. The use of this algorithm typology was possible by the availability to use a dataset, containing about 120 thousands of data, real measured, as described above. The network implemented belongs to the Feed-Forward Neural Network architecture, and it contains: 3 hidden layer with 32 nodes for each, the output layer and the input one. Three features were used as inputs of the model: the velocity of the vehicle v [$\frac{m}{s}$], the acceleration \dot{v} [$\frac{m}{s^2}$], derived from the velocity, and the slope angle of the road θ [rad]. The last term was obtained from the knowledge of the elevation and distance profile. The target choice for the network was the battery power consumption P_{BATT} [kW]. After, the output of the network was integrated, obtaining so the energy; then, the state of charge was derived. As can be seen, the idea behind the working principle of this algorithm is simpler than the one of the first: this is due to the presence of the neural network, that is able to learn autonomously the relationship between the input and the output data. More details about the architecture used for design this second and last algorithm, and the results obtained are described in chapter 4. Obviously, also this algorithm is designed to work offline. So, the needed data (velocity, acceleration and road slope angle) are extracted starting from the knowledge of the geographic coordinates of the selected route, with the support of web services and APIs, using the same exact functions implemented for the model-based algorithm.

SOC MEASUREMENTS AND ESTIMATIONS TECHNIQUES

One of the parameters that could be used to improve the range estimation is the state of charge, since this lets the driver be able to check the battery level during the normal usage of the vehicle or to schedule a recharge stop.

The SOC cannot be measured directly, since it cannot be acquired by using sensors and so it can be derived indirectly, or better, has to be estimated starting from other variables, that instead are measurable. [17]

A good estimation of the state of charge is one of the most important targets to have a good Battery Management System (BMS), but since it depends by a lot of factors, as the application where the battery is used, the battery type, the external factors that affect the working conditions of the system in which it is applied and so on, it is not easy to obtain it. If we think about the automotive field, the growth, and the spread of the battery electric vehicles (BEV), of hybrid electric vehicles (HEV) and so, in general, of the Electric vehicle (EV), have forced the major car makers to invest a lot of resources and moneys on development of prototypes of electric vehicles and, in particular, to focus on the growth on the battery management system and battery pack, that is, maybe, the principal components of an electric vehicle. In fact, the increasing of the driving range, both with the develop of new batteries and with the new range estimation algorithm among the main aim of car makers. For these reasons, in the last 10 years, hundreds of research and development works have been developed to accomplish this problem, and so to obtain improving results in terms of accuracy and reliability in the estimation of the SOC. Nevertheless, the battery is a very complex physical system with a lot of non-linear behaviors, with high dynamic operating conditions that make unpredictable the charging and discharging phase, so it is very hard to study its working principles and therefore it is not easy to estimate the value of the SOC. [17]

In literature exist different methods designed to have a quite good approximation of the SOC value, some of these are based on measurement techniques, so called direct method, some other on model-based approaches and other ones on data-driven methods, but all of these are based on particular working conditions.

In this chapter, it will be evaluated several estimation techniques, also used in real applications by battery manufactures, or car makers company. Starting with the definition of the state of charge, there are listed three macro categories for the SOC estimation, that are: the direct measures, the model-based, and the data-driven.

2.1 SOC DEFINITION

A possible definition of the state of charge (SOC) could be “the percentage of the releasable capacity $C_{releasable}$ relative to the battery rated capacity C_{rated} , given by the manufacturer.” [18]

$$SOC(t) = \frac{C_{releasable}(t)}{C_{rated}} \cdot 100 \quad [\%] \quad (2.1)$$

where:

- $C_{releasable}$ [KWh], is the released capacity when the battery is discharged.
- C_{rated} [KWh], is the nominal battery capacity, given by the manufacturer, and represents the maximum amount of charge that can be stored in the battery. [12]

The state of charge can be expressed in percentage values, as a number between 0 and 100, or by a value between 0 and 1.

A value of this quantity equal to 100% indicates a full charge battery, while a value of 0% that it is empty, Figure 9.

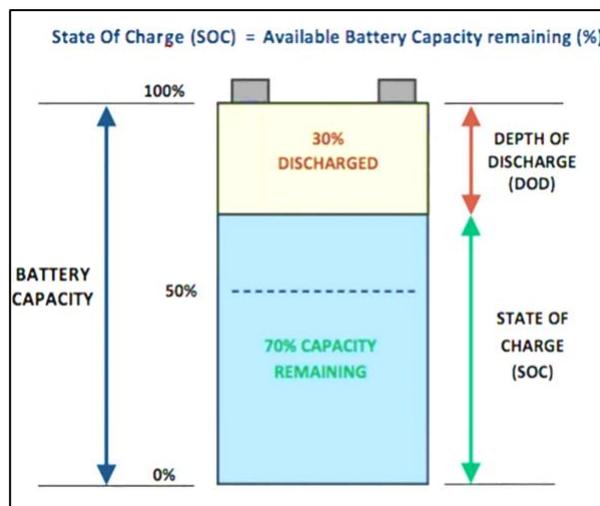


Figure 9. State of charge representation. [9]

Not only the state of charge but also the state of health (SOH) or the depth of discharge (DOD), could be measured, however this work is focalized on the state of charge coefficient. For purely informative purposes the definition of the SOH and DOD are given [18]:

$$SOH(t) = \frac{C_{max}(t)}{C_{rated}} \cdot 100 \quad [\%] \quad (2.2)$$

$$DOD(t) = \frac{C_{released}(t)}{C_{rated}} \cdot 100 \quad [\%] \quad (2.3)$$

where:

- C_{max} [KWh], is defined for new batteries and its value declines during the usage.
- $C_{released}$ [KWh], is the capacity discharge by any amount of current.

Exist also relationships between DOD, SOH, and SOC, as it can be possible to observe by the next two equations [18]:

$$SOC(t) = 100\% - DOD(t) \quad [\%] \quad (2.4)$$

$$SOC(t) = SOH(t) - DOD(t) \quad [\%] \quad (2.5)$$

To measure, or better, to have a good approximation of the SOC consumption level, different techniques have been developed in these years, but obviously all of these have pros and counters. Examples of classification for the SOC estimation, created by analyzing only certain methods present in literature, are showed in the Table 1.

Direct Measurement		Model-based		Data driven	
1.1)	Coulomb Counting method	1.1)	Equivalent circuit model	1.1)	Machine Learning
		1.2)	Estimation filter		
1.2)	OCV method				

Table 1. Classification of state of charge estimation techniques.

2.2 DIRECT MEASUREMENT METHODS

The direct measurement methods, as can be understood from the name itself, allow to estimate the state of charge directly from battery measurements.

These two methods use direct sensor's measurements and also look-up tables derived ad-hoc experiments. In the next two sub-paragraphs, are described the two most used techniques, that are the Coulomb counting method and the OCV or open circuit voltage.

2.2.1 COULOMB COUNTING METHOD

The Coulomb Counting method, also called current integration, or Ampere Hour Counting method, is an estimation technique based directly on the definition of the SOC, Figure 19. It is, maybe, the most simple but general approach to solve the problem and, also for this reason, it is a common technique used in real applications.

It is based on the integration, over the period time, in which the battery is used, of the measured discharging current, I [A].

In formula, it is expressed according to equation 6. [18]

$$SOC(t) = SOC(t_0) - \left(\frac{1}{C_{rated}} \cdot \int_{t_0}^{t_0+t} I d\tau \right) \cdot 100 \quad [\%] \quad (6)$$

where:

- $SOC(t_0)$, is the initial value of the state of charge with value in [0-100].
- C_{rated} [Ah], is the nominal capacity of the battery.
- I [A], is the current that flows on the battery for the period of time of measurement.

Note that the time must be converted from seconds to hours.

Obviously, the presented method has both advantages and disadvantages.

The advantages are that it is simple to be implemented and to be used. Basically, a good current sensor and an integrator system are enough to make the estimation and then it is quite inexpensive, moreover it can be used for online measurements.

Nevertheless, the disadvantages are many more: the first one is that a good accuracy depends on the goodness of the current sensor and also on the accurate estimation of the initial value of the SOC.

The second that, since the value of the nominal capacity of the battery changes over the year, with the usage of the vehicle battery, it is needed to know the value of the nominal capacity in that moment, and so it is necessary to estimate it.

The third fact is that the state of charge consumption depends also by intrinsic characteristics, and by the different operating conditions of the battery.

The last drawback is associated to the usage of the integrator, if the current measurement is affected by error or by noise, this will be amplified and so the estimation accuracy will

decrease. To solve this last error, in general the measurement system has to be recalibrated at each load cycle. [18]

Obviously, the formula presented here needs to be adjusted with more considerations; in fact, the estimation, even if it is based on this simple formula, depends on several other factors as the losses of the battery pack, the temperature effects, the cycle life of the battery and many others. [18]

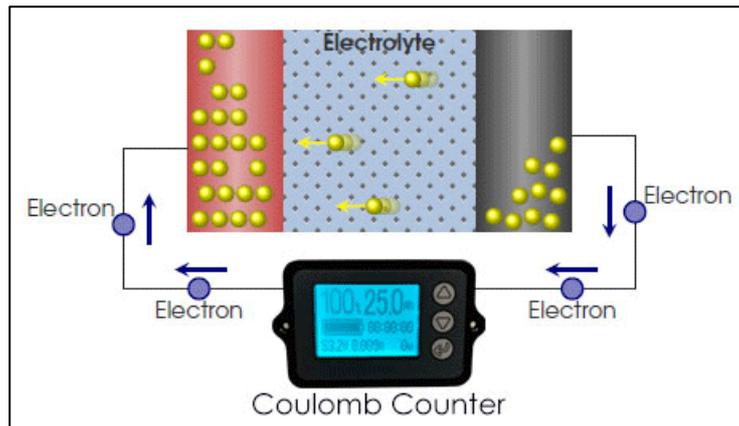


Figure 10. Example of Coulomb Counter sensor. [10]

2.2.2 OPEN CIRCUIT VOLTAGE METHOD

The Open Circuit Voltage method is based on data obtained by directly testing the battery system, under certain operating conditions. The idea behind this method is that every battery has this common behavior: the voltage at its terminals increases or decreased according with its state of charge level. In general, higher voltage corresponds to a battery full charged, vice versa lower voltage to battery empty. [19]

As it is possible to see from Figure 11, the relationship depends by the battery technology used in the two examples, the Lead-Acid and the Lithium-Ion.

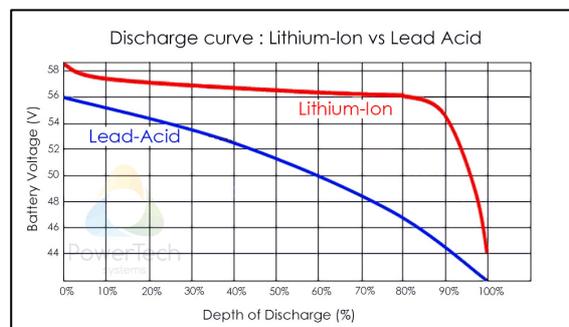


Figure 11. Comparison of discharge curve between a Lead-Acid and lithium-Ion. [10]

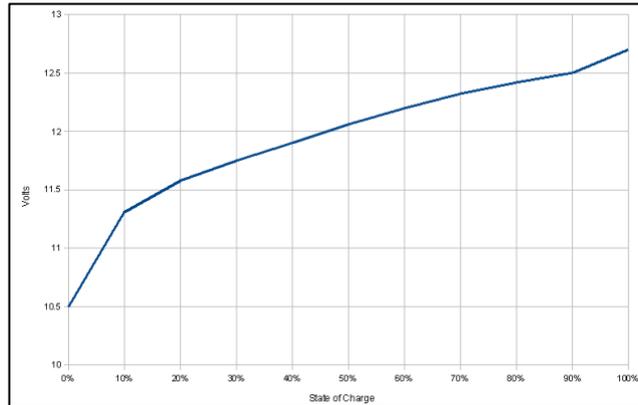


Figure 12. State of Charge versus OCV in Lead-Acid battery. [20]

From the Figure 12, it could be evidenced a linear behavior, in the lead-acid battery, between the SOC and the OCV, given by [12]:

$$V_{OC}(t) = a_1 \cdot SOC(t) + a_0 \quad [V] \quad (7)$$

where:

- $SOC(t)$, is the battery state of charge at time t .
- $V_{OC}(t)$ [V], is the open circuit voltage.
- a_0 , is the battery terminal voltage when $SOC = 0\%$.
- a_1 , is obtained when $SOC = 100\%$.

This method so estimates the SOC value by using look-up tables that describe the battery discharge curve in terms of Open-Circuit Voltage (OCV) versus SOC.

A typical behavior of the OCV with respect to the SOC, in Lithium-Ion batteries, is showed in Figure 13, where it is evidenced a non-linear function.

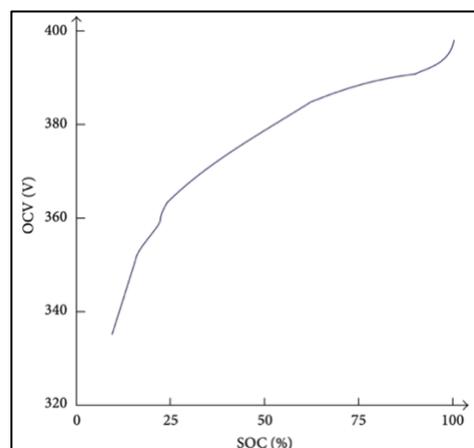


Figure 13. State of Charge versus OCV in Lead-Acid battery. [20]

The here described method can be used only when the battery is disconnected from the loads for a period longer than two hours, but, obviously, this is one of the main drawbacks of the model and so cannot be used in automotive case. The standard methodology to find the relationship between the OCV and SOC is characterized by a pulse load application, on the Li-ion battery, and then wait until the battery reaches the equilibrium. [12]

The negative point of the following technique is that it can be performed only offline, since the measurements, to create the tables, must be executed with the vehicle that is in rest condition and the battery works under specific operative conditions, as said before. Note that this relationship is not equal for all the batteries and, moreover, depends also by the battery temperature and so it is not sure that the estimation method works accurately. [12]

In automotive industries, the widely used batteries belong to the group of the lead-acid one, even if Tesla uses battery packs with many modules, each contains about hundreds of small Lithium-Ion cells, which are sourced by the Japanese tech giant, Panasonic.

2.3 MODEL-BASED METHODS

In this paragraph are described several methods that guarantee an estimation of the state of charge, based on physical knowledge of the problem and so on the so-called, Model-based approach. For this reason, it is necessary to create a battery model, as accurate as possible, and to simulate its behavior in a simulation environment, like MATLAB or Simulink [MathWorks].

Despite these methods are more complex to implement, due to the fact that it is very wasteful to replicate in a simulator a battery system, the results obtained are better and much closer to the real one. In addition, it is necessary to establish several conditions under which the model will work.

This approach, in general, could improve the level of SOC estimation accuracy, but obviously depends a lot on the modelling grade of the battery system. General approaches are based on equivalent circuit model of battery systems, as in Figure 14, and on Kalman filter applications that let the opportunity to estimates, in real time, or to forecasting, output and model parameters that change in time.

2.3.1 EQUIVALENT CIRCUIT MODEL FOR BATTERY

The electrical equivalent circuit model is used when it is necessary to estimate some battery parameters, like the state of charge, or in general to know the dynamic response

of the battery in real time, but a lot of details and specifications about the system are known.

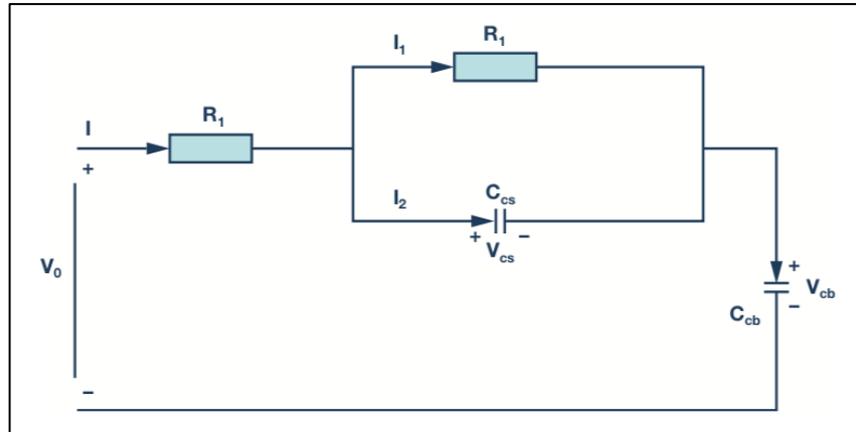


Figure 14. State of Charge versus OCV in Lead-Acid battery. [20]

Starting from the Figure 14, where it is displayed an equivalent model for lithium-ion battery with a single RC network, it is possible to estimate the SOC and other parameters that are influenced by temperature or other ones.

In the previous equivalent circuit, can be identified the following parameters [18]:

- C_{cb} , is the bulk capacitance, that represents the battery pack storage capacity.
- C_{cs} , is the surface capacitance, that represents the battery diffusion effects.
- R_i , is the internal resistance.
- R_1 , is the polarization resistance.
- V_{cb} , is the voltage across the bulk capacitor, or open-circuit voltage.
- V_{cs} , is the voltage across the surface capacitor.
- V_0 , is the terminal voltage across the battery pack.
- I , is the terminal current in the battery pack.

To model better some complex non-linear behavior, other RC networks can be added, but the computational time and the complexity of the simulations increase a lot. To use a model like this one, it is necessary to conduct some experiments to obtain data to create look-up tables, that can be used to identify some network parameters such that R_1 , R_i , C_{cs} , and C_{cb} . For example, the tests performed are the OCV ones, where the battery is discharged by injection of current pulses. [18]

In first analysis, it can be assumed that the parameters of the model depend by the voltage and temperature, so are not constant in time. To obtain these values, it can be

used different System Identification techniques, but it has to be known also some data coming from experiments performed on the battery.

The model in Figure 14 is governed by the following equations [18]:

$$\dot{V}_{cb} = \frac{1}{C_{cb}} I \quad [V] \quad (8)$$

$$\dot{V}_{cs} = \frac{1}{R_1 C_{cs}} V_{cs} + \frac{1}{C_{cs}} I \quad [V] \quad (9)$$

$$V_O = V_{cb} + V_{cs} + IR_i \quad [V] \quad (10)$$

Introducing the relationship between the OCV and the SOC, that can be approximated a linear one, as written in equation 11.

$$V_{cb}(t) = a_1 \cdot SOC(t) + a_0 \quad [V] \quad (11)$$

Coefficients a_1 and a_0 change with the ambient temperature and the battery SOC.

Working with the equations 8 and 11, it can be written the variation of the SOC as:

$$S\dot{O}C(t) = \frac{1}{a_1 C_{cb}} I \quad (12)$$

$$\dot{V}_{cs} = -\frac{1}{R_1 C_{cs}} V_{cs} + \frac{1}{C_{cs}} I \quad [V] \quad (13)$$

$$V_O = a_1 \cdot SOC(t) + a_0 + V_{cs} + IR_i \quad [V] \quad (14)$$

At this point, putting all these equation in matrix form, it can be possible to obtain the state space form, in continuous time, of the model:

$$\begin{bmatrix} S\dot{O}C \\ \dot{V}_{cs} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{R_1 C_{cs}} \end{bmatrix} \cdot \begin{bmatrix} SOC \\ V_{cs} \end{bmatrix} + \begin{bmatrix} \frac{1}{a_1 C_{cb}} \\ \frac{1}{C_{cs}} \end{bmatrix} \cdot [I] \quad (15)$$

$$V_O - a_0 = [a_1 \quad 1] \cdot \begin{bmatrix} SOC \\ V_{cs} \end{bmatrix} + [R_i] \cdot [I] \quad (16)$$

Since the system found is made by linear and non-linear equations, due for example, to V_{cb} , that is a function of SOC , it is possible to conclude that the system is non-linear. By the state space form, equations 15 and 16, can be evidenced that the states of the system are $x_1(t) = SOC$ and $x_2(t) = V_{cs}$, the input $u(t) = I$, while the output is $y(t) = V_O$.

A methodology, widely used, to estimate the SOC, in the lithium-ion battery packs, is the Kalman filter.

Note that the equation to pass from the battery pack to a single cell, and vice versa, are the following ones:

$$I = \frac{I_{pack}}{N_{parallel} \cdot N_{module}} \quad (17)$$

$$V_0 = \frac{V_{pack}}{N_{series} \cdot N_{module}} \quad (18)$$

$$C_{cb} = \frac{C_{nom}}{N_{parallel}} \quad (19)$$

where:

- $N_{parallel}$, is the number of cells in parallel.
- N_{series} , is the number of cells in series.
- N_{module} , is the number of battery modules present in the battery pack.

These data can be extracted from the battery manufacturer specifications.

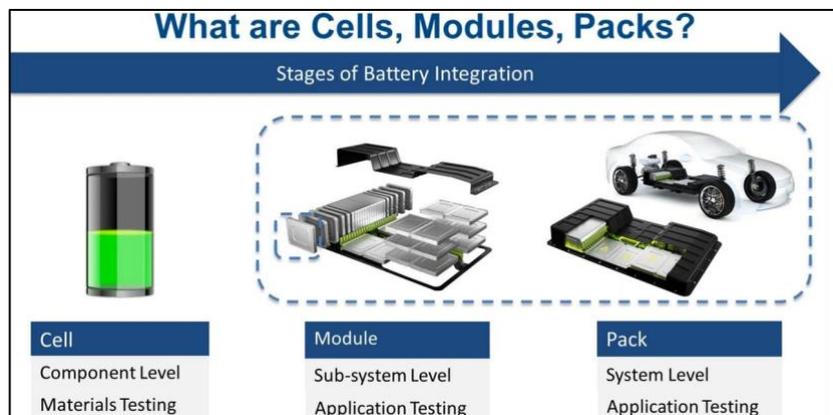


Figure 15. Stages of battery integration on an electric vehicle. [9]

2.3.2 ESTIMATION FILTER

In this paragraph, it will be described a possible estimation solution, that could be used to estimate the SOC of the battery pack, that is the Extended Kalman filter (EKF). Before talking about EKF, it is necessary to introduce, briefly, a bit of estimation theory. This is

a branch of statistic, focalized on the estimation of parameters that cannot be measured directly, using random data acquired and mathematical algorithms. [21]

In addition, the goal of the parametric estimation problem is to increase the quality information, from measured signals, going to reduce the noise and augment the robustness of the system in general. [22]

The estimation of the state is a problem that can be divided into three parts: the physical system, the measurement system, and the state estimator, according to Figure 16. Note that both the physical system and the measurement system are affected by errors, simulating a real situation.

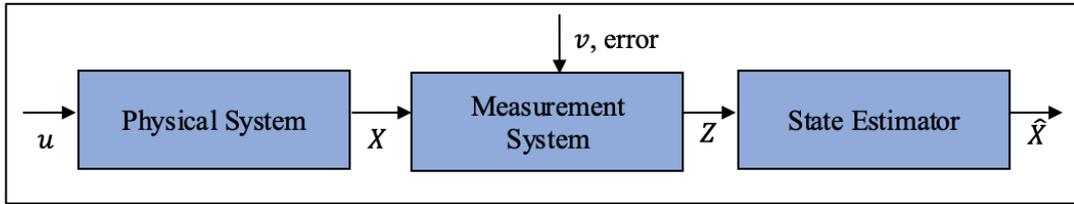


Figure 16. Block diagram for state estimation problem, where \mathbf{u} is the input data vector, \mathbf{X} is the true state vector of the system, \mathbf{Z} is the measurement vector, \mathbf{v} is the observation vector, $\hat{\mathbf{X}}$ is the estimated state vector. [21]

The Kalman filter is a state estimation algorithm, introduced by Rudolf Emil Kalman, in 1960, that published a recursive solution to the discrete-data linear filtering problem. The filter produces an estimation of variables that cannot be measured or that are inaccurate and uncertain. Moreover, it can be used for four purposes: as one-step prediction problem, as multi-step prediction problem, as filtering problem, and regularization problem. Its selling point is that it can be easily implemented in a programming software and under certain conditions reach the optimal solution, that is, in the filtering case, to obtain a state estimated vector $\hat{\mathbf{X}}$ as close as possible to the real one \mathbf{X} .

Starting from a state space model of a dynamical system \mathcal{S} , in discrete-time, linear time-invariant (LTI) form, it can be written according to the following two equations [23], [24]:

$$x(k+1) = Ax(k) + Bu(k) + w(k) \quad (20)$$

$$y(k) = Cx(k) + v(k) \quad (21)$$

where:

- $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, $C \in \mathbb{R}^{q \times n}$ are, respectively, the system matrix, the input matrix, and the output matrix, to be known.
- $x \in \mathbb{R}^n$, is the system state vector.
- $u \in \mathbb{R}^p$, is the input state vector, to be known.

- $y \in \mathbb{R}^q$, is the output measurement vector, to be known at any time.
- $w \in \mathbb{R}^n$, is the system noise vector.
- $v \in \mathbb{R}^q$, is the measurement noise vector.
- k , is the sampling time, $k = 1, 2, \dots, N$.

The two main assumptions of the KF are that w and v are white noise with zero mean value, known variance and, in addition, are independent with each other, so $p(w) \sim WN(0, Q)$, $p(v) \sim WN(0, R)$, and $V_{QR} = 0$; where $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{q \times q}$ are the system noise covariance matrix, the measurement noise covariance matrix, respectively and $V_{QR} \in \mathbb{R}^{n \times q}$, [23], [22], [24].

Moreover, the initial state vector $x(1)$ has to be identified as an unknown random vector $x(1) \sim (\bar{x}_1, P_1)$, where $\bar{x}_1 \in \mathbb{R}^n$ and $P_1 \in \mathbb{R}^{n \times n}$, and the last as to be uncorrelated with w and v . [23], [24]

Supposing to analyze the one-step predictor, a particular form commonly used, since provides a numerically reliable formulation of the equations used in the original form of the KF problem, is the predictor/corrector form, Figure 17, where the transition from $\hat{x}(N|N-1)$ to $\hat{x}(N+1|N)$ is performed in two steps [15]:

- In the first step, the filtered estimate $\hat{x}(N|N)$ is derived from $\hat{x}(N|N-1)$, using the Kalman filter gain matrix $K_0(N)$.
- In the second step, instead, the one-step prediction $\hat{x}(N+1|N)$ is derived updating $\hat{x}(N|N)$.

This algorithm, after being implemented, can be used both for one-step prediction and for filtering problems. From an equation point of view, it is based on two parts: the predictor, that contains time update equations, and the corrector (or filtering), that instead contains measurement update equations [24], as described below:

$$\textbf{Predictor: } \begin{cases} \hat{x}(N+1|N) = A\hat{x}(N|N) + Bu(N) \\ P(N+1) = AP_0(N)A^T + Q \end{cases}$$

$$\textbf{Corrector: } \begin{cases} K_0(N) = P(N)C^T[CP(N)C^T + R]^{-1} \\ \hat{x}(N|N) = \hat{x}(N|N-1) + K_0(N)[y(N) - C\hat{x}(N|N-1)] \\ P_0(N) = [I_n - K_0(N)C]P(N) \end{cases}$$

where:

- $K(N) \in \mathbb{R}^{n \times q}$, is the one-step Kalman predictor gain matrix.
- $K_0(N) \in \mathbb{R}^{n \times q}$, is the Kalman filter gain matrix.
- $[y(N) - C\hat{x}(N|N-1)] = e(N)$, is the innovation term.

- $P(N) \in \mathbb{R}^{n \times n}$, is the prediction error variance matrix of the state.

Note that, the previous equations are true under the hypothesis that the noise terms are uncorrelated with each other ($V_{QR} = 0$) and $R > 0$, since in this way the Kalman predictor gain matrix can be written as:

$$K(N) = [AP(N)C^T + V_{QR}][CP(N)C^T + R]^{-1} = AK_0(N) \quad (22)$$

The matrix P is computed by solving recursively the Difference Riccati Equation (DRE), which can be written in this form:

$$P(N + 1) = AP(N)A^T + Q - K(N)[CP(N)C^T + R]K(N)^T \quad (23)$$

Note that high values of P indicate high levels of uncertainty, vice versa for low values. In addition, the initialization of $P(1) = P_1$ and of $\hat{x}(1|0) = \bar{x}_1$ must be done to proceed with the algorithm. [23], [24]

For the application of the KF, the assumptions made have to be respected, nevertheless, even if it is not simple to satisfy all together, the results are good.

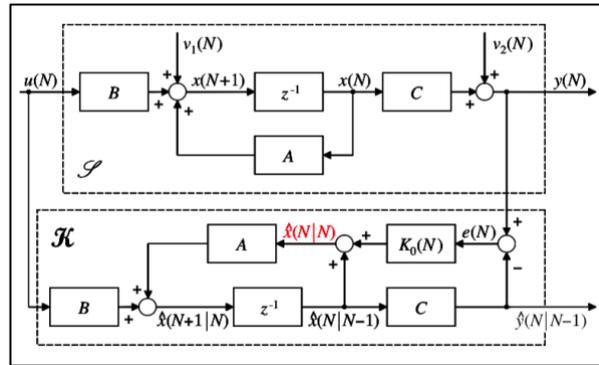


Figure 17. Block diagram of the Predictor/Corrector form of the Kalman filter algorithm. [24]

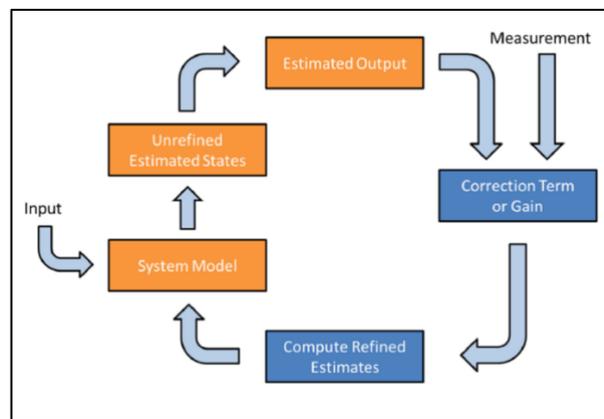


Figure 18. General review on Predictor/Corrector form of the KF. [13]

Since to estimate the SOC the battery model found is non-linear, the Kalman filter is not suitable to reach the target, and so, it must be used the non-linear version of the KF, or in practical terms, the so-called Extended Kalman filter (EKF). From here below, a general description about the EKF is done.

A non-linear, discrete time, time-variant, dynamic system \mathcal{S} , without external input u , can be described by the following two equations [24] :

$$x(k+1) = f(k, x(k)) + w(k) \quad (24)$$

$$y(k) = h(k, x(k)) + v(k) \quad (25)$$

where:

- $x \in \mathbb{R}^n$, is the system state vector.
- $y \in \mathbb{R}^q$, is the output measurement vector, to be known at any time.
- $w \in \mathbb{R}^n$, is the system noise vector.
- $v \in \mathbb{R}^q$, is the measurement noise vector.
- k , is the sampling time, $k = 1, 2, \dots, N$.

Also in this case, the two main assumptions are that w and v are white noise with zero mean value, known variance and, in addition, are independent with each other, so $p(w) \sim WN(0, Q)$ and $p(v) \sim WN(0, R)$, where $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{q \times q}$ are the system noise covariance matrix, the measurement noise covariance matrix and $V_{QR} = 0$. [23], [24]

Moreover, the initial state vector $x(1)$ has to be identified as an unknown random vector $x(1) \sim (\bar{x}_1, P_1)$, where $\bar{x}_1 \in \mathbb{R}^n$ and $P_1 \in \mathbb{R}^{n \times n}$, and the last as to be uncorrelated with w and v . [23], [24]

In addition, in this case, we have also two non-linear functions, $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^q$. [24]

The EKF, gives better result with respect the canonical non-linear KF, since the linearization of the functions $f(k, x(k))$ and $g(k, x(k))$ happens around the last estimated state $\hat{x}(N|N-1)$, instead of the nominal movement $\bar{x}(k)$.

This linearization, is translated in the following matrix computation [24], [25]:

$$\hat{A}(N|N-1) = \left. \frac{\partial f(k, x)}{\partial x} \right|_{\substack{k=N \\ x = \hat{x}(N|N-1)}}$$

$$\hat{C}(N|N-1) = \left. \frac{\partial h(k, x)}{\partial x} \right|_{\substack{k=N \\ x = \hat{x}(N|N-1)}}$$

Let \hat{A} , \hat{C} , the EKF algorithm is stated here:

$$\text{Predictor: } \begin{cases} \hat{x}(N+1|N) = f(N, \hat{x}(N|N)) \\ P(N+1) = \hat{A}(N|N-1)P_0(N)\hat{A}(N|N-1)^T + Q \end{cases}$$

$$\text{Corrector: } \begin{cases} K_0(N) = P(N)\hat{C}(N|N-1)^T[\hat{C}(N|N-1)P(N)\hat{C}(N|N-1)^T + R]^{-1} \\ \hat{x}(N|N) = \hat{x}(N|N-1) + K_0(N)[y(N) - h(N, \hat{x}(N|N-1))] \\ P_0(N) = [I_n - K_0(N)\hat{C}(N|N-1)]P(N) \end{cases}$$

Note that this filter is not optimal, but it is commonly used in real applications, like in mobile robot localization, or for GPS and so on. The drawback of the EKF is that, from a computational point of view, it is more demanding than the linearized case, since the linearization is computed online, going to linearize the system around the state estimate provided by the predictor at previous step, and so, the computation of the matrices (\hat{A} , \hat{C} , K , P_0 , and P) is performed in run-time. [24], [25]

In KF instead, the matrices K , P_0 and P are computed offline. Another difference is that the Kalan filter will converge, instead the extended version may diverge, due to the wrong linearization. [25]

2.4 DATA-DRIVEN METHODS

The last group of analyzed methods are based on the data-driven approach. This is a new methodology that began to spread with in concomitance with the Big Data era. As it can be realized by the name itself, this technique is founded on the collection of hundreds of data, and in general, the achieved results are better as bigger are the number of available data. These methods have the goodness that they do not need a physical model to obtain good final results. Among the methods, that can be developed with a data-driven approach, there is the Machine Learning one, that includes Support Vector Machine, Fuzzy Logic, Fuzzy Neural Network, Artificial Neural Network, Recurrent Neural Network and many others. [12]

The usage of these algorithms, in particular the Neural Networks with all its sub-algorithms, have been a great acceptance from the researchers' world community, thanks to their capacity to be adapted to non-linear systems. Although, this matter is still studied today, in order to find demonstrations and explanations for the impressive results returned. Talking about the thesis target, in the estimation of the state of charge consumption, this approach is widely used, but obviously it is necessary, as said before, having by hand a collection of many data, and despite this, the results are not perfects, due to the high complexity of the problem. The traditional path followed by the R&D

centers, is to try to estimate the state of charge starting from measurable variables of the battery system, such as the voltage, the current and the temperature. Due to the presence of non-linear behaviors, difficult to reproduce in the equivalent models, many studies are directed to use a data-driven methods, that try to learn from the relationship between input parameters and the state of charge, these complex relationships. [17]

Among the different works in the literature, the most use data that are measured directly from the battery pack, through precise experiments; in addition, some of these measurements are made directly in individual cells when, of course, this is possible.

It should also be noted that the works are almost never generally applicable, but depend on one, or more, specific operating conditions.

For example, there is a paper where is describes a model for the estimation of the SOC, characterized by a Nonlinear Auto Regressive network with exogenous inputs (NARX), that generates as output the voltage and, connected in series with a feed-forward network that returns as outcome the SOC; the temperature and the current of the single lithium-ion battery were used as inputs. [26]

Among the various work aimed at developing an artificial intelligence model for estimating the SOC, reference is made to the S. Bockrathwork work, where an algorithm based on a recurrent neural network (RNN) architecture was developed; in particular an LSTM architecture was implemented and trained with actual data taken from a lithium-ion battery. The algorithm achieves a RMSE error of 5.0%. [27]

In literature, there are several papers that, using artificial neural networks, point to predict the SOC level of consumption of a lithium-ion battery pack. In particular, it is possible to find studies that achieved 2.6% of Mean Square Error (MSE) error [6], others about 1% [14,15], and some authors that reached $e-04$ MSE [16].

Since one of the two algorithms, developed in this work, was designed following a data-driven approach (in particular, it contains an Artificial Neural Network model), an extensive and detailed description about this estimation technique and this model will be described, in chapter 4.

It is possible to anticipate that the artificial intelligence algorithm developed in this paper does not follow the classic approach recommended by the other works present in literature. In fact, the inputs used are: speed, acceleration and road slope angle, but not the standard voltage, current and temperature. This is a direct consequence of the approach followed to develop the estimation algorithm for the problem resolution, which is different from the one, normally, adopted.

A NOVEL MODEL-BASED APPROACH FOR OFFLINE SOC ESTIMATION

In this chapter the design part that has supported the development of the first algorithm, realized and proposed to solve the range estimation problem is described. The solution provided here is found on a model-based approach, that, however, it is different with respect the standard methods realized, as the ones described in chapter 2. In fact, the core of the resolution is the physical model of the problem, in terms of power and energy dissipated by the vehicle during the driving conditions.

By means of this approach, basic and simple formulas are used to compute an estimation of the SOC consumption, in order to evidence the goodness of the problem also using underlying formulations without going into the deep in modelling the problem.

The flowline pursued to develop the algorithm has been the following: starting from real measurements data, extracted from a Tesla Model 3 with a 75 kWh battery pack and provided by Bylogix S.r.l company, it was possible to compute the electric power consumed by the vehicle during the travelling time. This power was obtained by multiplying real measurements of voltage (V) and of current (A).

At this point, the algorithm was designed with the idea of estimate as best as possible the consumption of electrical power during the trip, using as input data, instead of real voltage and current measurements, only the point coordinates that make the route. By these coordinates, a lot of information were extracted, like the weather, distance, altitude, and road data, where some of them were used to obtain a prediction of the velocity profile, which was a fundamental working point for the model. Combining all these data and using physical formulations, it was possible to estimate the consumption power at wheels level required to complete the trip. This result was then used to calculate an estimation of the electrical power and so of the energy usage as well as of the state of charge. A comparison between the real SOC, given by the set of measured data, and the estimated one is made to validate the algorithm. Note that, before the testing process, all the formulations implemented had to be verified, and for this task were used the data provided by the company; for example, instead of using the velocity profile realized by the algorithm, it was used the velocity profile extracted by the measurements.

In the next paragraphs are described all the functions implemented to guarantee the algorithm working conditions; moreover, details about these features are illustrated. A short presentation of what the next paragraphs contains is made below here.

The first paragraph 3.1 contains a short overview of the problem with a focus on the main steps characterizing the solution developed.

In the sub-paragraphs from 3.2.1 to 3.2.7, it will be described in detail all the main data needed by the software as input; in addition, will be provided a clearer and more complete view on how the data were extracted and of the solutions adopted to achieve these, in terms of implemented functions.

In paragraphs 3.3, 3.4, and 3.5 will be illustrated all the equations and physic concepts used to obtain the estimation output data, such as battery power, the energy consumption, and the state of charge.

Then, in paragraph 3.6, a detailed characterization of the techniques used in the algorithm to obtain different parameters' values, needed to be adopted in the problem equations, will be shown.

In paragraph 3.7, an analysis of the algorithm results with respect to a real drive cycle, in terms of accuracy in the estimation of the SOC, is provided. Moreover, a statistical analysis is performed to check the general algorithm behavior with respect to 22 different drive cycles used as test. To conclude, in 3.8 and 3.9, respectively, are described the testing of the model in a simulated real scenario and the conclusions.

3.1 DESIGN OF THE ALGORITHM

To achieve this goal, estimation of the state of charge level of consumption during a trip, an algorithm was written in Python language.

The algorithm was developed with the idea to work in offline mode, so before the trip begins it returns data information about the trip itself, as the duration in time, the estimation of the energy consumption to complete it, but also graphs that show the estimated consumption energy and the SOC. As said, this is a baseline algorithm, designed to give a first general idea on the consumptions that a battery electric vehicle could have to complete a trip. A briefly description on how the algorithm operates is made below and represented in Figure 19.

The algorithm takes as input the velocity profile, the weather data and the altitude profile; after, several functions are implemented to extract other useful data and parameters to use in the algorithm, such as road slope angle, weather data and others. At this point, the conversion from the wheel side power consumption to the battery one

is realized and, integrating this last quantity, the energy is achieved. To conclude, the SOC level of consumption is derived from the knowledge of the energy.

If the algorithm is used in real life application, the working principles are the same, the only difference is that the velocity has to be predicted; this is done by combining road information and weather data. All the input data, have to be obtained starting by the knowledge of the route coordinates, in latitude and longitude format. [27], [28]

The main operations performed by the algorithm, in a real scenario, is listed here:

1. Select and extract the coordinates of the selected route, using the routing web-service (OpenSourceRoutingMachine).
2. Makes a request to Open Street Map (OSM) and extract road information.
3. Extracts information on the weather.
4. Combines the extracted data to predict a velocity profile.
5. Extracts information on the elevation profile.
6. Computation of the battery power consumption.
7. Estimation of the energy consumption and of the SOC.

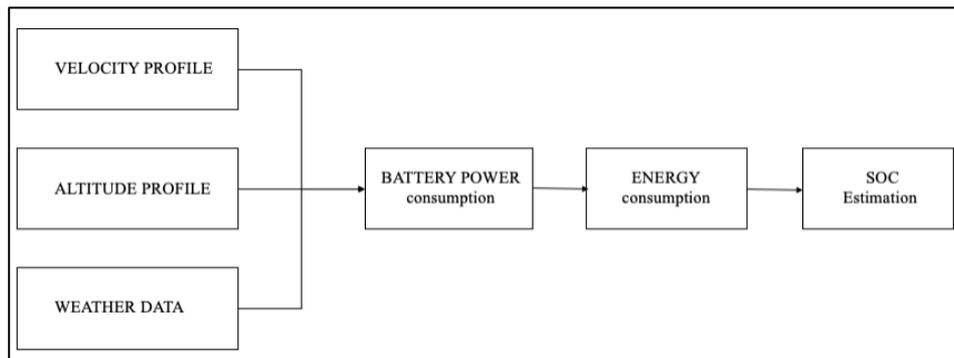


Figure 19. Working flow of the model-based algorithm for SOC estimation, in real application scenario.

3.2 INPUT DATA OF THE ALGORITHM

The main data used by the algorithm are the road information data, the elevation data, the distance data, and the weather data.

All these were obtained starting from the waypoints' coordinates, that are the input of the model. The coordinates of the route waypoints, expressed in terms of latitude and longitude, are stored in a file, whose extension is .gpx.

Another important data the algorithm needs to use, is the velocity profile that has to be predicted offline. In this chapter the velocity prediction data will be used only when is

made the simulation of the algorithm's working conditions in a real scenario. In all the other cases, the velocity used will be the one measured and provided in the dataset. In the next sub-paragraphs of this paragraphs, it will be described how these data were extracted and how were used in the algorithm.

3.2.1 WAYPOINTS COORDINATES

In this paragraph is described how the waypoints coordinates have been extracted and used to allow the extraction of all the data used by the algorithm.

As said before, the points' coordinates, that make a route, are stored in the computer as a GPX format file.

"A GPX file, also known as a GPS Exchange Format file, is an XML schema designed as a common GPS data format for software applications" that contains geographic information such as waypoints, tracks, and routes saved in it. [29]

According to the documentation, a GPX file has three data types:

- wptType, that is the single waypoint among a collection of point with no sequential relationship. It is identified with the WGS 84 (GPS) coordinates of a point.
- rteType, is a route, so an ordered list of waypoints.
- trkType, is a track, made at least by a segment containing waypoints, listed in an ordered way to describe a path.

The minimum properties of a GPX file are latitude and longitude for every single point. The latitude and longitude values are stored according to the WGS datum format and are expressed in decimal degree. [30]

Returning to the working mode description, the file can be downloaded from internet, using a web service called OpenSourceRoutingMachine (OSRM). [31]

In Figure 20, it is possible to observe an example of data contained in the downloaded GPX file, using this service.

From the web, it's possible reach the site and after having selected two points (start point in green, end point in red) a path will be generated automatically, finding, by default setting, the shortest path in the road network, Figure 21.

Note that the user has to select the car profile, otherwise by default, the system will find a path according to bike road network; this option can be selected by clicking on the button signed with label A, in Figure 21.

Optionally, it is possible to rearrange the path according to the user preference, creating the desired route by hand, clicking on the street desired.

a lot of this problem case had arisen, and these had been translated in trouble for the extraction of road information, since the algorithm did not return the correct results. After having evaluated different ideas, it was decided to use an approach, that will be described in sub-paragraph 3.3.3.1. Another reason why the OSRM service was chosen for the extraction of the route waypoints was that map is created using the data stored in the OSM's databases. [32]

3.2.2 OPEN STREET MAP

The state of charge consumption of a battery electric vehicle is influenced by several elements, among which the road information, such the road typology, the speed limit, the traffic lights presence and so on.

This is the reason why the algorithm needs a lot of road information.

In order to extract these data, the database provided by OpenStreetMap (OSM) was chosen. [32]

OpenStreetMap is an open-source service used to extract a lot of information from the route map. The fundamental characteristic of the geographic data present in OSM is that they are distributed with a free license, the Open Database License: so, it is possible to use them freely for any purpose, including commercial, with the only constraint of citing the source and using the same license for any works derived from OSM data.

Everyone can contribute by enriching or correcting the data. [31]

The OSM databases store route information, using three main types of data:

- Nodes.
- Ways.
- Relations.

The node is both one of the most important and the key element in OpenStreetMap data model. This is characterized by its latitude, longitude, and node id. The node can be used both as descriptor of a certain tag or as one of the points of one or more ways, making in this last case the so called 'path' of the way. In this latter case, the node can assume a certain tag. [33]

In Figure 22, it is possible to see an example of response relatives to a node, containing the node id, the coordinates, and the tag. The node symbol is .

```

<node id="25496583" lat="51.5173639" lon="-0.140043" version="1"
changeset="203496" user="80n" uid="1238" visible="true" timestamp="2007-01-
28T11:40:26Z">
  <tag k="highway" v="traffic_signals"/>
</node>

```

Figure 22. Example of node information in XML format. Note the presence of node id, latitude, longitude, and tag. [33]

The way instead represents a linear feature of the ground, e.g., a road, a sidewalk, a river. It is composed by an ordered list of nodes and has at least one tag associated. A way can be done make by a number of nodes that is in the range of 2-2.000. A way can be open or closed, the latter case happens when the last-point and the first-point coincide. [34]

In Figure 23, it is represented an example of response containing way information's, like the way id, the list of nodes making it and the tag associated.

```

<way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z" version="8"
changeset="816806" user="Blumpsy" uid="64226">
  <nd ref="822403"/>
  <nd ref="21533912"/>
  <nd ref="821601"/>
  <nd ref="21533910"/>
  <nd ref="135791608"/>
  <nd ref="333725784"/>
  <nd ref="333725781"/>
  <nd ref="333725774"/>
  <nd ref="333725776"/>
  <nd ref="823771"/>
  <tag k="highway" v="residential"/>
  <tag k="name" v="Clipstone Street"/>
  <tag k="oneway" v="yes"/>
</way>

```

Figure 23. Example of way information in XML format. Note the presence of way id, list of node ids and tags. [34]

To obtain this type of information about the road, the Overpass API (formerly known as OSM Server Side Scripting, or OSM3S before 2011) was used. [35]

Overpass is an Application Programming Interface used to read only customized part of data from the entire OSM database and work over the web. The user sends the request

or query to API and it gets back the data required, in terms of nodes, ways and relations. [35]

To do the request, the programming languages used are Overpass QL (Overpass Query Language) or Overpass XML, but in this work was adopted the first one.

To get familiar with this language and to play with it, it is possible to use an interactive Web-based frontend, called Overpass-turbo, whose link is: <https://overpass-turbo.eu/>. The complete documentation about Overpass API is reachable to the following link: <https://dev.overpass-api.de/overpass-doc/en/>.

Examples of requests are showed below:

```
[out:xml];
node(node_id);
(._>);
out;
```

Figure 24. Example of query with node.

```
(out:xml);
way[highway](way_id);
(._>);
out;
```

Figure 25. Example of query with way.

```
(out:xml);
way[highway]{{bbox}};
(._>);
out;
```

Figure 26. Example of query with way.

Since the output data returned by the server request are in json or xml format, it was necessary to write a function that performed the data parsing in order to store those in Python structures. In this way, the future manipulation of the data was simplified.

The two dictionary structures created to store the OSM information, are showed in Figure 27 and Figure 28.

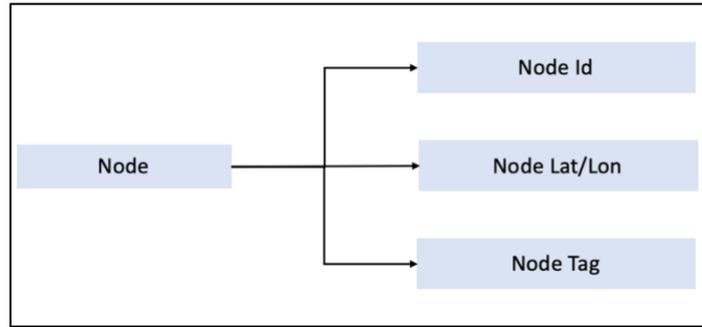


Figure 27. Example of data parsing for extraction of node information on a Python dictionary, named *node*.

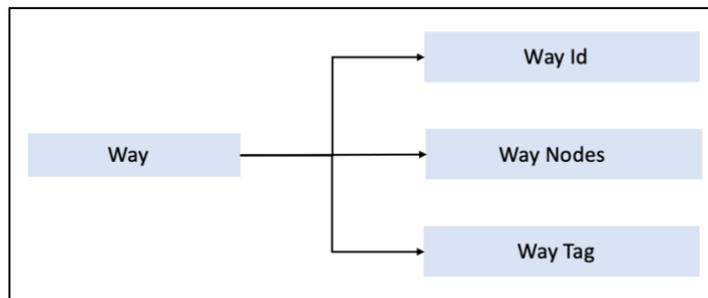


Figure 28. Example of data parsing for extraction of way information on a Python dictionary, named *way*.

The first dictionary, called *node*, contains three list structures: the 'Node Id', the 'Node Lat/Lon' and the 'Node Tag', used to save respectively, the id, the coordinates, and the tags of each node.

Also, the second dictionary, called *way*, contains three list structures: the 'Way Id', the 'Way Node' and the 'Way Tag', used to store respectively, the id, the list of nodes making the way, and the tags of each way.

These two dictionaries were stored inside another dictionary structure, called *parsed_osm*.

3.2.3 ROAD INFORMATION EXTRACTION

One of most hard challenges of this work was to understand how to extract the road information and feed them to the algorithm. Basically, these are needed for obtaining a prediction of the velocity profile, as will be seen in the next sub-paragraphs.

As explained in the previous sub-paragraph, it was decided to use the OpenStreetMap data [32], but the problem was how to select only those ways that belong to the imported route.

Two different extraction techniques are evaluated in this work, and both are described in the next two sub-paragraphs, analyzing the pros and counters for each of them. Briefly, it is explained here the selected and used procedure for the extraction, called Technique 1, Figure 29.

Firstly, after collecting all the road data, the nodes that were closer to the waypoints of the route were selected; secondly from these nodes it was possible obtain the ways of the route; to conclude the extraction of the road information was performed.

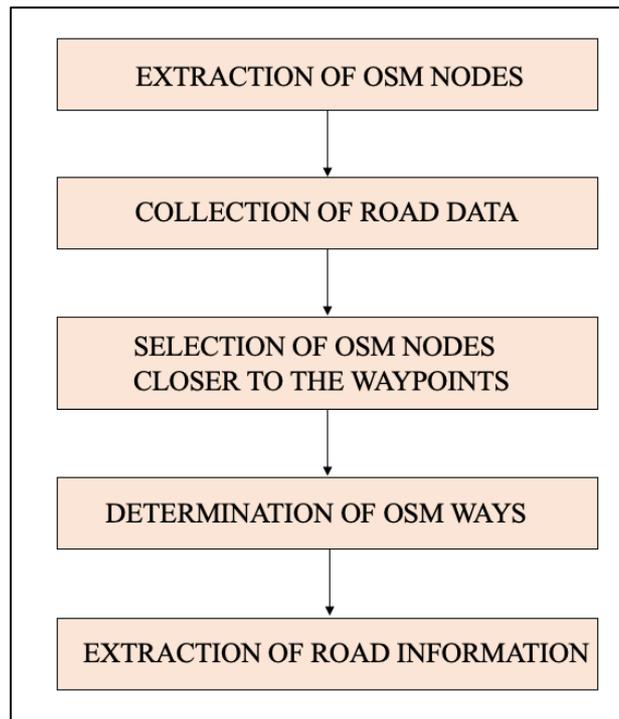


Figure 29. Operation flow followed to extract the road data from OSM, Technique 1.

3.2.3.1 EXTRACTION TECHNIQUE 1

The first approach thought was based on the following command request:

```
[out:xml];
way[highway]{{box_range}};
(._>);
out;
```

Figure 30. Query to download road information, written Overpass QL language.

The request in Figure 30, shows an example of query to Overpass to return the list of ways, with highway tag, present inside the area whose bounds coordinates are written the section called, box_range.

At this point a server request is send, and the output data are returned in Extensible Markup Language (XML) format, containing the dataset made by all the nodes and ways presented in the selected area.

In turn each way contains an ordered list of nodes that make it and a series of tags or information too.

An example of complete request made in the algorithm is the following:

```
[out:xml];
way[highway][["highway"!~"footway"]["highway"!~"track"]["highway"!~"path"]["highw
ay"!~"pedestrian"]["highway"!~"raceway"]["highway"!~"bridleway"]
["highway"!~"steps"]["highway"!~"corridor"]{{box_range}};
(._;>);
out;
```

Figure 31. Example of complete request to download road data, in Overpass QL language, using technique 1.

Observing the previous request example, Figure 31, other restrictions are added to skim all the unnecessary output data, so nodes, ways and tags returned by the server, avoiding both weighting a lot on the algorithm memory and in term of download useless data too. In Figure 31, the user requires all the ways located in the area, whose bounds are defined by the box_range parameter, according to the filters imposed; the output data will be provided in xml format.

Once obtained the only nodes and relative ways considered useful to obtain the desired road information, an ad-hoc function was designed and written to extract the ways belonging to the imported route.

This algorithm, explained in sub-paragraph 3.2.3.3, will allow to trace the ways making the route, starting by the knowledge of the selected OSM's nodes.

But first is necessary to select the only nodes that belong to the route imported, and how this is done, is explained below. In Figure 32, it is possible to see an example of the OSM server response, in terms of nodes returned, relatives to the selected area, according to the request exposed in Figure 8.

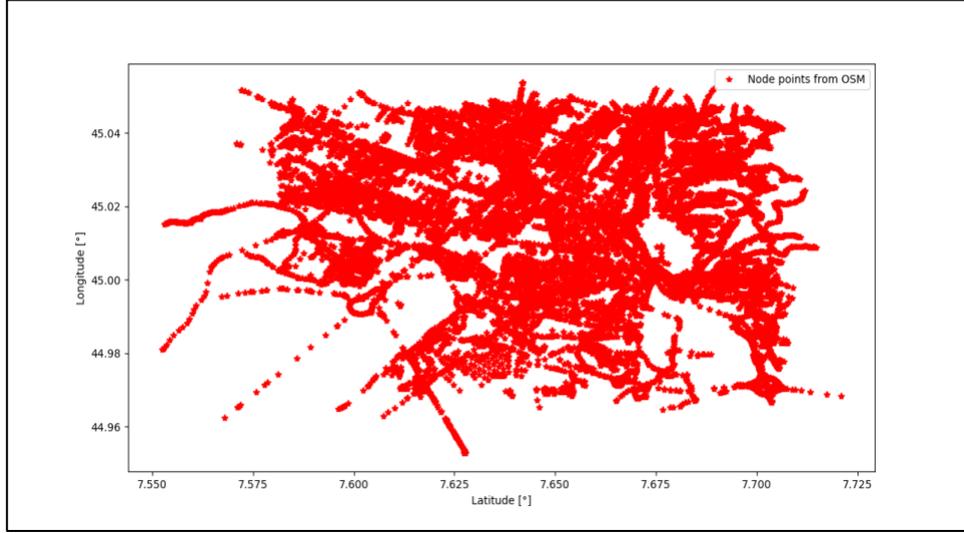


Figure 32. Example of OSM nodes presented in the selected area.

In order to select among the high number of nodes given by OSM, only those corresponding to the waypoints of the selected route, it was thought to select only those nodes whose distance with respect the waypoints was the smallest.

To do this, it was needed to compute the distance between each node and each waypoint, using the classical formula to compute the distance between two points in a plane.

To better understand the following formulations, for this paragraph it was used the following nomenclature, the longitude it was indicated with the x , the latitude with the y . So, identifying the coordinates x and y of the OSM's node i -th as X_{Ni} , Y_{Ni} respectively and the coordinates x and y of the waypoint j -th as X_{Wj} , Y_{Wj} , it was possible to obtain the distance applying the Pitagora's theorem, according to the following equations:

$$d_{ij} = \sqrt{dist_{x_{ij}}^2 + dist_{y_{ij}}^2} \quad (3.1)$$

where:

$$dist_{x_{ij}} = X_{Ni} - X_{Wj} \quad (3.2)$$

$$dist_{y_{ij}} = Y_{Ni} - Y_{Wj} \quad (3.3)$$

Since the considered system is made by n -nodes and m -waypoints, combining all the distances, it will be obtained a matrix, called D matrix, whose dimension is $\mathbb{R}^{n \times m}$.

Each element of this matrix D , is indicated with d_{ij} and it indicates the distance between the node i and the waypoint j .

$$D = \begin{bmatrix} d_{11} & \cdots & d_{1j} \\ \vdots & \ddots & \vdots \\ d_{i1} & \cdots & d_{ij} \end{bmatrix} \quad \begin{array}{l} i \text{ is the general OSM node,} \\ j \text{ is the general waypoint} \end{array}$$

The matrix D has the property to be symmetric, so $d_{ij} = d_{ji}$, for $i \neq j$.

To select only the OSM's nodes closer to the waypoints of the route, it was applied a function that returns a vector, called *min_dist* with dimension $\mathbb{R}^{1 \times m}$, containing the minimum distance value along the axis 0 (in other words, it was selected the minimum d value along the rows for each column of the matrix). In addition, another array, indicated with *index_min*, was computed, whose dimension is $\mathbb{R}^{1 \times m}$ too, that is made by indices. Each index points out the position where the OSM's node have place inside the original array containing all the OSM's nodes.

An example about this last point is done to explain better what the indices are; supposed to have only 3 nodes from OSM, and 5 waypoints, Figure 33.

So, $node = [n_1, n_2, n_3]$ and $waypoint = [w_1, w_2, w_3, w_4, w_5]$.

The matrix D has a dimension $\mathbb{R}^{3 \times 5}$, instead the arrays *min_dist* and *index_min* belongs to $\mathbb{R}^{1 \times 5}$.

The first element of *index_min* array identifies the position of the node n_i that is closer to the first waypoint w_1 ; position referred to the index position where the selected node is in the *node* array. The same for the other waypoints.

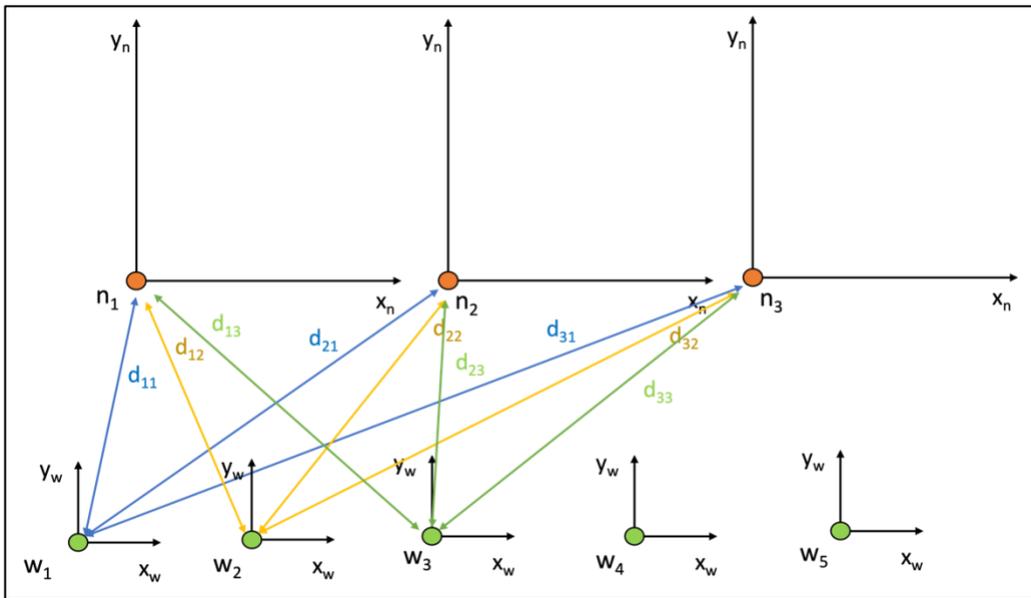


Figure 33. Example of application of technique 1, with 5 OSM's nodes and 3 waypoints.

At this point, knowing the indices of the nodes closer to the waypoints, it was possible to extract only the needed OSM's node and discard others, and to use these ones to understand what ways belong to the imported route. The array containing all the selected

OSM's nodes, was called *node_optimized*, and as said has the same dimension of the waypoints vector.

This method implemented is very fast, since it does only one request to the server and all the other operations, like the selection and extraction of the nodes are made offline. [28] The bad point is that it works fine only with the service supplied by OSRM, in the GPX format. [31]

If the file containing the coordinates of the route is exported from a GPS or Google Maps, the extraction result is not good as expected but there will be a map-matching problem; practically it will happen that the coordinate imported will not correspond to the correct OSM's node.

3.2.3.2 EXTRACTION TECHNIQUE 2

The second technique used to extract road information is designed on the following Overpass query, in Figure 34.

```
[out:xml];
way [highway] (around:1,55.8141,26.8400);
(._>);
out;
```

Figure 34. Example of complete request to download road data, in Overpass QL language, using technique 2.

In the previous query example, are requested the ways with tag *highway*, that are inside the circle centered in the coordinates 55.8141 and 26.8400, with a radius of 1 meter. More in general, the function, using this query, checks if some OSM ways are inside the circle area, centered in the coordinates and with radius r . Since the ways, added by the users, can be placed not at fixed distance with respect the coordinates, it was decided to create a for loop, inside which the dimension of the radius is increased of a constant term at each iteration.

Inside the loop, a condition has to be checked: if the ways are found inside the circle with radius r , then the function ends and the new coordinate is given as input and perfume the same operations; instead, if the conditions is not true, the radius value increases until a way is found.

This is the positive, since it is possible to give as input the waypoints coordinates, everywhere they are taken, and find the corresponding way from OSM database.

The drawback of the function is the time spent for the research on the server, in fact it makes a request for every pair of coordinates and in addition, it is necessary to consider the number of iterations done until a way is found.

For these reasons, the technique described here is not used in the algorithm, but it is preferred the first one.

3.2.3.3 ROAD STREET IDENTIFICATION

In this sub-paragraph, it is described how the information associated to the ways of the selected route are extracted. The method has a reference with the first extraction method, previously defined as technique 1.

Firstly, it was necessary to understand how to select what ways there are in the selected route, using as input the *node_optimized* vector, containing the list of OSM's node closer to the waypoints.

From OSM's documentation, it was possible to read that a way is composed by an ordered list of nodes and so, using this information, a function was developed to return the list of ways that are inside the route.

The conceived idea consists of evaluating pairs of two elements of the *node_optimized array* at times, and for each way, check two conditions: the first one is to verify if the two items belong to the nodes' list of that way, while the second one if they are consecutive in that list.

If these two conditions are true, then the road information are extracted into appropriate Python lists; otherwise, if the two nodes selected are not consecutive in any way, then it is chosen the previous selected way. A little part of the algorithm for the road data extraction is described below here, in Figure 35.

```
1. for i in range(len(index_min)-1):
2.
3.     way_found = 0
4.     id_node1 = np.array(parsed_osm['node']['id'][0,index_min[i]])
5.     id_node2 = np.array(parsed_osm['node']['id'][0,index_min[i+1]])
6.
7.     tot_way = len(parsed_osm['way']['id']) # total number of ways
8.
9.     for j in range(tot_way):
10.
11.         node_of_way = np.array(parsed_osm['way']['node'][j],
dtype=int) # take the list of
nodes correspondent to the selected way
12.
13.         if any(np.equal(id_node1, node_of_way)) and
any(np.equal(id_node2, node_of_way)): #
check if the two selected nodes belongs to the selected way
14.
15.             cond1 = id_node1 == node_of_way
16.             cond2 = id_node2 == node_of_way
```

```

17.
18.     ind_cond1 = []
19.     ind_cond2 = []
20.
21. for l in range(len(cond1)):
22.     if cond1[l] == True:
23.         ind_cond1.append(l)
24.
25.     for n in range(len(cond2)):
26.         if cond2[n] == True:
27.             ind_cond2.append(n)
28.
29.     if np.absolute(ind_cond2[0] - ind_cond1[0]) == 1:
30.         way_found += 1
31.         tot_way_found += 1
32.
33.         id_way = parsed_osm['way']['id'][j]
34.         num_tag_way = len(parsed_osm['way']['tag'][way_index])
35.
36.         ''' ROAD INFORMATION EXTRACTION '''
37.         .....
38.         ..... Code not showed for the extraction of the information
39.         ..... Code not showed for the extraction of the information
40.         .....
41.         ''' END EXTRACTION '''
42.
43.     if way_found == 0 and i != 0:
44.         take the information associated to the previous way found

```

Figure 35. Algorithm for road data extraction.

The information extracted from the OSM's databases, relative to the ways, are the id, the name, the asphalt type, the way classification, the speed limit imposed by signs, if present, and the speed limits associated to the road typology. All these data were saved first into Python lists structure, then were stored into a Python dictionary to be managed easily.

3.2.4 VELOCITY PROFILE PREDICTION

The algorithm to work needs to know a very important information, that is the velocity profile and since the program is launched before the trip begins, it is necessary to know it a priori. Having a prediction of the velocity profile is not an easy task to achieve, since it depends by a lot of factors, such as the driving style of the driver, the weather conditions, the speed limit imposed, the traffic conditions, the surface and road typology.

Due to the offline working mode of the algorithm, it was decided to obtain a velocity profile based on three information, the speed limits imposed by the road typology, the speed limits established by the vertical and horizontal signs, and by the weather conditions. In this sub-paragraph is described step-by-step how the velocity profile was obtained. Firstly, it was considered a profile based on the speed limit regulated by the Italian law, according to the road typology and to the sign's presence. To obtain the needed information it was used OSM. [32]

The function, designed to obtain the legal speed limit profile, is based on the following idea: given an array made by all the ways present in the selected route. For each way, the selection criterion operates according to the ensuing conditions: first, it checks if there is some speed limit sign (both vertical and horizontal), and, if this condition is true, the function imposes that value in the selected stretch of road. Instead, if the condition is false, the function sets as velocity value the one associated to the road classification, according to Table 2.

Road classification	Good weather conditions	Bad weather conditions
Motorway	130 Km/h	110 Km/h
Trunk	110 Km/h	90 Km/h
Primary	90 Km/h	90 Km/h
Secondary	90 Km/h	90 Km/h
Tertiary	90 Km/h	90 Km/h
Residential	50 Km/h	50 Km/h
Living Street	50 Km/h	50 Km/h
Track	50 Km/h	50 Km/h
Unclassified	50 Km/h	50 Km/h
Service	20 Km/h	20 Km/h

Table 2. Speed limit according to road classification. [36]

The speed limit, associated to the road classification coincides exactly with that imposed by Italian Law on road safety; although in some roads the speed limit can be changed if it is expressly stated in signs. [36]

The final result of this approach is showed in Figure 36, where is represented the legal speed limit profile. Nevertheless, the result obtained is not so good: first, it does not consider the vehicle acceleration and, second, it can be observed the presence of high steps, in correspondence of the speed change, that describes an unrealistic behavior. The presence of these steps produces an acceleration not physically achievable; remember that the acceleration is the first derivate of the velocity in time, and that the derivate of a step signal is the impulse function but the impulse function is not a reproducible signal.

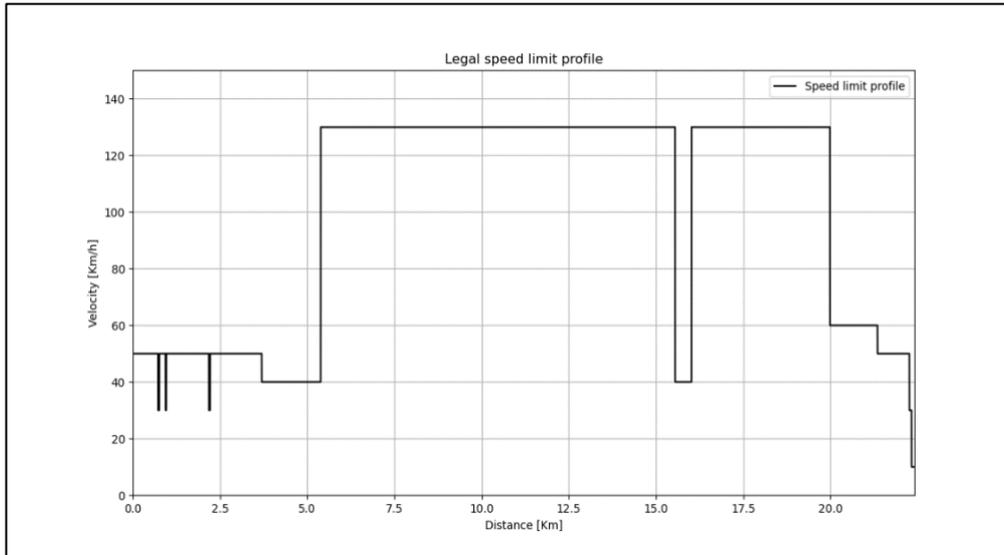


Figure 36. Legal speed limit profile obtained by the algorithm, according to both road signs and road classification.

To avoid having problems like these and to have a more realistic result, it was considered an acceleration-velocity model that the vehicle can follow.

To take into account it, the acceleration profile was obtained from real velocity measurements and then by plotting a velocity-acceleration profile, it was extracted the maximum longitudinal acceleration and deceleration trend, Figure 20. [28]

Obviously, the acceleration was obtained from the speed real data, using the backward discrete difference algorithm, as in Figure 37, or the Python function *diff*, whose belong to the NumPy module. [37]

$$x(i + 1) = \frac{x(i) - x(i-1)}{\Delta t}$$

with $\Delta t = 1$ since the sampling time is 1 second.

Figure 37. Implementation of the backward algorithm.

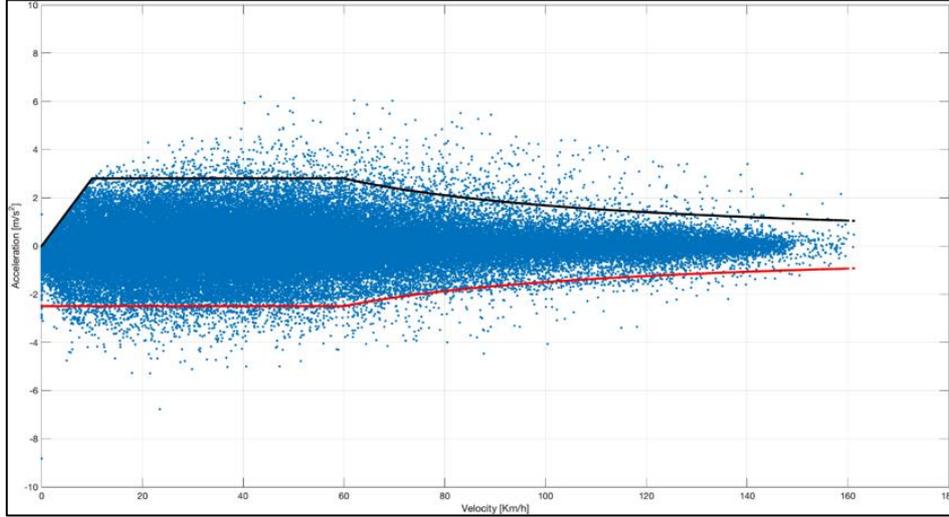


Figure 38. Velocity-acceleration profile according to real collected data. In red the deceleration trend, in black the acceleration one as function of the velocity.

At this point it was possible to get the velocity profile, with also a regard to the maximum acceleration and deceleration values, as function of the velocity, by the numerical integration between each of the points of the distance array, according to equations 3.4, 3.5. [27], [28]

$$v(s_{i+1}) = \min\left(v(s_i) + \int_{t(s_i)}^{t(s_{i+1})} a_{max,acc}(v) dt, v_{max}(s_{i+1})\right) \left[\frac{m}{s}\right] \quad (3.4)$$

$$v(s_{i+1}) = \max\left(v(s_i) + \int_{t(s_i)}^{t(s_{i+1})} a_{max,dec}(v) dt, v_{max}(s_{i+1})\right) \left[\frac{m}{s}\right] \quad (3.5)$$

where:

- $a_{max,acc}$, is the maximum longitudinal acceleration.
- $a_{max,dec}$, is the maximum longitudinal deceleration.

The values of $a_{max,acc}$ and $a_{max,dec}$, change with velocity and are chosen according to plot displayed in Figure 38. The equations above are computed for $i = 1, \dots, N - 1$, where N is the number of waypoints points the route.

The first equation, 3.4, is referred to the acceleration case, the second one, 3.5, to the deceleration case. Combining the two results, the complete legal speed limit profile is obtained. The initial and the final velocity are imposed equal to zero, assuming the vehicle starts and ends the trip standstill.

In Figure 39, it is possible to observe the effect of the acceleration term, visible as red line, with a more realistic result in the transition from 0 to 50 Km/h, for example. The opposite situation is showed in Figure 40, where is displayed the deceleration case, in blue line.

The combination of the two situations, so acceleration and deceleration case, is showed in Figure 41, where the green line represents the velocity profile, with respect to the black line exemplify the original limit speed profile.

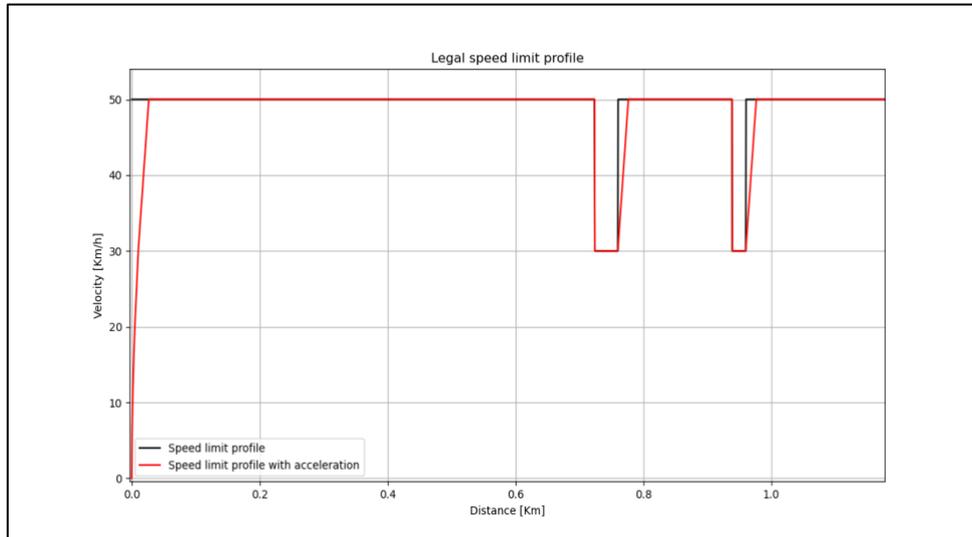


Figure 39. Legal speed limit profile. In red the velocity profile considering the acceleration, in black the basic case without acceleration.

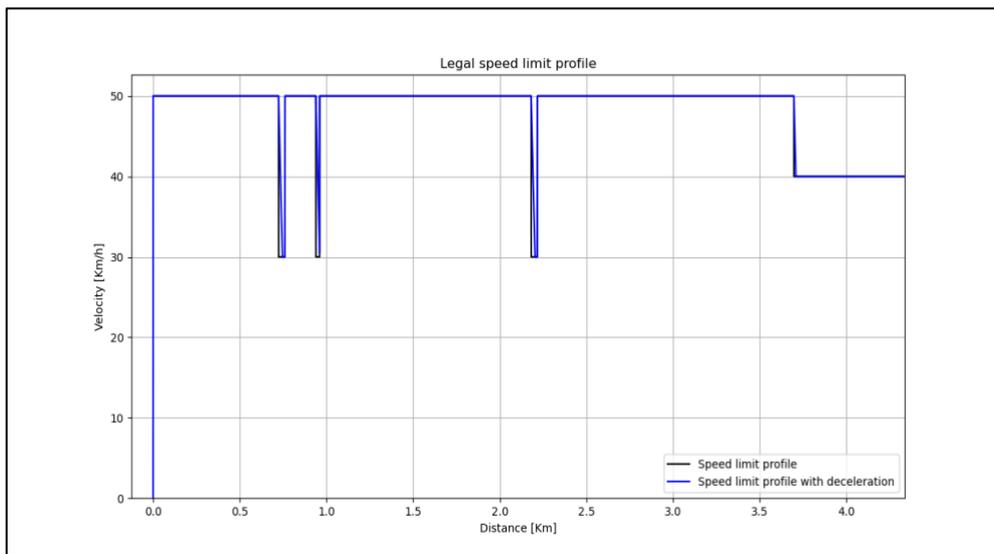


Figure 40. Legal speed limit profile. In red the velocity profile considering the deceleration, in black the basic case without deceleration.

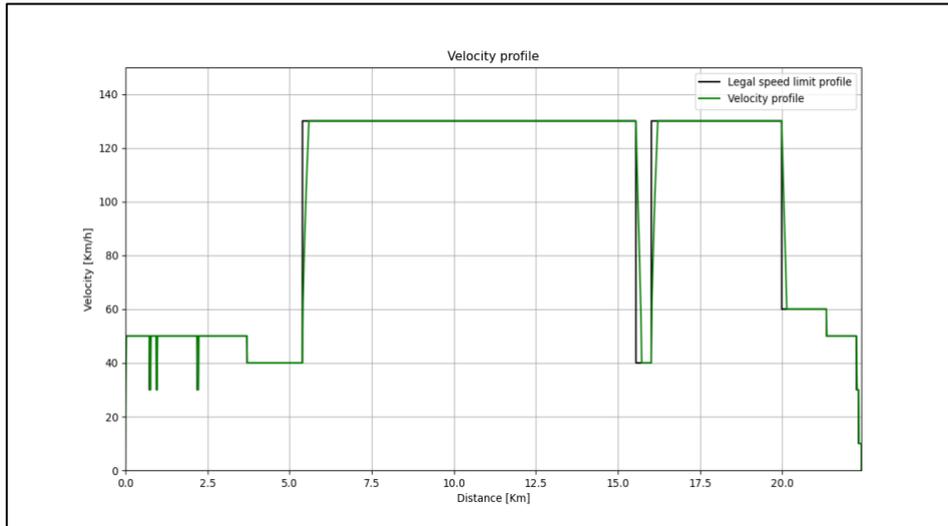


Figure 41. Example of velocity profile obtained.

It is also possible to determine a personal velocity profile, by changing a factor, called *bias velocity factor* or *BVF*, that changes the amplitude of the velocity profile obtained by the algorithm, according to the equation 3.6.

Changing this factor, also the final result will change, so the energy consumption and the state of charge associated.

$$v = v \cdot BVF \quad \left[\frac{m}{s} \right] \quad (3.6)$$

Note that:

- if $0 < BVF < 1$, the velocity is reduced,
- else if $BVF = 1$, the velocity remains at legal limit speed,
- else $BVF > 1$, the velocity is gained.

By default setting the BVF is imposed equal to 0.90, to be far from the speed limit values. Moreover, to have a smooth profile, it was applied a digital Butterworth low-pass filter, to the velocity profile previously obtained.

In Figure 42, it is showed the speed profile returned by the function, in three cases: in black line the velocity profile without filter and BVF applied; in red, the velocity filtered and with the BVF applied; in blue, the case equal to the red one but without filter usage. The Figure 43, Figure 44, Figure 45, instead show a zoom of the profile, where it is possible to appreciate the smooth effect created by the filter.

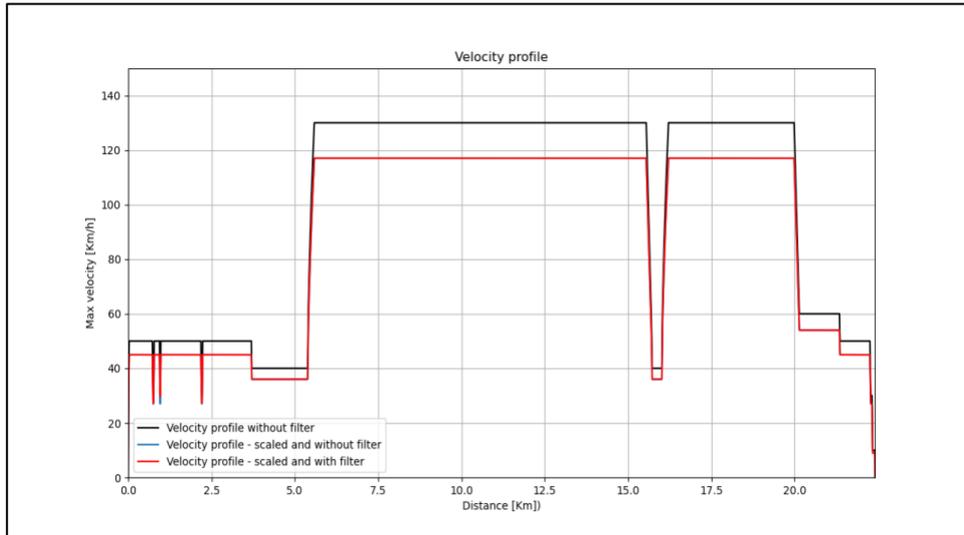


Figure 42. Example of velocity profile obtained by the algorithm.

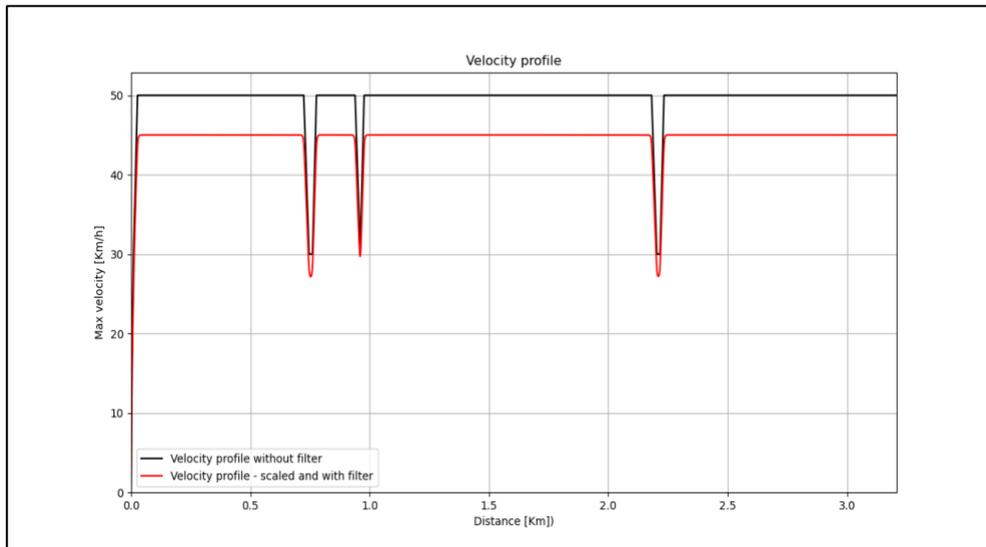


Figure 43. Example of velocity profile obtained by the algorithm, with a zoom on the first 3 Km.

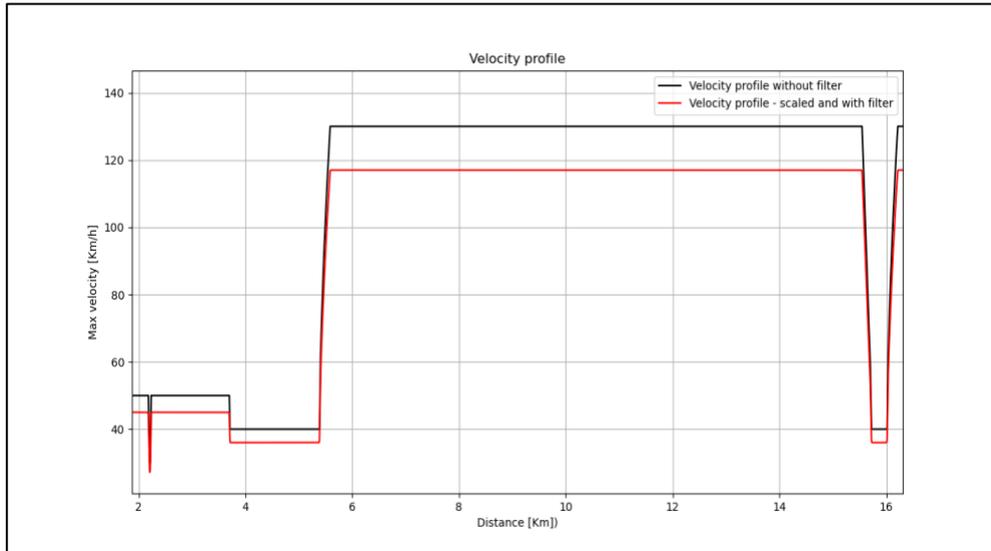


Figure 44. Example of velocity profile obtained by the algorithm, with a zoom on range 2-16 Km.

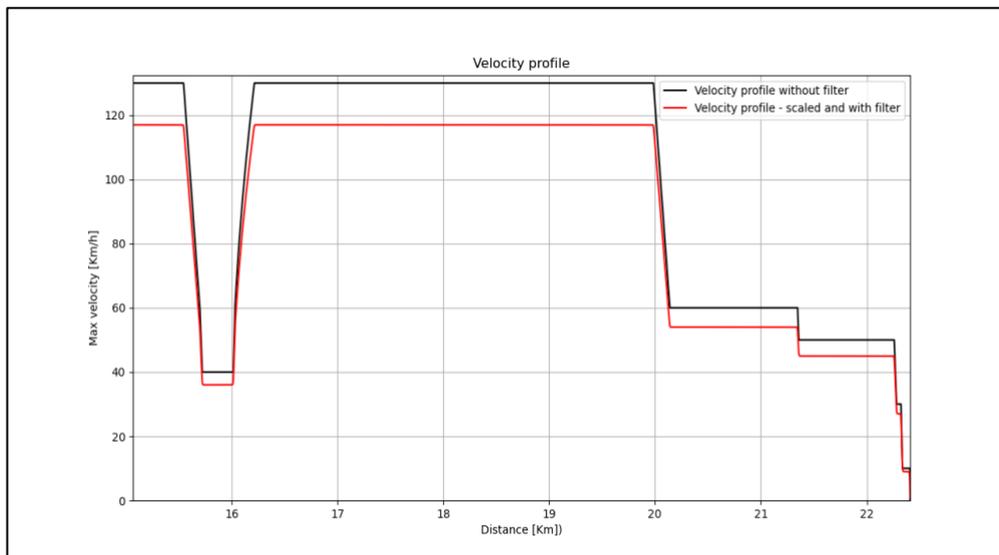


Figure 45. Example of velocity profile obtained by the algorithm, with a zoom on the last 7 Km.

To conclude, starting from the information about the road, it was possible to obtain a velocity profile entirely based on the speed limit imposed by the Italian law in the field of transport and by Polizia di Stato. [36]

The flow chart followed to create the velocity profile is presented in Figure 46. Note that the prediction of the speed profile will be used both in the real application and in the testing part; instead, in the next chapters, in order to verify if the implemented formulas work well, it is used a velocity profile extracted from real measurements of the vehicle.

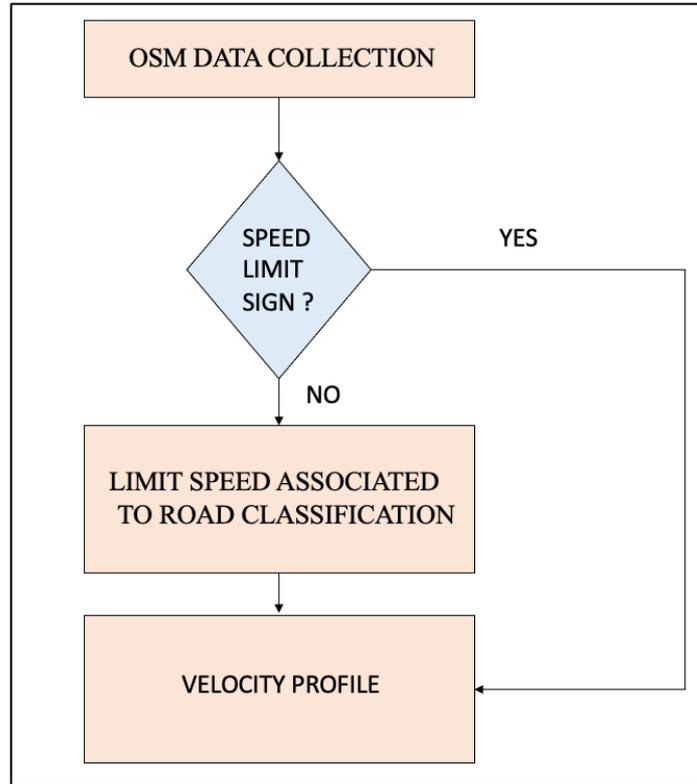


Figure 46. Flow chart for the legal speed limit profile generation.

3.2.5 DISTANCE PROFILE

Another important function implemented in this algorithm was the one able to compute the length of the route and the distance between two consecutive waypoints.

Although the computation of the distance between two points, from A to B, may be considered as an easy task, it is not. The reason why the algorithm needs this data, among the others, is to compute the slope of the route, since it needs elevation and length data. The idea was to divide the route in several segments, called ways, where each segment is made by two points, or nodes, described by latitude and longitude coordinates, Figure 47.

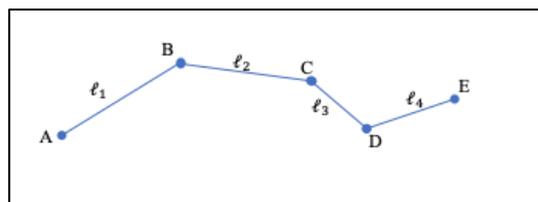


Figure 47. Example of route made by 5 nodes and 4 ways. The total length is $l_{TOT} = l_1 + l_2 + l_3 + l_4$.

To compute the total length of the route, the algorithm first computes the length of each segment and then, summing these values, produces the total route length, in meters, as in equation 3.7.

$$\ell_{TOT} = \sum_{i=0}^n \ell_i \quad [\text{m}] \quad (3.7)$$

where:

- ℓ_i [m], is the length of the single segment.
- ℓ_{TOT} [m], is the total length of the route.

The example in Figure 47, shows a route that starts in A and ends in E, that is made by 5 nodes (A, B, C, D, E) and by 4 ways (first way A-B, second B-C, third C-D, fourth D-E). To compute the single segment length, the implemented function needs the coordinates, in latitude and longitude, of the two points of which the distance has to be calculated and then, according with the haversine's formula, the distance among the two points is returned.

"The Haversine formula is used to compute the great-circle distance between two points", or, in other words, it is the shortest distance over the earth surface. The distance is computed in fly, ignoring so any hills or mountain. [38]

The equations implemented are [38]:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (3.8)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (3.9)$$

$$d = R \cdot c \quad [\text{m}] \quad (3.10)$$

where:

- $R = 6,337 \cdot 10^6$ [m], is the earth's radius (mean value).
- φ , is the latitude, in radians.
- λ , is the longitude, in radians.
- d , is the distance between two waypoints, in meters.

Note that the coordinates must be converted from degree to radians.

This formulation assumes that the earth is spherical, ignoring the ellipsoidal effects, but it is accurate enough for the thesis purposes, in force of the fact the distance between the

couples of points is very small. Using the spherical model, the error made is up to 0.3%. [38]

3.2.6 ELEVATION PROFILE

Another input of the algorithm is the elevation profile, which is a diagram made by all the altitude values of each GPS point, measured taking the water level as reference point. The altitude values can be obtained either using the GPS system, installed in the car itself or, otherwise, using alternative approaches, like it was done in this circumstance, where it was used elevation data provided by the Shuttle Radar Topography Mission (SRTM) database. [28], [39]

The Shuttle Radar Topography Mission is an international research effort, spearheaded by the U.S National Geospatial-Intelligence Agency (NGA) and U.S National Aeronautics and Space Administration (NASA), whose goal was to obtain digital elevation models on a near-global scale from 56 degrees south to 60 degrees north latitude, which comprises almost 80% of the earth total landmass. [39]

The data has been released with a horizontal resolution of 1 arc-second, that correspond to 30 m. The project was able to generate the most complete high-resolution digital topographic database of Earth prior to the release of the ASTER GDEM in 2009. [39]

Other ways to obtain this data are to use an API, like Elevation API, provided by Google, but unfortunately this service is free only for a limited number of requests and so it has been necessary to use other techniques. However, several other APIs are available on internet, also with a free usage, but they have not been considered.

In this algorithm a Python module, called `srtm.py`, written and released by Tomo Krajina, was used. The module reads the SRTM files and extracts the elevation data relative to the position recorded in the GPX file. [40]

In Figure 25 it is possible to see the altitude profile relatives to a generic travel of about 25 kilometers, obtained using the `srtm.py` module.

Analyzing the Figure 48 can be seen that the trip begins with an altitude value of 272 meter above the level of water and then goes down until reach the 226 meters.

Nevertheless, even if the results are quite good, the elevation profile is not as is best as it could be expected in a real situation, due to the presence of a step behaviors or of spikes presence. Figure 49 shows a zoom of the elevation profile, relative to the firsts 2.5 Km of the same trip used in Figure 48, where can be shown these problems.

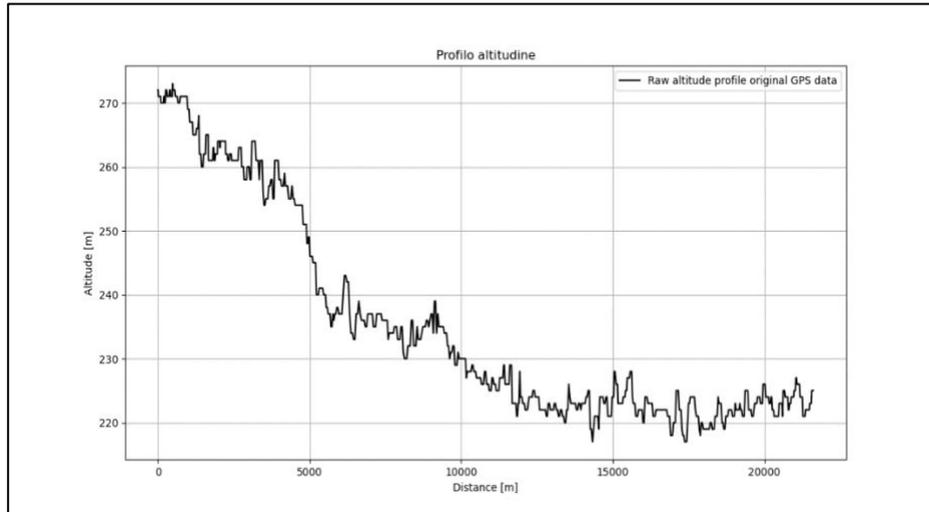


Figure 48. Elevation profile obtained from the algorithm.

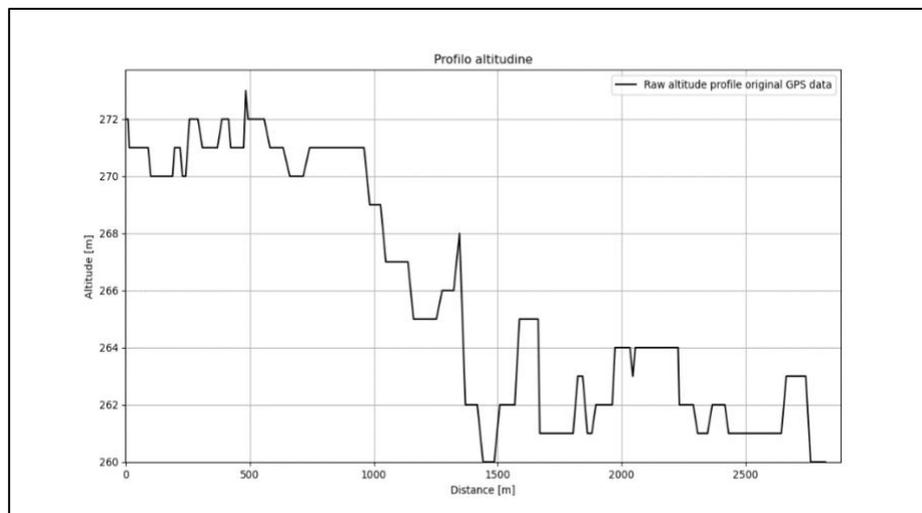


Figure 49. Elevation profile, focused on the firsts 2.5 Km.

For having more realistic results, it was chosen to use a digital low-pass Butterworth filter and apply it to the elevation profile obtained, in order to smooth the altitude profile. [28] “The Butterworth filter is a typical signal processing filter designed to have a frequency response as flat as possible in the pass band”. [41]

The filter was implemented in the Python module called SciPy, that is an open-source software for mathematics, engineering and science. [42], [43]

The Python Butterworth filter implementation is showed in Figure 50.

```
b is the numerator of the filter & a is the denominator
b, a = signal.butter(N, Wn, 'low', false, fs)
```

Figure 50. Implementation of the Butterworth digital filter in Python. [42]

After several trial-and-error procedure, it has been selected the order of the filter and, analyzing the frequency response of the data, it has been possible to establish all the necessary parameters to implement the filter, as listed in Table 3.

The result is showed in Figure 51 and Figure 52, where it is possible to observe a comparison between the original raw elevation profile and the filtered one. Note that, the coefficient W_n is the normalized coefficient, computed as $W_n = \frac{f_{cut}}{\frac{f_s}{2}}$.

Parameter	Value
Filter typology	Low-Pass
Filter order, N	3
Sampling frequency, f_s	0.2
Cut-off frequency, f_{cut}	0.004
W_n	0.001

Table 3. Butterworth filter parameters.

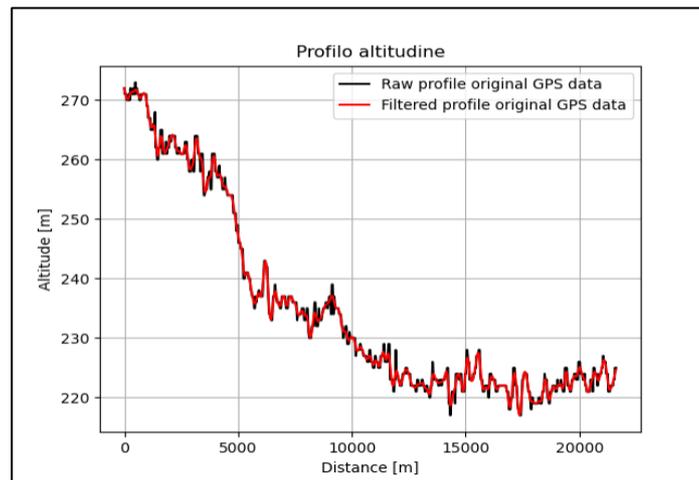


Figure 51. Elevation profile obtained from the algorithm. In red the filtered profile, in black the original one.

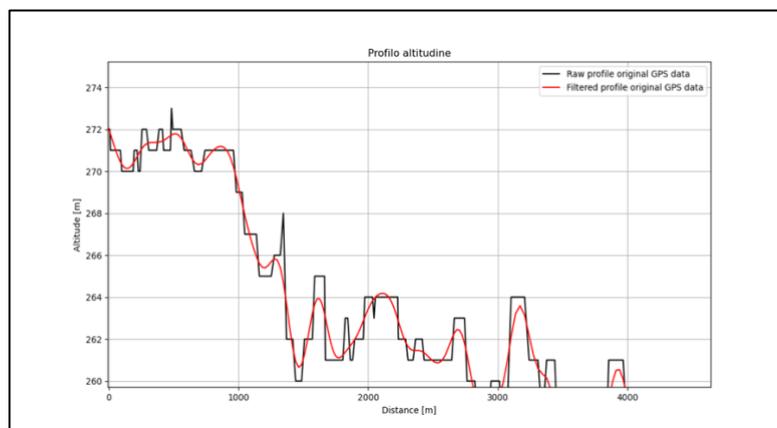


Figure 52. Elevation profile, focused on the firsts 2.5 Km. In red the filtered profile, in black the original one.

3.2.7 WHEATER DATA

To accomplish the algorithm goal, some weather data are necessary, such as the ambient temperature T , the relative humidity level RH , the atmospheric pressure P , the wind information like direction β and absolute speed v_w and to conclude general weather indications.

To collect all these basic information it was decided to use a service provided by OpenWeatherMap. [44]

OpenWeatherMap is an online service owned by OpenWeather Ltd, that provides global weather data via API. The powerful of this service is that it provides both past and present weather information as well as weather forecasts relative to exact locations or regions of the earth, simply giving as input the points coordinates, in latitude and longitude.

To use these services an API key is needed, since in this way it is possible to count the number of requests done by the user. To have wide experience with the services offered, one of the several payment plans must be activated, but obviously this is not free.

For this work a developer plan was activated, whose characteristics were: 3000 calls per minute if current data API is used, and 50000 calls per day for the historical API. [44]

In this thesis were used both past and actual weather data, to verify if the implemented algorithm works well and so, for testing purposes and for normal usage, respectively.

To extract past data, the Historical Weather API was used, while, for the current data, the Current Weather Data API. The API returns the data in json format, so in order to parse and extract the needed data, an ad-hoc function was written.

Example of request for the extraction of current weather data, based on geographic coordinates is displayed below here [44]:

```
http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}.
```

Since for every pair of coordinates, it can be made a request, to avoid making a lot of them, which means overload the server space and increase the data download time, the algorithm is configured to send a request every Δ kilometers, where Δ is an integer value imposed by the user.

The downloaded data were used to compute several parameters, like the air density and the relative wind velocity, as will be explained in paragraph 3.7.

Note that, where necessary, the downloaded data was converted according to the European metric of measure.

3.3 VEHICLE LONGITUDINAL DYNAMICS

Since the algorithm here developed is conceptually created following a model-based approach, it was necessary to implement the main formulas to compute the power needed by the vehicle to complete the trip, and so for the computation of the energy consumption.

In the algorithm, the fundamental principles of physic relative to the vehicle design, and in particular the Newton's second law of motion, were used.

In the vehicle, during the motion, several forces act on it and the resultant force governs the motion according to the Newton's law, Figure53.

The electric motor, that is the propulsion unit, gives to the vehicle the force to go forward and overcome the resisting forces due to the air, rolling friction, gravity resistance and so on. [45]

In this paragraph the equation showed is based on the Newton's second law, relative to the vehicle longitudinal dynamic.

Starting with an overview of the forces acting on the vehicle, in the next sub-paragraph will be described each of these ones.

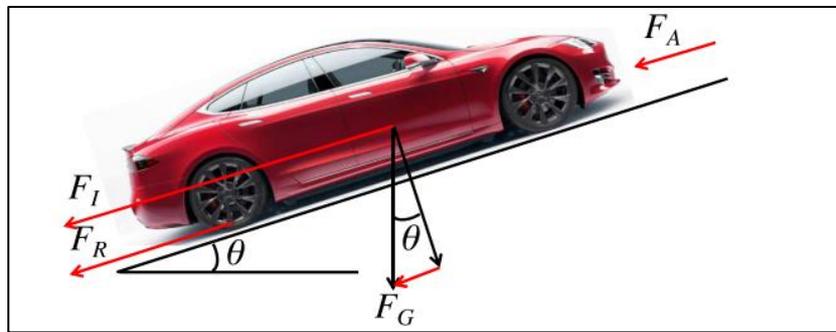


Figure 53. Total forces acting on a BEV [46].

For the Newton's second law of motion [45], the acceleration of the vehicle can be expressed as

$$\dot{v} = \frac{\Sigma F_W - \Sigma F_{res}}{M_{eq}} = \frac{\Sigma F_W - \Sigma F_{res}}{M_c \cdot (1 + \lambda)} \quad \left[\frac{m}{s^2} \right] \quad (3.11)$$

where:

- $v \left[\frac{m}{s} \right]$, is the longitudinal vehicle speed.
- $\Sigma F_{res} [N]$, is the total resistance force.

- ΣF_W [N], is the total traction force at wheels level.
- M_{eq} [Kg], is the equivalent real mass of the vehicle.
- M_c [Kg], is the vehicle mass.
- λ , is a factor used to convert the rotational inertias of the rotational components in translational masses.

However, it is necessary to be clearer about the term at the denominator of the equation 3.10. The equivalent real mass of the vehicle, indicated with M_{eq} , is given by $M_c + \left(\frac{J_W \cdot 4}{r^2} + \frac{J_m \cdot i^2}{r^2} \right)$, where J_W, J_m, r, i are respectively, the inertia of the wheels [Kg · m²], the inertia of the motor [Kg · m²], the radius of the tires [m], and the gearbox ratio [-]. [27] Since several of these data about the vehicle were not available, the rotational part components have been substituted with a factor expressed as $\lambda \cdot M_c$, obtaining so $M_{eq} = M_c \cdot (1 + \lambda)$. [45]

3.3.1 TOTAL TRACTION FORCE

The total traction force, also defined as road load force, at wheels level (F_W) is the sum of all the forces that act on the vehicle, obtained from longitudinal dynamic equation. [45]

Rearranging the equation 3.11, the total traction force, needed by the driving wheels to guarantee the forward motion, can be expressed as

$$F_W = F_G + F_{AIR} + F_R + F_I \quad [\text{N}] \quad (3.12)$$

Each of the elements of the equation 3.12, is described below:

$$- F_G = M_c \cdot g \cdot \sin(\theta) \quad [\text{N}], \text{ is the gravitational force component.} \quad (3.13)$$

$$- F_{AIR} = 0.5 \cdot C_d \cdot A_f \cdot \rho_{air} \cdot v_{wc}^2 \quad [\text{N}], \text{ is the aerodynamic force component.} \quad (3.14)$$

$$- F_R = f_d \cdot M_c \cdot g \cdot \cos(\theta) \quad [\text{N}], \text{ is the rolling force component.} \quad (3.15)$$

$$- F_I = M_c \cdot (1 + \lambda) \cdot \dot{v} = M_{eq} \cdot \dot{v} \quad [\text{N}], \text{ is the inertia force component.} \quad (3.16)$$

where:

- M_c [Kg], is the mass of the vehicle.

- M_{eq} [Kg], is the equivalent mass of the vehicle.
- g [$\frac{m}{s^2}$], is the gravitational term.
- θ [rad], is the slope angle of the road.
- C_d [-], is the aerodynamic drag coefficient.
- A_f [m^2], is the frontal area of the vehicle.
- ρ_{air} [$\frac{Kg}{m^3}$], is the air density coefficient.
- v_{wc} [$\frac{m}{s}$], is the relative wind speed.
- f_d [-], is the rolling resistance coefficient.
- v [$\frac{m}{s}$], is the longitudinal vehicle speed.
- \dot{v} [$\frac{m}{s^2}$], is the longitudinal vehicle acceleration.

The contribution of each term of the equation 3.12 is displayed in Figure 54.

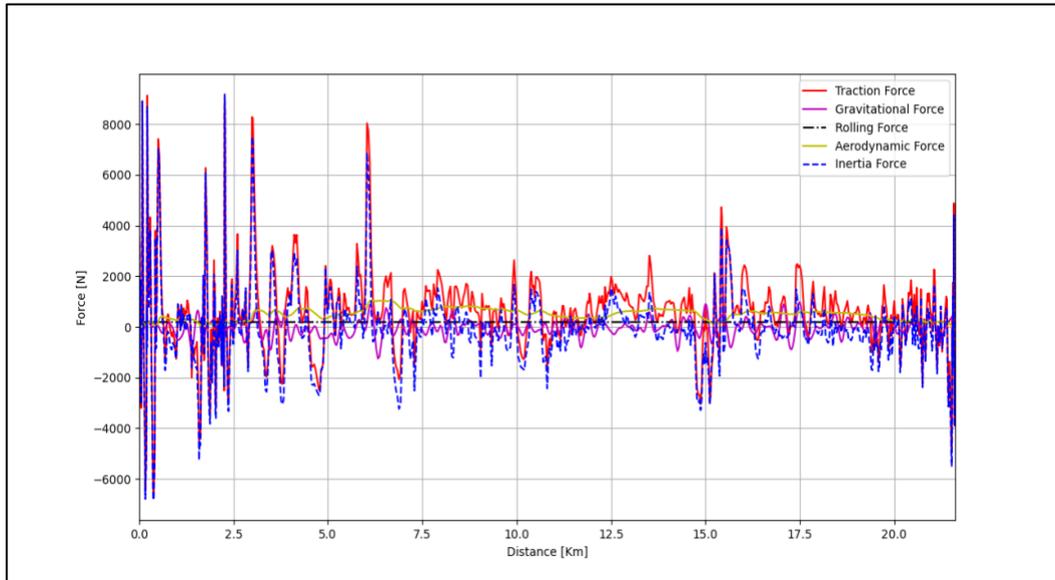


Figure 54. Comparison of all the forces acting a vehicle according to the longitudinal dynamic equation.

From the Figure 54, it is possible to observe how the component of the force that major influences the traction force is the inertia (in blue), then there is the aerodynamic term (in yellow), that is almost equivalent with the gravitational force (in purple) and to conclude, the less influencing term, is the rolling friction term (in black).

Since the inertia has the biggest influence on the total force spent to move the vehicle, it will be necessary to estimate as best as possible the acceleration and so the velocity profile.

3.3.2 TOTAL TRACTION POWER AT WHEELS LEVEL

To compute the total traction power delivered, at wheels level (P_W), both the vehicle speed (v) and the total traction force (F_W) are needed. The equation used is the 3.17. [41]

$$P_W = F_W \cdot v = (F_G + F_{AIR} + F_R + F_I) \cdot v \quad [W] \quad (3.17)$$

At this point, the algorithm merges all the data previously collected to compute the longitudinal dynamic force at wheel stage (F_W) and the relative power (P_W), according to equations 3.12 and 3.17.

In Figure 55, are showed each component of the total traction power (P_W), so the gravitational, the rolling friction, the aerodynamic and the inertia power.

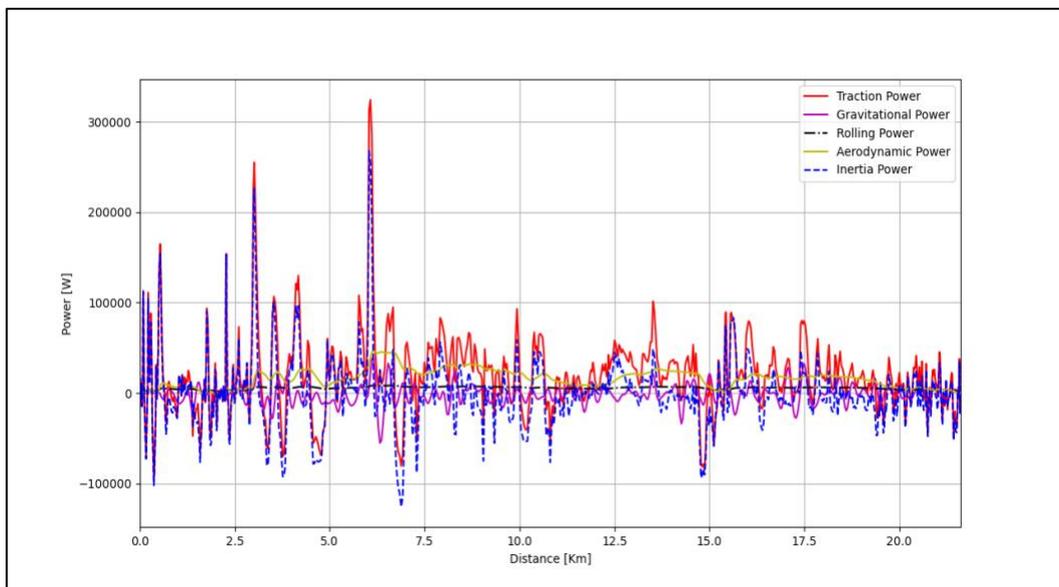


Figure 55. Comparison of all the power acting a vehicle according to the longitudinal dynamic equation.

Since a battery electric vehicle is studied, the power can have both positive and negative values. This behavior is due to the presence of the electric motor, that can work in two operational conditions, as motor and as generator. In Figure 55, is showed the power consumption at wheel stage and it is possible to evidence both positive and negative values.

Positive values ($P_W > 0$) indicate that the vehicle is in acceleration mode and so the electric motor sends power from the battery system to the wheels for moving forward the vehicle, Diagram 1.

The opposite situation, ($P_W < 0$) is when the vehicle is in deceleration mode, that could happen in two scenarios: when the brake pedal is pressed or when the throttle is left free. In both these situations the motor turns in generator mode and the battery starts to

recover power, and so energy, moving the quantity from the wheels to the battery, Diagram 2.

Instead, the case with power equal to zero ($P_W = 0$) is referred to the vehicle in rest position, that happens when the velocity is null.

Note that, the case of recovering power when the throttle pedal is left free happens not in all the battery electric vehicles, but only in those ones that have implemented this function, e.g., in Tesla cars there is deployed the Regenerative Braking System (RBS), also called regen.

The name regenerative derives from the fact that the system converts the vehicle's kinetic energy into chemical one to be stored in the battery system, where it can be used again by the vehicle. Instead, braking term come from the fact that it serves also to slow the vehicle velocity, without pressing the brake pedal. [47]

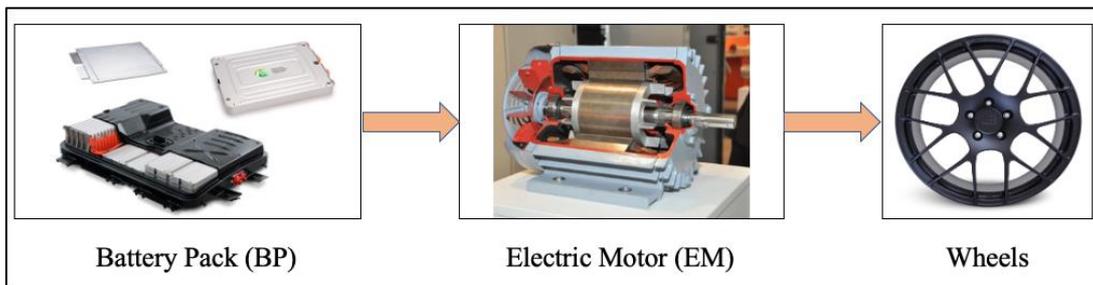


Diagram 1. Power flow in case of power consumption in BEV. The arrow indicates the direction in which the power goes.

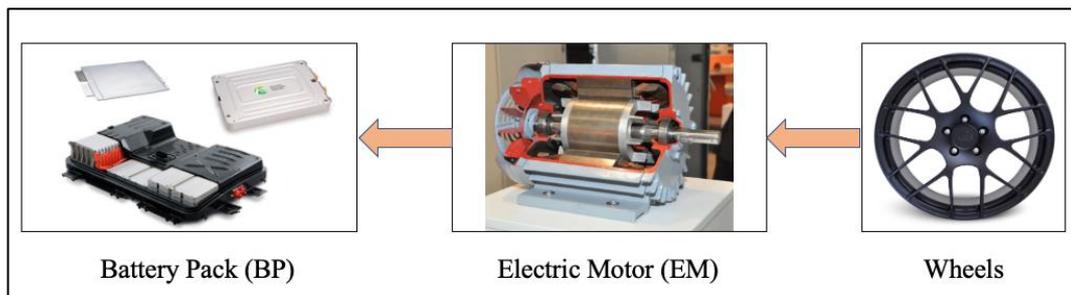


Diagram 2. Power flow in case of power recovery in BEV. The arrow indicates the direction in which the power goes.

The power flow described in Diagram 1 (or Diagram 2), is referred to a simple vehicle model, in fact in a more realistic cases, the vehicle's subsystems are more. Figure 56, 57 show examples of electric vehicle architecture systems with associated efficiency terms.

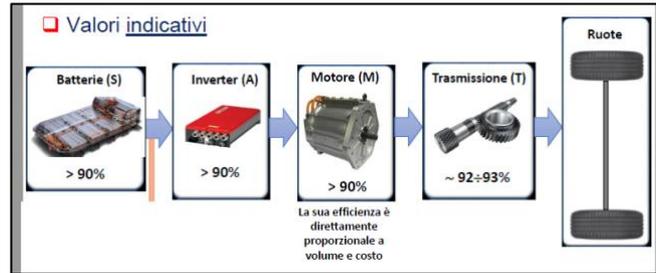


Figure 56. Main components of a total electric vehicle. [44]



Figure 57. Pre-flight briefing of Volkswagen VW ID.3.

3.3.3 POWER CONVERSION FROM WHEELS TO BATTERY SIDE

This chapter describes the battery power estimation of a BEV, in terms of both formulations needed to obtain this scalar quantity and technical too.

Since the goal is to find an estimation of the SOC consumption, the energy delivered by the battery system has to be obtained but so also the battery power used by the vehicle. In a battery electric vehicle, the power is delivered from the wheels to the battery system, passing from the differential and the electric motor, according to the Diagram 1. In this work, it was decided to avoid investigating in detail all the subsystems composing the powertrain, due mainly to the lack of time and resources but are used only some of the systems in Figure 56.

For this reason, an approach based on simple equations, based on efficiency coefficient η , was used to describe the exchange of power between two different subsystems. The idea comes from the knowledge that when two different systems exchange power, some of the power exchanged is loss and the ration between the useful output power of an energy conversion machine and the input, in energy terms, is defined as efficiency conversion coefficient (η). The previous definition can be applied to every system, where the input, as well as the useful output, are chemical, electric power, mechanical work, light (radiation), or heat. [48]

In formula, the efficiency conversion term is expressed according to equation 3.18.

$$\frac{P_{OUT}}{P_{IN}} = \eta \quad [-] \quad (3.18)$$

The energy coefficient term is in range $[0 \div 1]$ and can be expressed in percentage.

Starting from the result achieved in the sub-paragraph 3.4.2, that was the power at wheels level (P_W), here the goal was to obtain an estimation of the battery power (P_{BATT}) consumption, related to a trip.

To obtain this result, the conversion formulas, relative to the transmission of power from a system to another one, was needed to be considered.

Starting by the general equations listed from 3.19 to 3.21, and by manipulating them, it was possible to achieve the battery power consumption values from the wheels power, both in the case of power supplied and absorbed.

$$\frac{P_W}{P_{EM}} = \eta_{EM} = 0.95 \quad [-] \quad (3.19)$$

$$\frac{P_{EM,SUPPLIED}}{P_{BATT,SUPPLIED}} = \eta_{BATT} = 0.93 \quad [-] \quad (3.20)$$

$$\frac{P_{BATT,ABSORBED}}{P_{EM,ABSORBED}} = \eta_{BATT,REG} = 0.60 \quad [-] \quad (3.21)$$

where:

- P_W [W], is the power referred to the wheels stage.
- P_{EM} [W], is the power referred to the electric motor.
- P_{BATT} [W], is the power referred to the battery system.
- η_{EM} , is the efficiency conversion coefficient relative to the electric motor.
- η_{BATT} , is the efficiency conversion coefficient relative to the battery, in consumption mode.
- $\eta_{BATT,REG}$, is the efficiency conversion coefficient relative to the battery, in regenerative mode.

The previous three equations are obtained considering a vehicle made by only 3 systems, the wheels, the electric motor with the transmission part, and the battery with the inverter, according to the schema in Diagram3.

The modern electric motor, have an efficiency around the 95% and 98%, depending on the motor typology, that is bigger than the heat motors, whose efficiency is about 40%. [49]

Instead, regard the battery system, typical values of efficiency factors are greater or equal to 90% and this depends on the quality of the battery system and on its health state;

instead, in the scenario where the battery is in recharging mode, using the power coming from the electric motor that works as generator, the efficiency is more than 50%. The flow of the conversion process executed is described in Diagram 3.

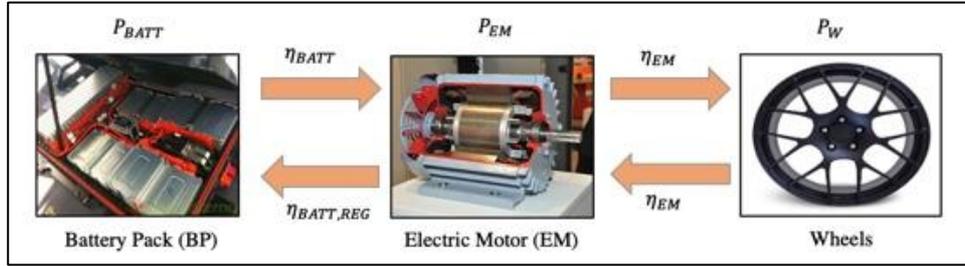


Diagram 3. Power flow exchanged by the subsystems considered in the algorithm.

The conversion between the power consumption at wheels level (P_W) and at electric motor level (P_{EM}) was obtained according to the equations 3.22, and 3.23.

$$P_{EM,SUPPLIED} = \frac{P_W}{\eta_{EM}} \quad [W] \quad (3.22)$$

$$P_{EM,ABSORBED} = P_W \cdot \eta_{EM} \quad [W] \quad (3.23)$$

$$P_{EM} = P_{EM,SUPPLIED} + P_{EM,ABSORBED} \quad [W] \quad (3.24)$$

Figure 58 shows the effect of the power conversion from the wheels to the electric motor level, and vice versa, achieved according to the previous equations.

The result obtained is correct since focusing on the positive values of the power ($P > 0$), it is possible to observe that the electric motor generates more power that the quantity used by the wheels to move the vehicle, and this is due to the fact the electric motor efficiency is about 0.95 and this means that a very little part of the power send will be lost. The same is true in the opposite situation, for ($P < 0$).

Once the electric motor power was accomplished, the battery power could be estimated. Specifically, it was reached both the battery power supplied ($P_{BATT,SUPPLIED}$), according to the equation 3.25, and the battery power absorbed ($P_{BATT,ABSORBED}$), equation 3.26.

$$P_{BATT,SUPPLIED} = \frac{P_{EM,SUPPLIED}}{\eta_{BATT}} \quad [W] \quad (3.25)$$

$$P_{BATT,ABSORBED} = P_{EM,ABSORBED} \cdot \eta_{BATT,REG} \quad [W] \quad (3.26)$$

$$P_{BATT} = P_{BATT,ABSORBED} + P_{BATT,SUPPLIED} \quad [W] \quad (3.27)$$

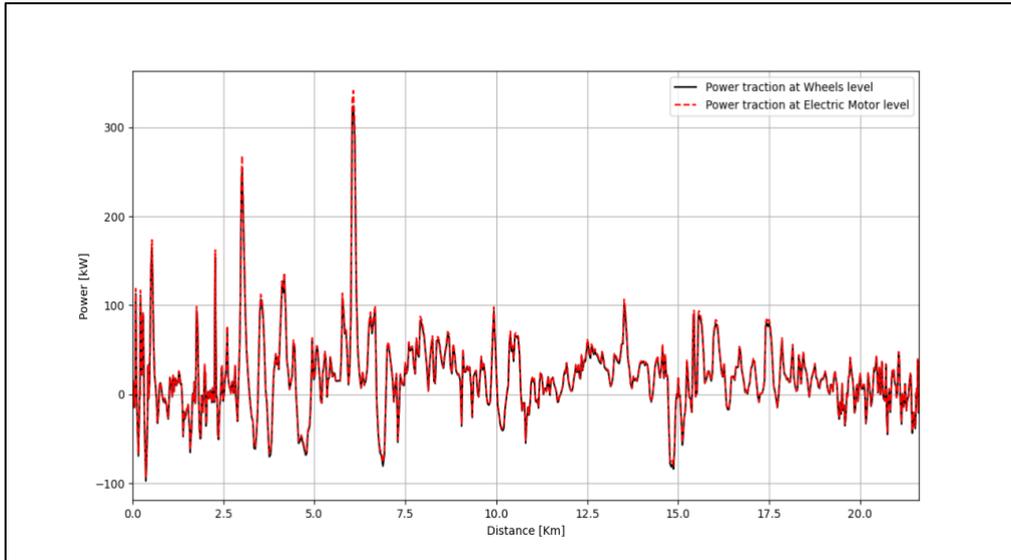


Figure 58. Comparison between the power consumption at wheels level (black) and power consumption at electric motor level (red), associated to a trip.

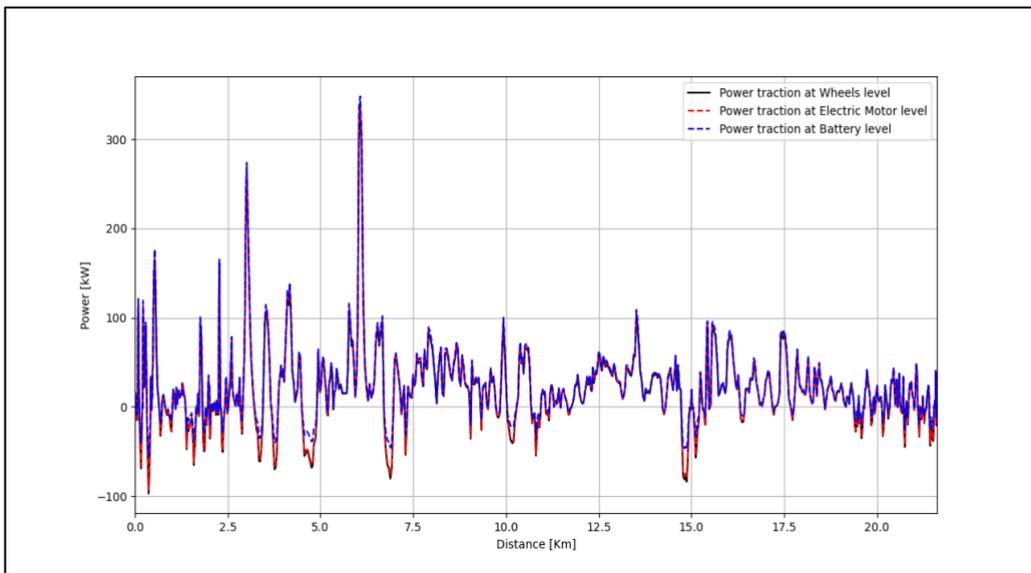


Figure 59. Comparison among the power consumption at wheels level (black), power consumption at electric motor level (red) and power consumption at battery level (blue), associated to a trip.

The plot in Figure 59 puts in evidence the goodness of the results obtained, in fact the battery power supply, for positive values, ($P_{BATT} > 0$) is higher than the electric motor one, as is reasonable to expect.

Observing the negative power values ($P_{BATT} < 0$), the differences between the electric motor and battery power are due to the usage of a $\eta_{BATT,REG} = 0.60$, that means the algorithm has supposed that only the 60% of the power absorbed by the electric motor as generator is transmitted in the battery; in real cases, there is a more sophisticated recovery system, that allows to recover more power.

The quantity measured until now is the power associated to the motion of the vehicle, but in real scenario it is influenced by the so-called auxiliaries' systems (P_{AUX}).

To have a more accurate state of charge estimation, it is necessary also consider the power consumption due to several loads typically present in a vehicle, as the lights, the air conditioning system, or the windscreen wipers.

However, evaluating the impact that each of these systems have on the total power term is not an easy task, and moreover requires to have a very deep knowledge about them.

Since is not simple at all make this type of analysis, in this thesis it was assumed that the relative P_{AUX} value was about 600 [W].

From now on, the battery power (P_{BATT}) will be indicated according to equation 3.28.

$$P_{BATT} = P_{BATT} + P_{AUX} \quad [W] \quad (3.28)$$

3.4 ENERGY CONSUMPTION

In this chapter are described all the equations to calculate the energy associated to a power consumption. Starting from the definition of the work, it goes to the enunciation of the relationship between power and work and then, to the energy description.

After the battery power used to complete a trip is obtained, the algorithm needs to compute the associated energy consumption.

From physical knowledge, it is known that the work (W) can be defined as the amount of the energy (E) exchanged from two systems by a force (F). The work is a scalar quantity with SI unit of Joule (J).

The work can be defined as the dot product between the force (F) and the displacement (Δx), that in formula is

$$W = F \circ x = |F| \cdot |\Delta x| \cdot \cos(\alpha) \quad [J] \quad (3.29)$$

Note that ' \circ ' means dot product, while ' \cdot ' simply product.

Since in this work are evaluated the longitudinal forces necessary to guarantee the motion of the vehicle, it was assumed that vectors F and Δx have the same direction and verse, so are parallel vectors.

This means that $\alpha = 0^\circ$, and so equation 3.29 can be rewritten as

$$W = F \cdot x \quad [J] \quad (3.30)$$

At this point, it is possible to define the power as the work done in a unit of time, or, in other words, as the derivative of the work in time. The measure unit of the energy in SI is the Watt (W).

In formula, the power is expressed as

$$P = \frac{dW}{dt} \quad [W] \quad (3.31)$$

By substituting the equation 3.30 in 3.31 and remembering that the velocity is the derivative of the displacement in time, the power can be also defined as the force times the velocity, in accord to equation 3.32.

$$P = \frac{dW}{dt} = \frac{d(F \cdot x)}{dt} = \frac{F \cdot dx}{dt} = F \cdot v \quad [W] \quad (3.32)$$

Since the energy is stored in the work, it is possible to write:

$$P = \frac{dW}{dt} = \frac{dE}{dt} \quad [W] \quad (3.33)$$

Integrating both the side of the previous equation, the energy can be computed as the integral of the power in an amount of time.

$$E(t_f) = E(0) + \int_0^{t_f} P(t) \cdot dt \quad [J] \quad (3.34)$$

According to the nomenclature given until now in this thesis, the energy associated to the battery power consumption (E_{BATT}) is determined as

$$E_{BATT}(t_f) = E_{BATT}(0) + \int_0^{t_f} P_{BATT}(t) \cdot dt \quad [J] \quad (3.35)$$

where the t_f is the final driving time, in seconds [s], $E_{BATT}(0)$ is the initial energy, assumed to be zero, and P_{BATT} is the battery power. Since in automotive, but in general in battery systems, the energy is usually referred in Kilowatt hour (or kWh) and not in Joule, the conversion is necessary. To pass from J to kWh, the energy values have to be divided by $3.6 \cdot 10^6$. So, in short, the algorithm takes the estimated battery power and, applying a discrete integration, returns the associated energy consumption.

3.5 SOC ESTIMATION

The state of charge is maybe the most important parameter to be estimated in battery a management system, of the EV, since among the several target in which it is used, it can help, the driver to understand if it can complete the trip or not. As described in the chapter 2, there are several ways to compute the SOC using both offline and online methods, but in this algorithm an offline approach is used. To obtain the estimation of the SOC consumption, it was noted a strong relationship between the state of charge and the energy consumption relative to a certain trip.

Firstly, it is observed that from the state of charge measurement, it was possible to derive the energy consumption (E_{SOC}) to complete a trip, according to equation 3.36.

$$E_{SOC} = C_{nominal} \cdot \left(\frac{SOC(t_f) - SOC(t_i)}{100} \right) \quad [\text{kWh}] \quad (3.36)$$

where:

- $SOC(t_f)$ is the final state of charge value.
- $SOC(t_i)$ is the initial state of charge value.

This formula from a dimensional analysis works fine but, to check if the obtained result is good or not, it was necessary to compare the energy consumption obtained using the equation 3.36 with the real on, equation 3.35.

In Figure 61 can be seen a comparison between the E_{SOC} and $E_{BATT,R}$. numerical values are extracted and reported in Table 4. The MAE and RMSE values obtained by considering the two quantity are showed in Figure 60.

	E_{SOC}	$E_{BATT,R}$
Energy consumption	19.43 [kWh/100Km]	20.22 [kWh/100Km]
Final Energy value	4.2 [kWh]	4.37 [kWh]

Table 4. Energy consumptions values relative to the quantity obtained using the equation 3.36 and the 3.35.

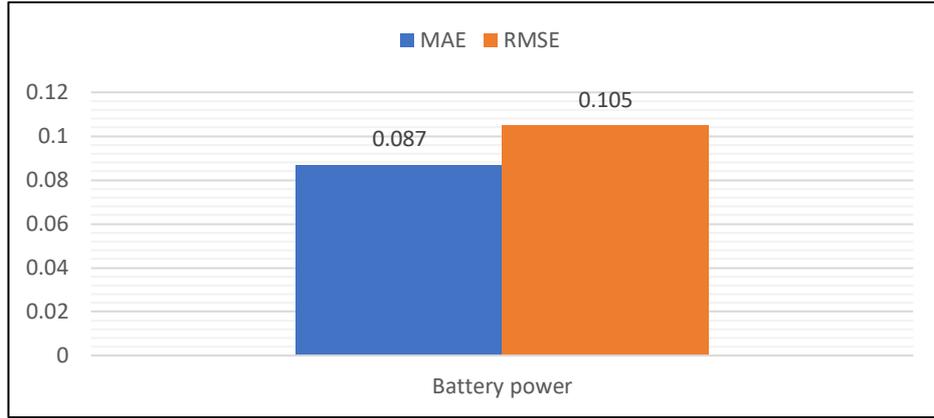


Figure 60. RMSE and MAE values associated to the accuracy computed from data in Figure 61, plot (a).

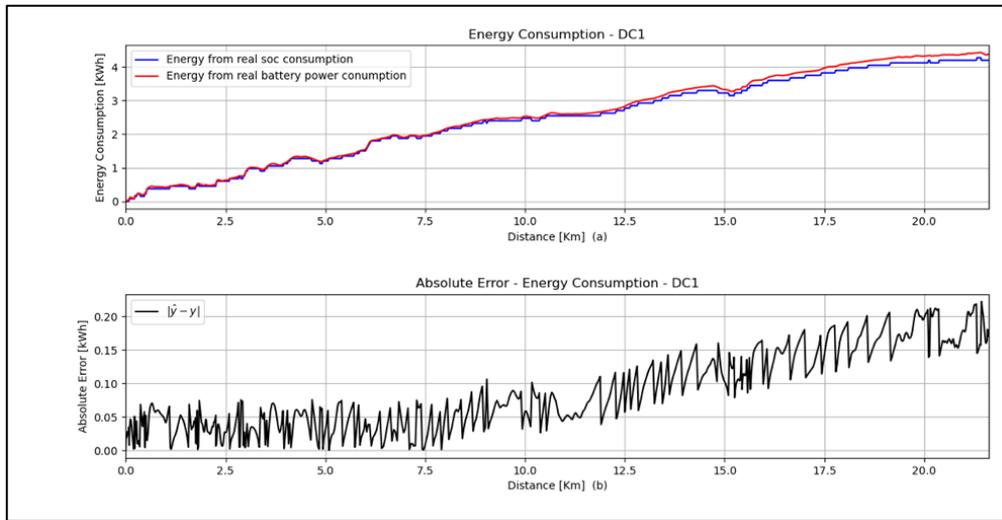


Figure 61. Comparison between the energy computed from measured battery power (eq. 3.35) and energy from measured SOC (eq. 3.36) (a). Absolute error between the two quantities (b).

Since these two values are very close, and the error is very low, this means that the formula 3.36 works and can be used.

The equation to compute the SOC consumption is obtained by rearranging the equation 3.35, under the hypothesis to know the battery energy consumption E_{BATT} , the initial value of the state of charge $SOC(t_0)$ and the nominal battery capacity ($C_{nominal}$). It is expressed by the equation 3.37.

$$SOC(t) = SOC(t_0) - \frac{E_{BATT}(t)}{C_{nominal}} \cdot 100 \quad [\%] \quad (3.37)$$

3.6 PARAMETERS IDENTIFICATION

Obviously, a model-based approach is founded on the physical knowledge of the problem and in this work the starting point is the Newton's second law of the motion, equation 3.11, so several parameters must be identified to be used in the previous equations and to obtain better results. Some of these parameters are constant, e.g., the gravitational term g or the vehicle mass M_c , other ones are strongly influenced by weather data and by other external factors and others are weakly influence by external conditions. In the algorithm, several functions are written to compute the following parameters: the air density ρ_{air} , the relative wind speed acting on the vehicle v_{wc} , the road slope angle θ and the rolling resistance coefficient f_d .

The parameters showed in Table 5, are considered always constant in time.

Some of the vehicle parameters, like the curb weight and the drag coefficient, come from the specifications of the Tesla Model 3 Long-Range Dual-Motor AWD, with a 75 kWh battery pack. [50]

Parameters	Values
Vehicle weight + 1 person Mass	1847 + 80 [Kg]
Vehicle Drag Coefficient C_D	0.23 [-]
Vehicle Frontal area A_f	2.34 [m ²]
Coefficient λ	0.05 [-]
Gravitational term g	9.81 [$\frac{m}{s^2}$]

Table 5. Constant parameters of the algorithm. [45], [50]

The next four sub-paragraphs are dedicated to the description of the function implemented in this model to compute the parameters, with a focusing of the formulations used.

3.6.1 AIR DENSITY

In the model, the air density parameter plays an important role, since it appears in the calculation of the aerodynamic force (F_{AIR}) and so, it is necessary, for having a better result, compute it for estimating as faithful as possible the total power consumed by the vehicle during a trip.

The density of the air, denoted in this work as ρ_{air} , is defined as the mass per unit of volume of Earth's atmosphere [$\frac{Kg}{m^3}$]. [51]

It mainly depends by three factors:

- P [hPa], is the atmospheric pressure.

- RH [%], is the relative humidity.
- T [°C], is the ambient temperature.

For higher temperature, it can be proven, that the humidity has a minor influence on the air pressure. According to the equations described below (3.38 – 3.43), the behavior of the air density as function of the ambient temperature and relative pressure, fixed the humidity at 70%, is showed in Figure 62.

Moreover, for the Stevino’s law, the air density depends also by the elevation profile, in fact going up the value of the pressure increase and so the air density decrease, vice versa going down.

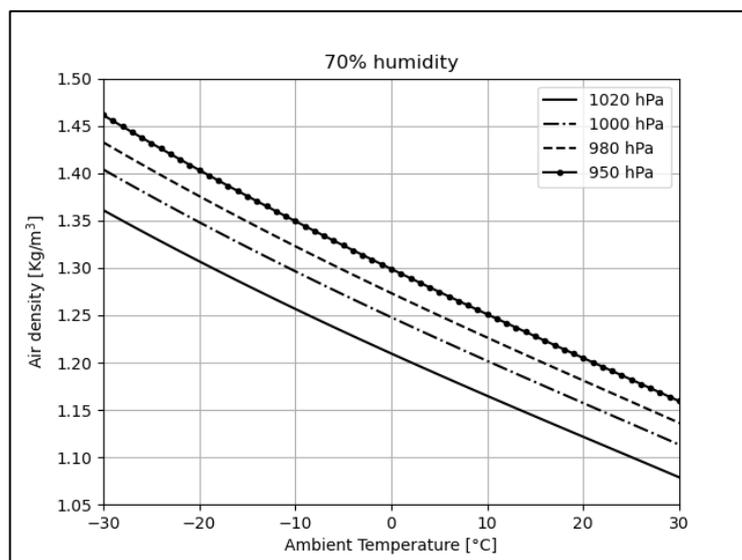


Figure 62. Air density dependency on ambient temperature and air pressure, fixed the humidity level.

For the ρ_{air} computation, was written a Python function which uses as inputs the temperature, the humidity, and the atmospheric pressure.

The function starts with the computation of the actual water vapor pressure in the air (P_v), equation 3.38. [52], [53]

$$P_v = E_s \quad \text{[hPa]} \quad (3.38)$$

Then, it can be noticed that the water vapor pressure is equal to the saturation pressure of the water vapor (E_s), computed at the dew point temperature, see equation 3.39. This equation was developed by Herman Wobus. [52], [53]

In practice, to compute the saturation pressure of the water vapor, the dew temperature must be substituted in the equation 3.40. Remember also to convert millibar [mb] to hectopascal [hPa]. [52], [53]

$$Es = \frac{es_0}{p^8} \quad [\text{mb}] \quad (3.39)$$

where:

$$p = Co + T(c1 + T(c2 + T(c3 + T(c4 + T(c5 + T(c6 + T(c7 + T(c8 + T(c9))))))))))$$

where T [°C] is the temperature measured at that moment.

$$es_0 = 6.1078$$

$$c_0 = 0.99999683$$

$$c_1 = -0.90826951 \cdot 10^{-2}$$

$$c_2 = 0.78736169 \cdot 10^{-4}$$

$$c_3 = -0.61117958 \cdot 10^{-6}$$

$$c_4 = 0.43884187 \cdot 10^{-8}$$

$$c_5 = 0.29883885 \cdot 10^{-10}$$

$$c_6 = 0.21874425 \cdot 10^{-12}$$

$$c_7 = -0.17892321 \cdot 10^{-14}$$

$$c_8 = 0.11112018 \cdot 10^{-16}$$

$$c_9 = -0.30994571 \cdot 10^{-19}$$

The total measured actual pressure (P) is the sum of the pressure of the dry air (Pd) and the vapor pressure (Pv), equation 3.40 [52], [53]:

$$P = P_d + P_v \quad [\text{hPa}] \quad (3.40)$$

At this point, rearranging the previous formula, since (P) and (Pv) are known terms, the pressure of the dry air (Pd) can be calculated, according to 3.41. [52], [53]

$$P_d = P - P_v \quad [\text{hPa}] \quad (3.41)$$

By a simple addition, the density of the air ρ_{air} is obtained using equation 3.42. [53]

$$\rho_{air} = \frac{P_d}{R_d \cdot T_K} + \frac{P_v}{R_v \cdot T_K} \quad \left[\frac{\text{Kg}}{\text{m}^3}\right] \quad (3.42)$$

where:

- $R_d = 287.0531 \left[\frac{\text{J}}{\text{Kg} \cdot \text{K}}\right]$, is the specific gas constant relative to the dry air.

- $R_V = 461.4964 \left[\frac{\text{J}}{\text{kg}\cdot\text{K}} \right]$, is the specific gas constant relative to the water vapor.
- $T_K = T \text{ (}^\circ\text{C)} + 273.15 \text{ [K]}$, is the ambient temperature measured in Kelvin degree.

Since OpenWeatherMap API returns the relative humidity value, it is convenient measure the water vapor pressure from the relative humidity, according to the equation 3.43.

The formula derives from the definition of relative humidity, according to which it is defined as the ratio of the actual water vapor pressure (P_v) to the saturation water vapor pressure (E_s) at a given temperature, moreover E_s is expressed in percentage (%). [52], [53]

$$P_v = Rh \cdot E_s \quad [\text{hPa}] \quad (3.43)$$

3.6.2 RELATIVE WIND VELOCITY

Another important parameter to compute is the relative wind speed v_{wc} , that is the relative wind between the absolute wind speed v_w and the vehicle speed v .

This is an important parameter since the aerodynamic force (F_{AIR}) is proportional to the square of the relative wind velocity, v_{wc} . The chapter starts with a short description of what the relative wind is and ends with the presentation of the formulas used to compute it.

To achieve the relative wind speed, four data are needed to be known:

- $v \left[\frac{\text{m}}{\text{s}} \right]$, longitudinal vehicle speed.
- $\alpha \text{ [}^\circ \text{ degree]}$, bearing angle of the vehicle.
- $v_w \left[\frac{\text{m}}{\text{s}} \right]$, absolute wind speed.
- $\beta \text{ [}^\circ \text{ degree]}$, wind direction.

Note that both the wind direction β and the head angle α are computed taking as reference the compass angles, so are referred with respect to the North or 0° , Figure 63.

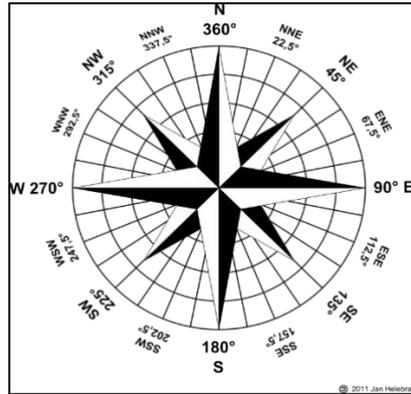


Figure 63. Example of main wind direction in a compass diagram. [54]

The main four wind directions are North (N), South (S), West (W) and East (E) and these can be combined to create new directions, Figure 63.

The measure unit of the wind are the directions themselves or the azimuth degrees, from 0-360°. [55]

A list of the principal wind direction is:

- 0° = 360° is a North wind
- 45° is a Northeast wind
- 90° is East wind
- 135° is a Southeast wind
- 180° is a South wind
- 225° is a Southwest wind
- 270° is a West wind
- 315° is a Northwest wind

A particularity of the wind direction comes from the definition itself, in fact, the wind direction is defined as that from which the wind blows, and so this means that a North wind (0° or 360°) comes from the North and blows toward the South and so, graphically it is represented by an arrow whose direction is from North to South.

The first operation made by the Python function, written to compute the relative wind, is to change the angle associated to the wind direction. This is done for the purpose of changing the view from which the wind direction is related to where the wind blows and not from which it comes, direction computed always with respect to the North.

An example is presented below to explain better the idea, Figure 64.

If a wind whose direction $\beta = 30^\circ$ is considered, the function at first time computes the new angle $\beta' = 180^\circ + \beta$. Note that since the angle range is in 0-360°, if the angle β is greater or equal than 180° the new angle $\beta' = (180^\circ + \beta) - 360^\circ$; otherwise $\beta' = 180^\circ + \beta$.

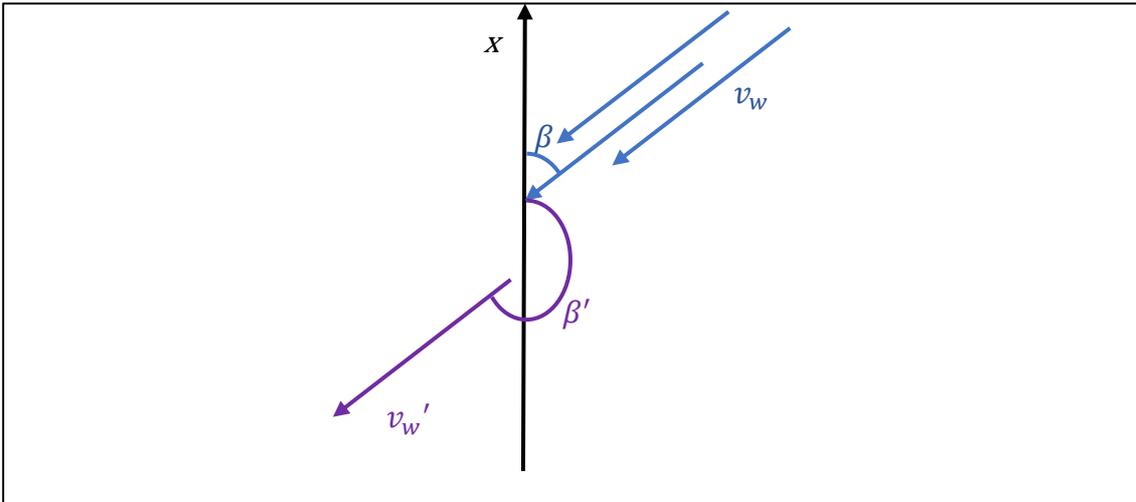


Figure 64. Diagram representing a wind blowing from NE (30°). The black arrow is the vehicle longitudinal direction, the blue one is the wind direction, and the purple is the new wind direction.

From here all the equations used to compute the relative wind are showed.

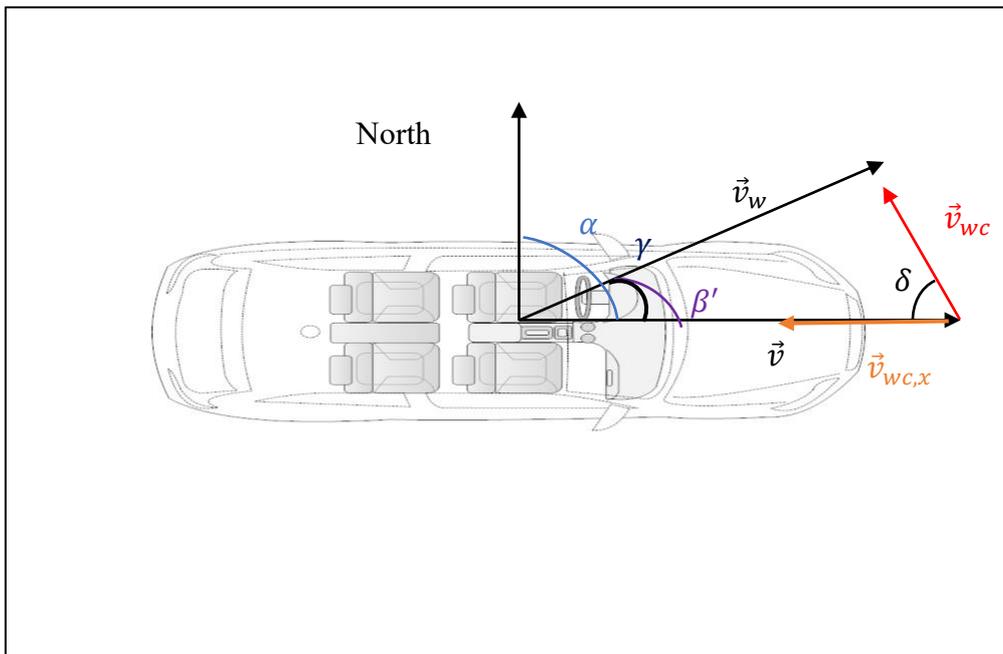


Figure 65. Diagram with the main vectors to be considered in relative wind speed computation.

By definition, the relative wind speed is the difference between the absolute wind speed v_w and the longitudinal vehicle speed v , that in formula it can be written as

$$v_{wc} = v_w - v \quad \left[\frac{m}{s} \right] \quad (3.44)$$

Using a graphical approach, it is possible to obtain the relative wind speed, in terms of absolute value (v_{wc}) and application angle (δ) of the vector, Figure 65.

To obtain the modulus of the relative wind speed, the cosine's theorem can be used, equation 3.45; instead, for the angle, the sine's theorem has to be used, equation 3.46.

$$|V_{wc}| = \sqrt{|v_w|^2 + |v|^2 - 2 \cdot |v_w| \cdot |v| \cdot \cos(\gamma)} \left[\frac{m}{s} \right] \quad (3.45)$$

$$\frac{|v_w|}{\sin(\delta)} = \frac{|v_{wc}|}{\sin(\gamma)} \quad (3.46)$$

The angle $\gamma = \alpha - \beta'$.

Rearranging the equation 3.46, it is possible to evidence and extract the angle δ .

$$\delta = \sin^{-1} \left(\left(\frac{|v_w|}{|v_{wc}|} \right) \cdot \sin(\gamma) \right) \quad [^\circ \text{ degree}] \quad (3.47)$$

At the end, the longitudinal component of the relative wind speed is computed as

$$|v_{wc,x}| = |v_{wc}| \cdot \cos(\delta) \quad \left[\frac{m}{s} \right] \quad (3.48)$$

The last term, obtained from equation 3.48, is what the function returns as output and that is used in the F_{AIR} computation.

Analyzing the previous formulations, it can be noted as when the vehicle is in rest position, the relative wind velocity is equal to the wind velocity (both in module and in direction). Instead, when the vehicle is at high velocity, the relative wind velocity is very close to the vehicle speed, while the angle δ tends to decrease and, therefore, the relative wind direction tends to be aligned to the vehicle longitudinal direction.

Another important consideration to make is that, in city environment, the presence of the buildings around the vehicles, attenuate the wind effect on the car, and so the wind effect could be approximated as null. While in highways or primary road, where the presence of constructions is weak, the wind effect on the car has a major impact and so it is important to consider it. However, the algorithm computes it always, for every possible scenario.

3.6.3 ROAD SLOPE

Another important parameter to estimate is the road slope coefficient θ , also called slope or grade road. It depends by two factors: the altitude level and the length of the segment route considered. The slope refers to the tangent of the angle of the surface, on the horizontal line.

To compute this coefficient value, a function was realized; it takes as input the length Δd and the altitude Δh of a single road segment and, after a processing of engineering work, it returns the slope level, both in degree and in percentage, Figure 66. Obviously, this is iterated for each segment that makes the road.

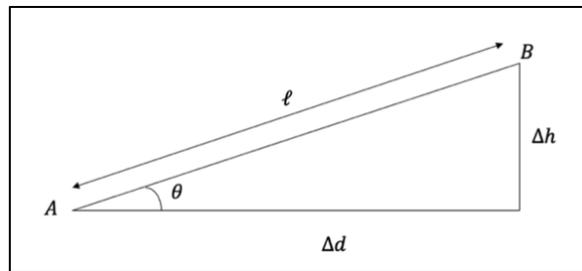


Figure 66. Example of road with a slope. Δd = distance from A to B, Δh = altitude level from A to B, l = total length, θ = slope angle.

The starting equation is

$$\tan(\theta) = \frac{\Delta h}{\Delta d} \quad [-] \quad (3.49)$$

From 3.49, by using inverse trigonometric formula, it is possible to extract the angle θ , as

$$\theta = \arctan\left(\frac{\Delta h}{\Delta d}\right) \quad [\text{rad}] \quad (3.50)$$

Note that the angle could be expressed in percentage terms, simply using the formula 3.51.

$$\theta = \frac{\Delta h}{\Delta d} \cdot 100 \quad [\%] \quad (3.51)$$

As seen in the previous paragraphs, the road slope parameter is necessary for computing both the gravitational force F_G and the rolling force F_R . Figure 67 shows an example of slope profile of a selected route.

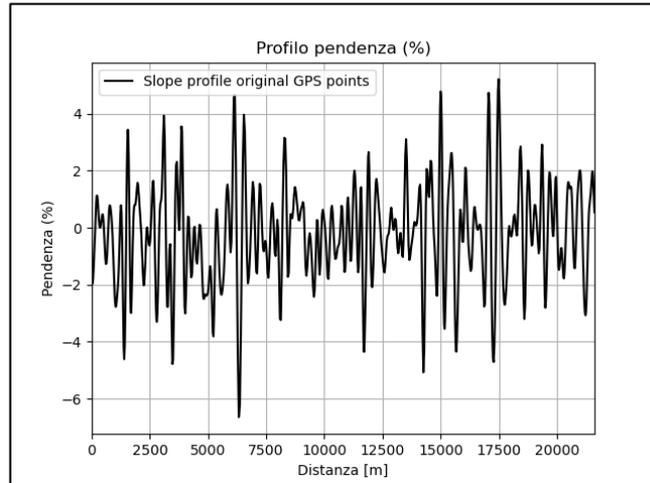


Figure 67. Example of the slope profile, expressed in percentage.

3.6.4 ROLLING RESISTANCE COEFFICIENT

The last parameter that the model developed computes, is the rolling resistance coefficient f_a . This element is not easy to estimate, since it depends by a lot of factors, such as the road typology, the surface type, the velocity of the vehicle, the weather conditions and many more factors. In Table 5, it is possible to see several f_a reference values, according to different scenarios. [45]

Conditions	Values
Car tire on smooth tarmac road	0.01 [-]
Car tire on concrete road	0.011 [-]
Car tire on a rolled gravel road	0.02 [-]
Unpaved road	0.05 [-]
Loose sand	0.15-0.3 [-]

Table 5. Several reference values of rolling resistance coefficient. [45]

Since it was not possible to make some experimental test on the vehicle, like the cost-down test, the only way to compute the rolling resistance coefficient was to use empirical formulations, according to the given data. [27]

The one chosen, for this model, was the equation 3.52, that derives the rolling resistance coefficient as function of the vehicle speed.

$$f_a = f_0 \cdot \left(1 + \frac{v}{160}\right) \quad [-] \quad (3.52)$$

where:

- f_a [-], is the rolling resistance coefficient.

- v [$\frac{km}{h}$], is the vehicle velocity.
- f_0 [-], is the nominal rolling resistance coefficient.

The previous equation can predict the values of f_d with an acceptable accuracy for speed up to 128 km/h. [16]

Since all the experimental data are obtained by trips made on route whose surface type is the asphalt, the $f_0 = 0.011$. In the algorithm, the rolling resistance coefficient is imposed constant, with a value of 0.011.

3.7 VALIDATION OF THE ALGORITHM

Goal of this paragraph is to check if the implemented formulations, starting with the longitudinal dynamics equation of a vehicle and ending with the state of charge level of consumption, have worked. Moreover, to ensure the accuracy of the algorithm in the SOC estimation. To perform this verifications, a certain number of drive cycles were extracted from the data and used as test samples.

The hypothesis done for this validation part are:

- $E_{BATT}(t_0) = 0$ [kW].
- $\widehat{SOC}(t_0) = SOC(t_0)$.
- $C_{nominal} = 75$ [kWh].

The first hypothesis holds up since it is supposed that the model works offline, and no memory effect of the previous travel is present. So, this means that when a trip ends, all the variables are cleaned and imposed to their initial values.

The second hypothesis is very strong and can be assumed only if there is a system able to measure the SOC initial value; in real application scenario, it is assumed a $\widehat{SOC}(t_0) = 90\%$. The third one is associated to the nominal capacity value of the battery pack installed on the Tesla Model 3; in a real scenario, this parameter changes in the years due to the battery consumption. [50]

Note that in this validation test, the only real measured data used are: the velocity profile, the battery power consumption (obtained by multiplying the voltage [V] and the current [A]) and the SOC level of consumption. All the other data are extracted from internet, by using APIs and web services, according to what it was described in the previous paragraphs of the chapter. An example of analysis is done on the first drive cycle, called DC1 and described in the next sub-paragraph.

3.7.1 DRIVE CYCLE, DC1

DC1 corresponds to a route whose total travel length is about 22 Km. The initial elevation is 273 m and the final one 223 m, with a range of -50 m. The measured SOC consumption goes from 89.4 % to 83.8 %. A resume of the main route characteristics is showed in Figure 68.

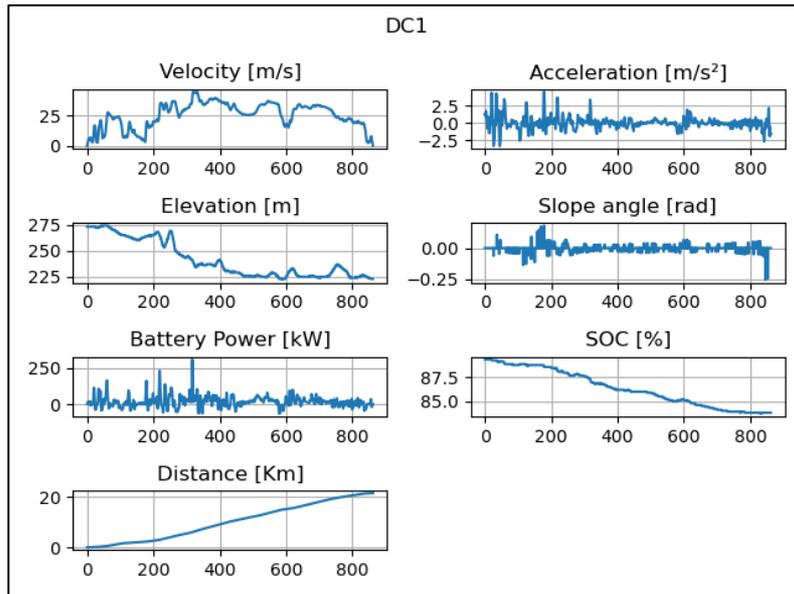


Figure 68. Some data about the drive cycle 1.

After extracted the other necessary data, such as the weather information, the elevation profile, and the road slope, the algorithm computes an estimation of the battery power, using equation 3.28. In Figure 69 it is possible to check the real battery power consumption with respect the estimated one.

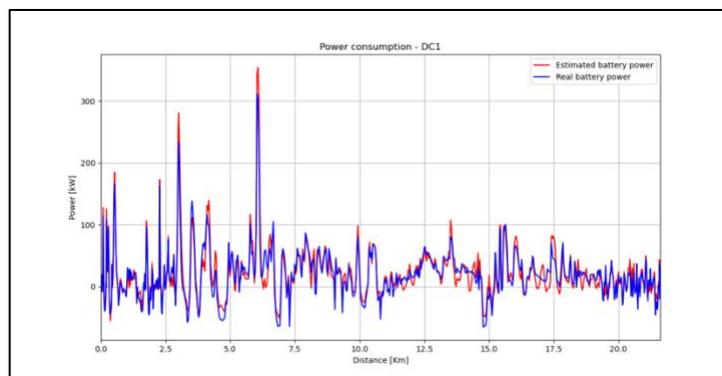


Figure 69. Comparison between real and estimated power consumption associate to DC1.

Using equations 3.35 and 3.37 the energy and the SOC level of consumption are obtained. In Figure 70 it is possible to check a comparison between the real energy and the estimated one, and the same is done for the SOC in Figure 71. Also, in the figures, the absolute error, computed as: $|e| = |\hat{y} - y|$ is showed. The most important values, extracted from these two plots, such as the energy consumption and the final SOC value, are showed in Table 6.

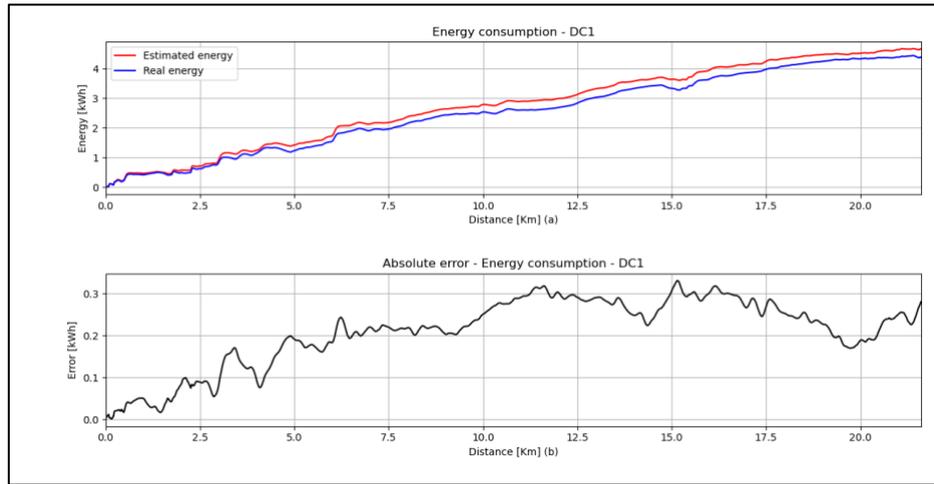


Figure 70. Comparison between real and estimated energy consumption (a) and the associated absolute error (b), relative to DC1.

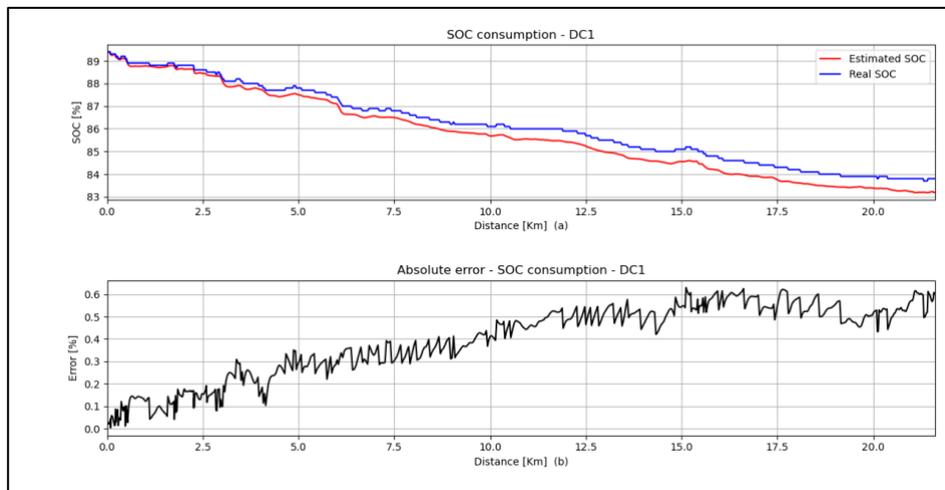


Figure 71. Comparison between real and estimated SOC consumption (a) and the associated absolute error (b), relative to DC1.

	Real Measurements	Estimated Measurements
Energy consumption1	4.37 [kWh]	4.65 [kWh]
Energy consumption2	20.22 [kWh/100Km]	21.52 [kWh/100Km]
SOC final value	83.8 [%]	83.2 [%]

Table 6. Numerical information about the energy and consumption associated to DC1.

In Figure 72 a resume of the algorithm performances in terms of RMSE and MAE, all computed for the estimation of the three quantities: the battery power, the energy, and the SOC, is showed.

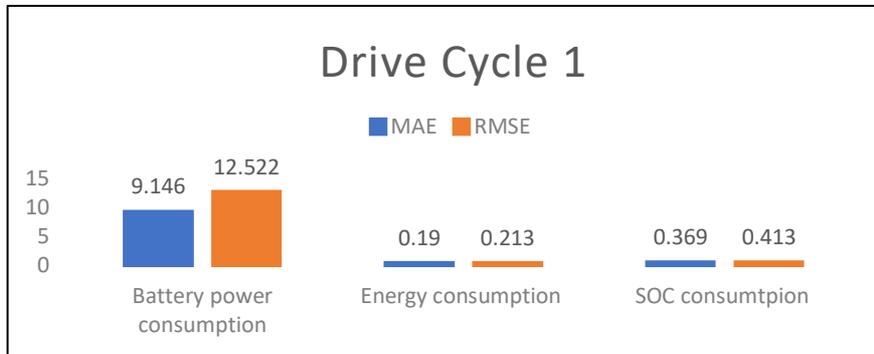


Figure 72. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC1.

In this drive cycle, the performances measured with respect to the SOC are very good, instead, for the battery power the results are a bit worse. The goodness of the model can be observed checking the comparison between the real energy consumption and the estimated one. Obviously, some errors were expected, and these are due, mainly, to two problems. The first one is associated to the usage of efficiency conversion factors η to explain the relationship among the power deliver from wheels level to the battery one, and vice-versa. The other source of error is the integration function used to obtain the energy consumption from the power. The integrator, in fact, introduces a systematic error that increases over the time and cannot be removed. This drawback can be observed in the Figure 70 and 71, in subplot b. Nevertheless, the algorithm seems to estimate the state of charge in a good way. This type of analysis, to observe more general results, was made on several trips, going to compare the estimated results with the real and measured ones, as can be seen in the next sub-paragraph.

3.7.2 ANALYSIS ON A TEST SET

To see the effects of the algorithm in the SOC estimation problem, it was interesting to check its performances on 22 different drive cycles. The target of this analysis was to evaluate only the estimated SOC results, so both power and energy were forgotten. Firstly, the MAE and RMSE accuracy values were computed for each single drive cycle and the results are showed in Figure 73. Obviously, the data used to obtain these accuracy values were: the real state of charge and the estimated one. How it can be seen from this figure, all the values (both for RMSE and MAE) are lower than 1, and the main are around

0.2; this is an indicator of the good behavior of the algorithm. The only exceptions correspond to the drive cycles, where the estimation of the battery power is far from the real measured one.

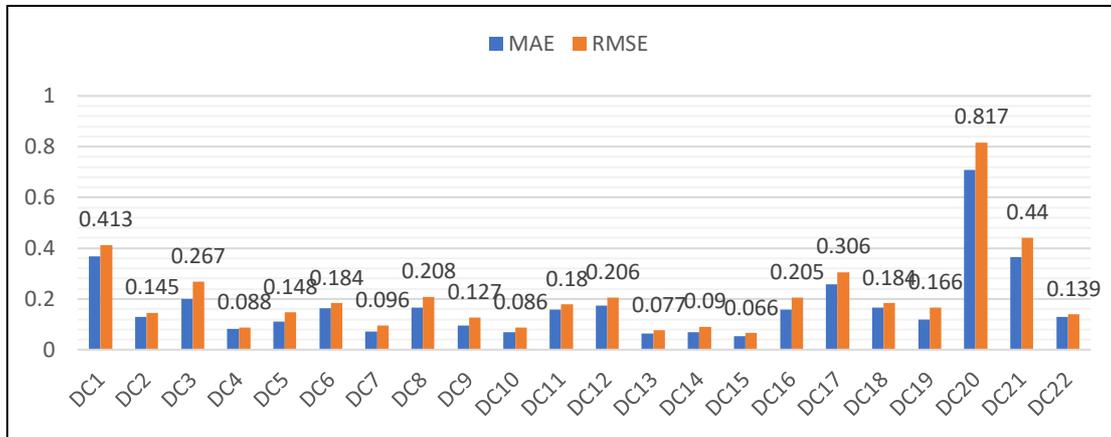


Figure 73. Metrics performances for estimation of SOC consumption, related to all the 22 drive cycles, used for testing.

It was interesting also to make a bit of analysis for the absolute error computed between the real SOC and the estimated one, $|e| = |\widehat{SOC} - SOC|$.

In Figure 74, the evolution of the absolute error over the time, for all the test routes, is shown. The global trend of the error is to increase over the time, and this phenomena was expected, due to the presence of an integration function in the algorithm.

The integrator introduces a cumulative error, that is intrinsic, in physical sense, in the solution found and cannot be removed.

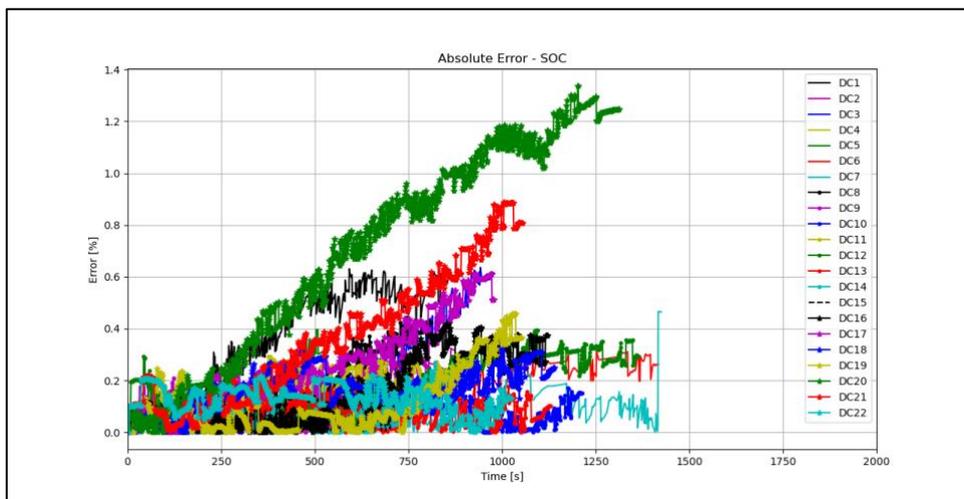


Figure 74. Absolute error relatives to the SOC computation over the time, for all the 22 drive cycles used for testing.

In addition, for each instant of time, the mean and standard deviation values relative to the absolute SOC error of all drive cycle tests used, were computed. In this way it was possible to obtain a general view of the behavior of the mean error (+/- the standard deviation) committed by the algorithm in SOC estimation, over the time. How it can be seen from Figure 75, the mean error tends to increase over time, and the same happens for the standard deviation. This is an effect of the integration function implemented in the algorithm and, as said many time, it cannot be solved in any way, unless of changing the resolving approach and so, by changing the model.

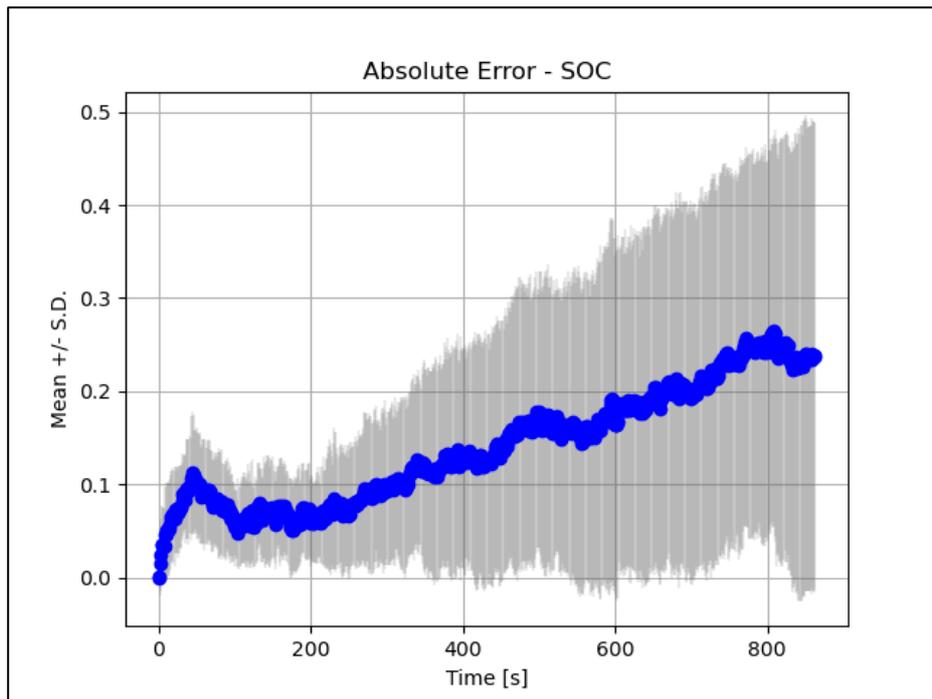


Figure 75. Plot with mean (blue) and standard deviation (gray) values relative to the absolute SOC error over time, computed considering all the 22 routes for testing.

After obtained these results, it was computed a vector containing the maximum absolute error, committed by each route used for testing in the total period of time, and another one, for the final absolute error value. From these two arrays, the mean value and the standard deviation were obtained, Table 7. The chart with the derived results is showed in Figure 76.

	Final absolute error	Maximum absolute error
Mean [%]	0.369	0.436
Standard deviation (S.D.)	0.266	0.261

Table 7. Resume of mean and standard deviation values relative to final and maximum absolute error value for SOC, computed considering the 22 routes for testing.

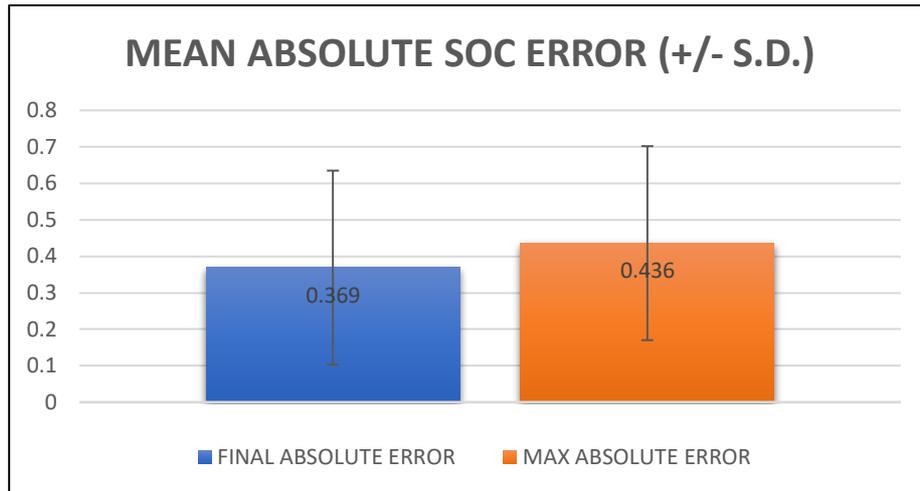


Figure 76. Mean value with standard deviation, related to the final and maximum absolute error value, associated to the SOC quantity, computed considering the 22 routes for testing.

3.8 TEST OF THE MODEL-BASED ALGORITHM IN A REAL SCENARIO

This last paragraph is dedicated to the simulation, in a real application scenario, of the algorithm designed to compute a range estimation for a battery electric vehicle.

Thus, all the data used, as inputs of the algorithm, are obtained starting from coordinates' points of route selected waypoints. To obtain these data are used the functions explained in the previous paragraphs of these chapter. To conclude, the output results, returned by the model, are compared with the real measured data.

The process starts with the extraction of a route from the databased containing all the data coming from the vehicle measurements; then, it is reproduced the same path in the OSRM web-service, in order to obtain the waypoints coordinates to feed to the software. The journey, extracted and analyzed, is indicated with the name of 'trip A', and starts from the Bylogix headquarter, placed at Strada Comunale del Portone, Grugliasco (indicated with label A) and ends to via Corrado Corradini, Torino (label B), as showed in Figure 77. The total route length is about 22 Km, and it is mainly made by tertiary, primary and motorway roads, so by national road and highways, while only for a short part of the route by city streets, as can be seen in Figure 78. To complete the trip, the estimated time, calculated by the algorithm, is approximately 17 minutes, while from measured data the duration is about 15 minutes. These differences are due to several approximation made by the algorithm, one among all, the predicted velocity profile that is different with respect the measured one. The elevation profile evidences that altitude range, computed as difference between the maximum and the minimum elevation value,

is about 58 m. The general trend shows how the route is characterized, in the first stretch of road, by a downhill, so it is expected to not spend a lot of energy there. Figures 70, 80 show the altitude profile and the slope road profile, as function of the distance. Also, the weather data was downloaded using the OpenWeatherMap API service. A recap of the information about the trip A is listed below, in Table 8.

TRIP A	Information
Total distance	22.4 Km
Total duration time	00:17:15 hh:mm:ss
Maximum elevation	276 m
Minimum elevation	218 m
Max legal velocity	130 Km/h
Air Temperature	23 °C
Weather	Good weather - Sun
Wind	1.03 m/s - from NE
SOC(t_0)	90 %
$C_{nominal}$	75 kWh

Table 8. Information about the vehicle, the weather, and the selected route, for Trip A.

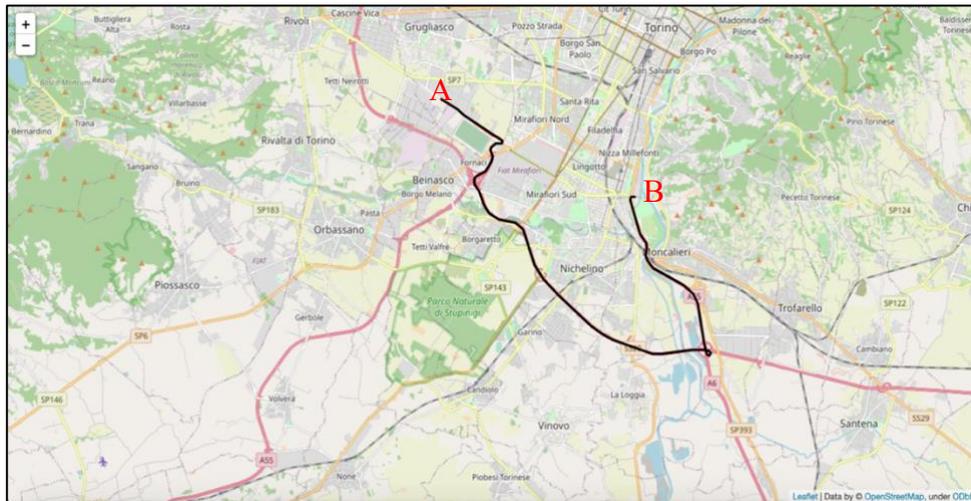


Figure 77. Route selected for testing test, from Bylogix S.r.l HQ, Grugliasco (A) to via Corrado Corradini, Torino (B).

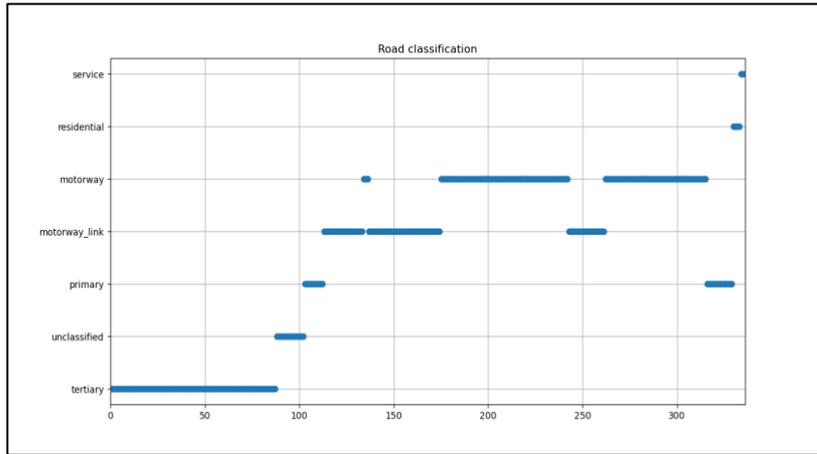


Figure 78. Road classification extracted from OpenStreetMap, relative to Trip A. [4]

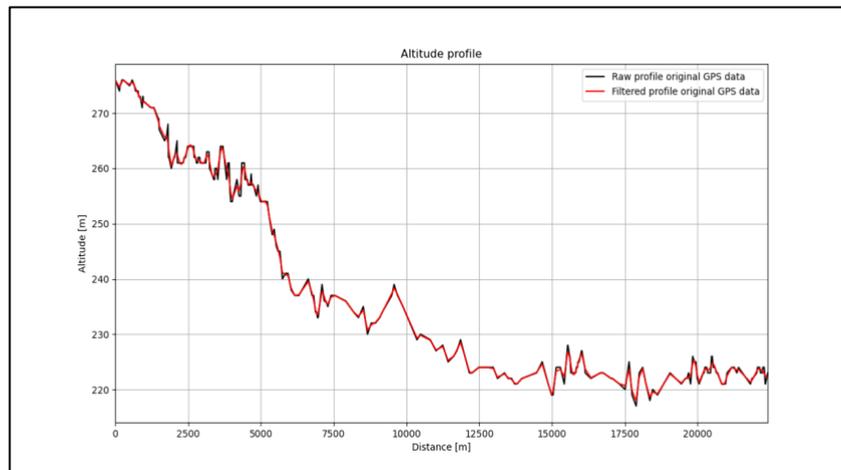


Figure 79. Elevation profile of the selected route, Trip A.

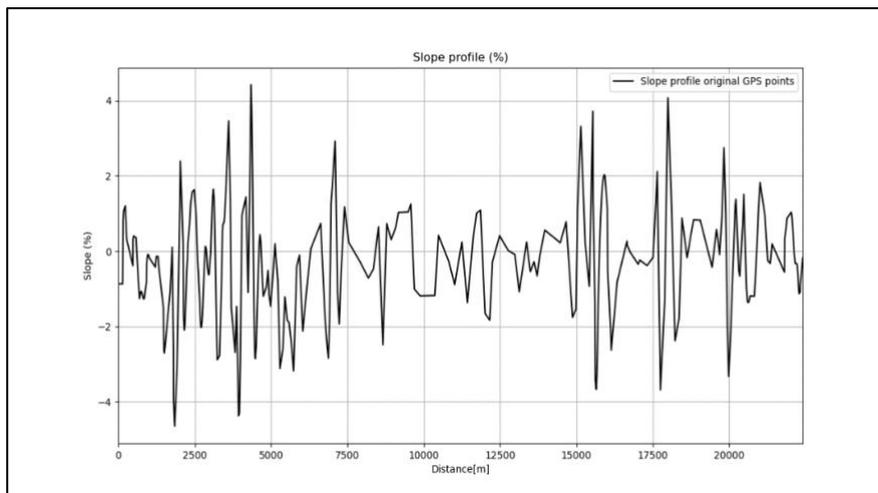


Figure 80. Slope profile of the selected route, in [%], relatives to Trip A.

For the computation of the output results, it is necessary to predict the velocity profile, and as said in the previous paragraphs, it is necessary to extract several road information. Using the data provided by OSM, it is possible to get these.

Using these information, and, also, by the knowledge of the speed limit signs, the legal velocity profile is computed. In Figure 81, it is showed a comparison between the predicted speed profile, returned by the algorithm, and the real velocity measurements. Obviously, the estimated and the real speed profile do not match perfectly, since the prediction of the velocity depends by several factors, such as the driving style of the driver, that is not possible to know a priori. Moreover, it can be observed, from the general trend of the velocity profile measured, that the maximum velocity reached was about 160 Km/h in highway, where the limit was 130 Km/h, but also that the real velocity was higher than the speed limits imposed by road classification.

For these reasons, it is expected an error in the estimation of the SOC consumption, with respect to the measured one. Note that, the Bias Velocity Factor (BVF) is imposed equal to 0.90, but if this factor is increased also the estimation of the SOC consumption moves closer to the measured one. However, to give some other information, also the acceleration profile is derived from the velocity one, Figure 82.

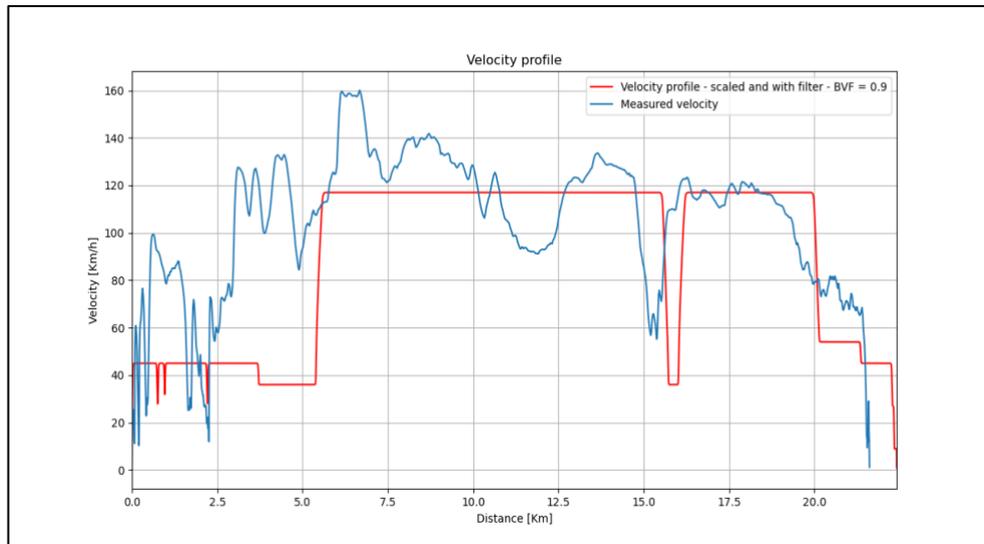


Figure 81. Velocity profile predicted by the algorithm with BVF=0.90 (red), compared with the real velocity profile measured (blue).

After having obtained the predicted legal velocity profile and all the other data extracted from internet, needed by the algorithm to work, it is possible to estimate the total traction power computed at different stages: wheels, electric motor, and battery. Figure 83 shows a comparison of these three power consumptions estimation, that are computed according to the equations 3.17, 3.24, 3.27.

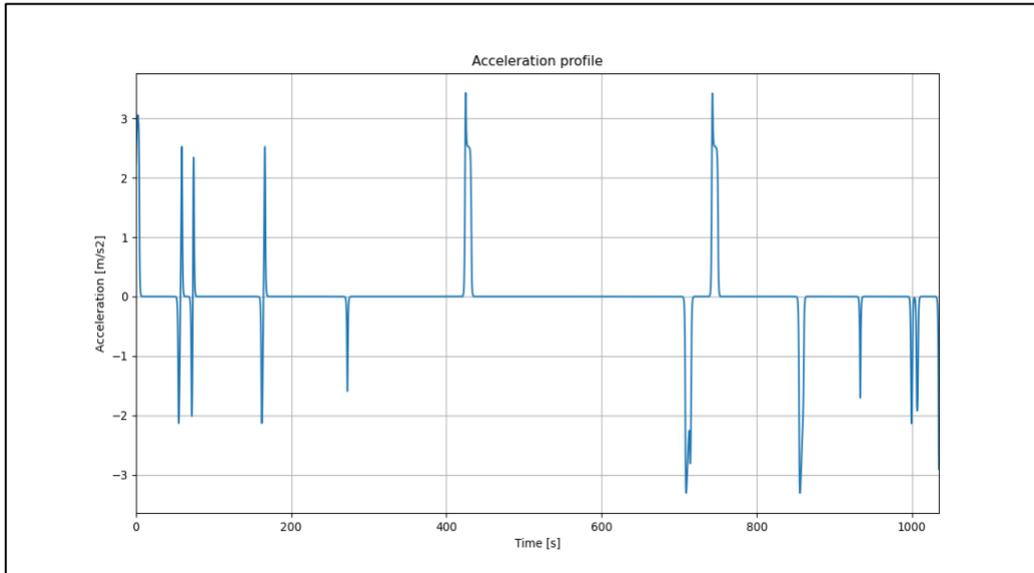


Figure 82. Acceleration profile derived from the predicted velocity profile.

As expected, it can be noticed that the estimated battery power is the highest among the other ones, if considered positive power values; instead, it is the lowest if considered negative values.

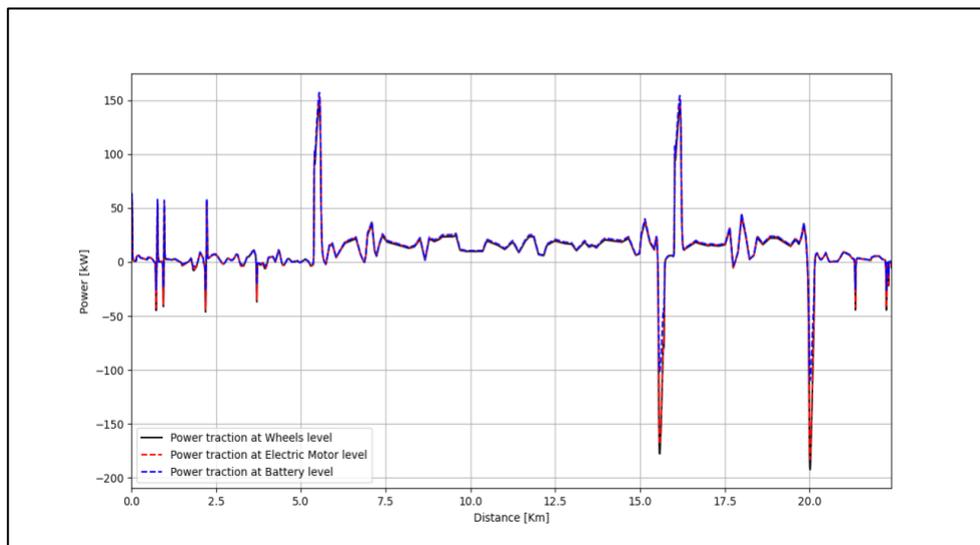


Figure 83. Power consumption at different vehicle's systems.

After having computed the estimation of total power supplied and recharged by the battery system, it was possible to compute an estimation of the energy consumption, using the equation 3.35

From Figure 84, it is possible to make some considerations about the energy consumption associated to the trip A.

For example, there are some sudden steps around Km 5 or Km 16, that happens in correspondence of rapid change of velocity; these could be removed going to improve the function designed to predict the velocity profile.

However, since the prediction of the velocity is not an easy task, this drawback can be acceptable due to the complexity of the problem.

At this point, the state of charge estimation is obtained using the equation 3.37, as function of the energy consumption previously estimated, of the $SOC(t_0)$ and $C_{nominal}$. Given a $SOC(t_0) = 90\%$ and a $C_{nominal} = 75$ [kWh], the SOC consumption is computed and compared with respect to the measured one, as displayed in Figure 85. [50]

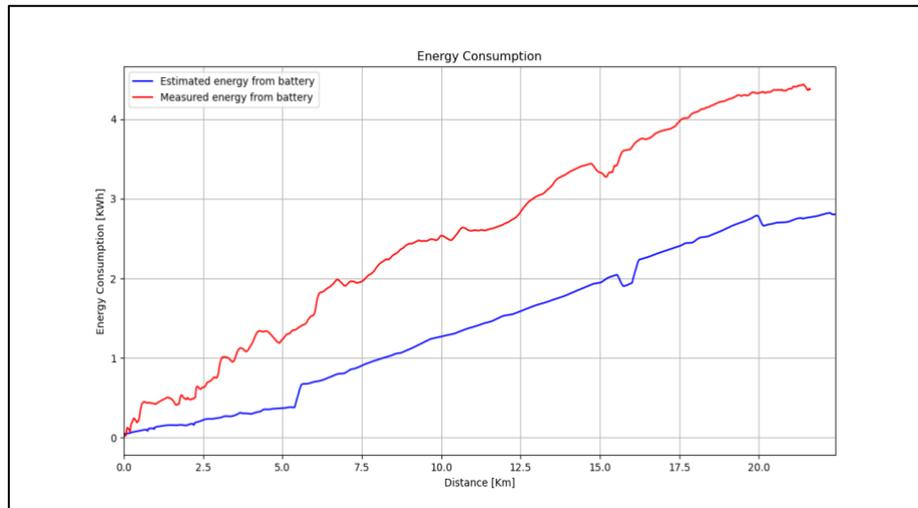


Figure 84. Energy consumption measured (red) and energy consumption estimated (blue) from battery power, relative to the trip A. BVF = 0.90.

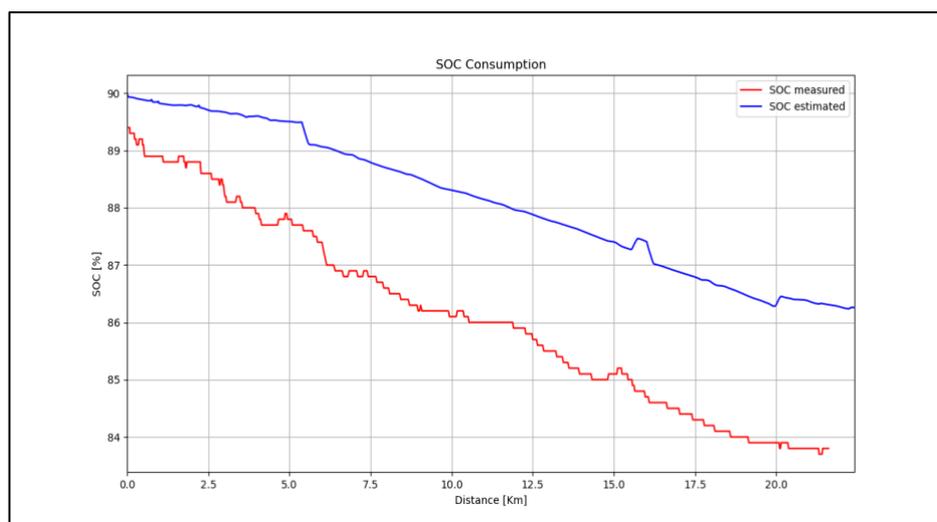


Figure 85. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. $SOC(t_0) = 90$, BVF = 0.90.

Analyzing the red line and the blue one in Figure 84, and 85, it is possible to observe that both follow the same trend, with some obviously errors. Focusing for example in SOC consumption plot, it is possible to clarify why the real and the estimated SOC behavior are different. The reasons why this happens are several, the first one is the assumption on the initial SOC value. In fact, the algorithm settings are set up to have a $SOC(t_0) = 90$, but this is not always true: if the real initial value of state of charge could be measured and used in the equation 3.37, the general trend of the estimated SOC consumption is different, closer to real case at least in the first part, but nevertheless, the general behavior does not change a lot, Figure 86. However, changing the initial SOC values nothing changes in terms of model accuracy, since the only effect is a translation of the estimated curve.

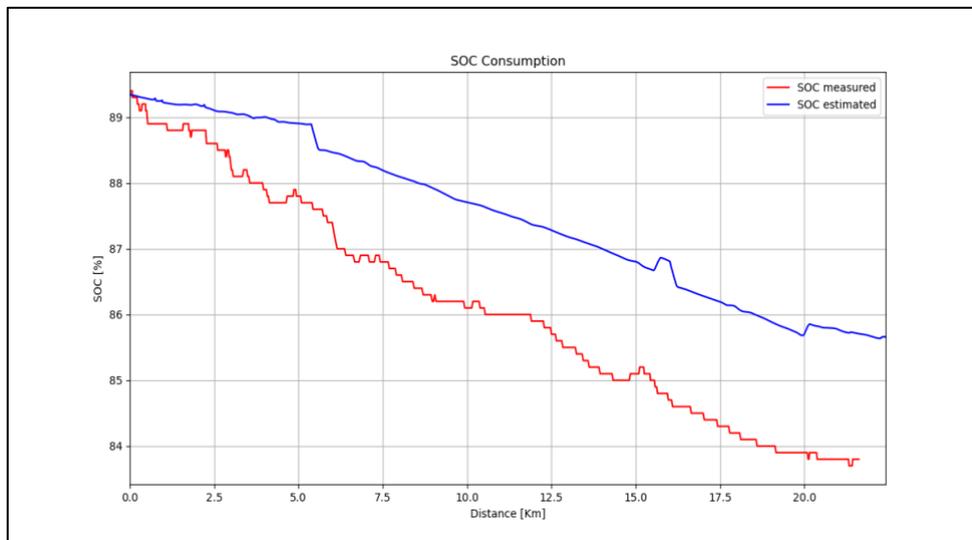


Figure 86. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. $SOC(t_0) = 89.40$, $BVF = 0.90$.

The second fact is the integrator used to compute the energy from the power consumption; it will introduce a systematic error to the problem.

The third factor, that also have a huge influence, is the velocity profile.

In general, less is the velocity of the vehicle, lower will be the consumption, and vice versa for high speed. From a review of the Figure 81, it can be realized that the speed profile predicted tends to be lower than the measured ones, so obviously the consumption estimated will be lower than the real one.

A proof of this fact can be realized going to increase the BVF factor from 0.90 to 1.02, obtaining the new velocity profile, Figure 87. Nevertheless, going to increase the BVF, the speed limit will not be considered yet, so this example is done only to demonstrate that the results are better with respect to the previous case, with BVF equal to 0.90. In real

application scenario, the software respects all the limits imposed by the Italian Law on road safety. [36]

Analyzing the data results, it is possible to evidence that the absolute error, both in terms of energy consumption and of SOC consumption, is reduced compared to the case with $BVF=0.90$, as can be saw in Table 8. The new energy and state of charge consumption's plots are showed in Figure 88 and Figure 89.

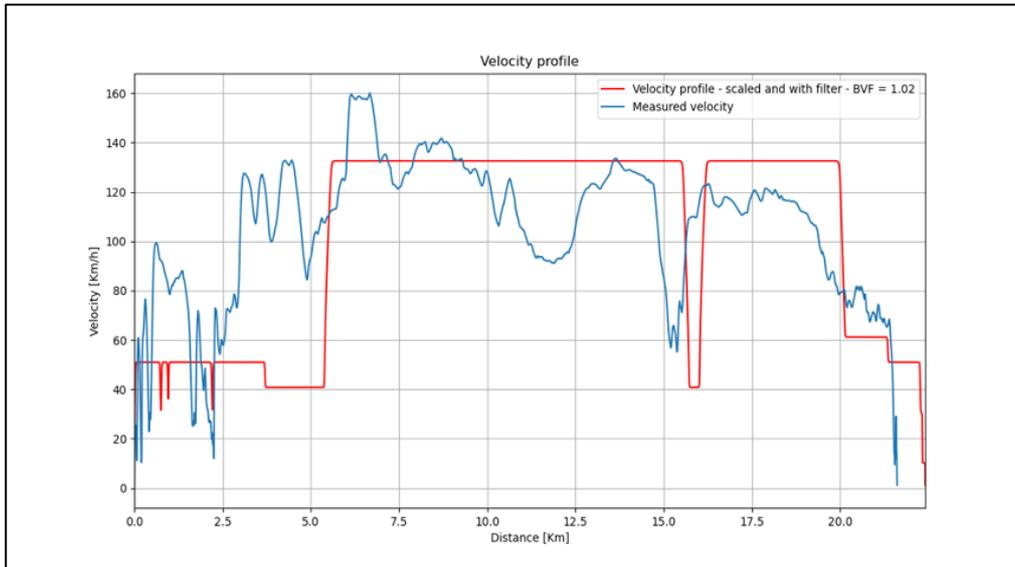


Figure 87. Velocity profile predicted by the algorithm with $BVF=1.02$.

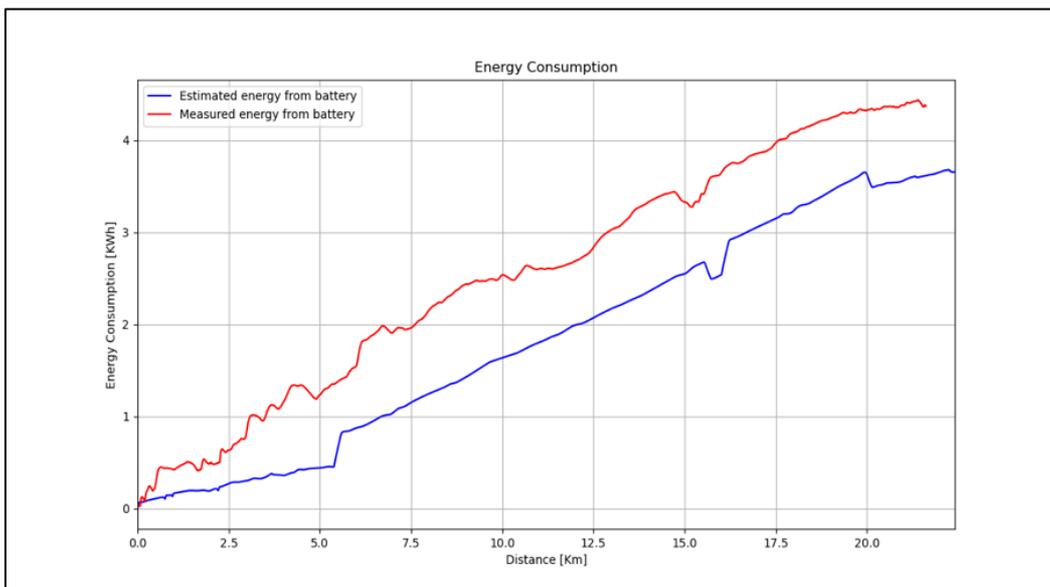


Figure 88. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. $SOC(t_0) = 90$, $BVF = 1.02$.

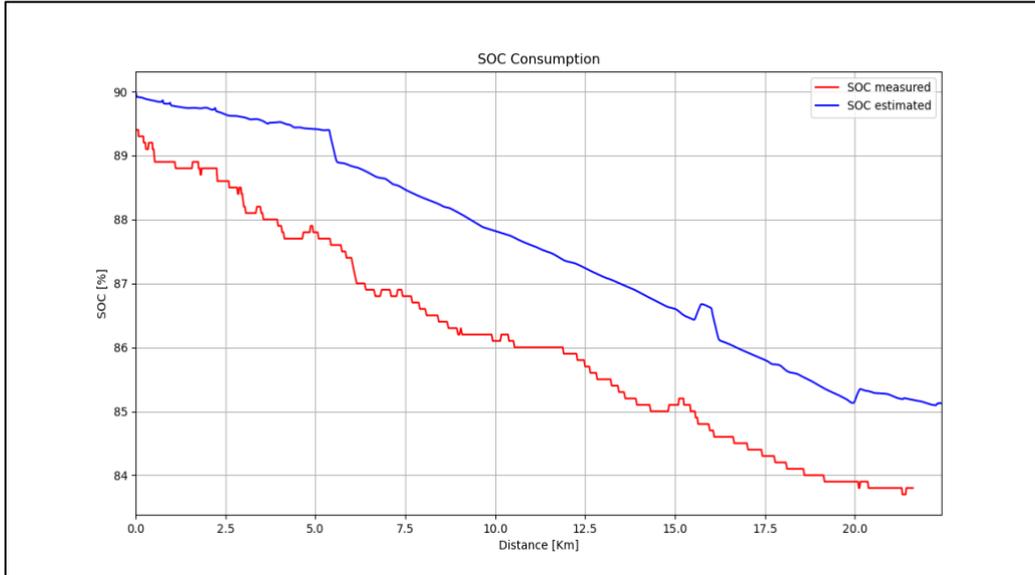


Figure 89. Comparison of the measured SOC consumption (red) and of the estimated one (blue), relative to the trip A. $SOC(t_0) = 89.40$, $BVF = 1.02$.

TRIP A	Real Measurements	Estimate Measurements
CASE 1 WITH BVF = 0.90		
Energy consumption	20.23 [kWh/100Km]	12.52 [kWh/100Km]
Final energy consumption value	4.37 [kWh]	2.80 [kWh]
SOC final value	83.80 [%]	86.25 [%]
$ \Delta E $	1.57 [kWh]	
$ \Delta SOC $	2.45 [%]	
CASE 1 WITH BVF = 1.02		
Energy consumption	20.23 [kWh/100Km]	16.32 [kWh/100Km]
Final energy consumption value	4.37 [kWh]	3.66 [kWh]
SOC final value	83.80 [%]	84.52 [%]
$ \Delta E $	0.71 [kWh]	
$ \Delta SOC $	1.28 [%]	

Table 8. Recap of the energy and SOC consumption values associated to the trip A. $SOC(t_0) = 90$.

The main reasons, causing these differences in values, are due to the difficulty to have a perfect prediction of the velocity profile and, moreover, to the unmatching between the real speed profile measured and the predicted one.

This last problem is owing to the hard driving style of the driver, characterized by a lot of sudden and high accelerations and by elevated velocity, and so major consumptions. To conclude, a comparison between different energy and state of charge consumptions, obtained by using different BVF: 0.90 (blue), 1 (black), 1.02 (purple) and 1.10 (yellow), is done to evaluate the effect of the growth of the velocity, as displayed in Figure 90, 91, 92. From Table 9, it is possible to observe that increasing the BVF, and so the velocity profile, the estimated values, and the real ones, both for energy and state of charge consumption, are closer.

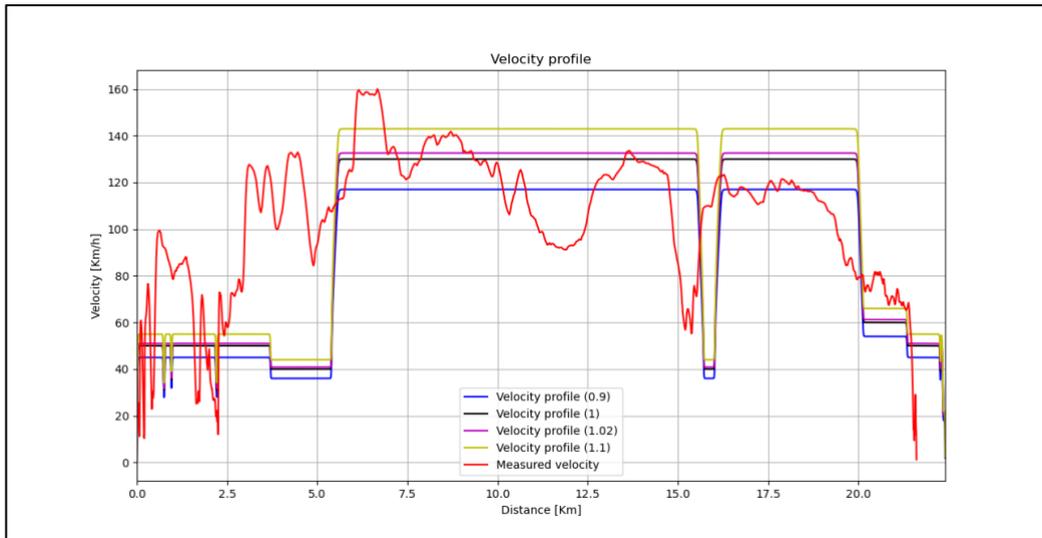


Figure 90. Comparison of different velocity profiles, according to different BVF, with respect the measured velocity (red line), relative to trip A.

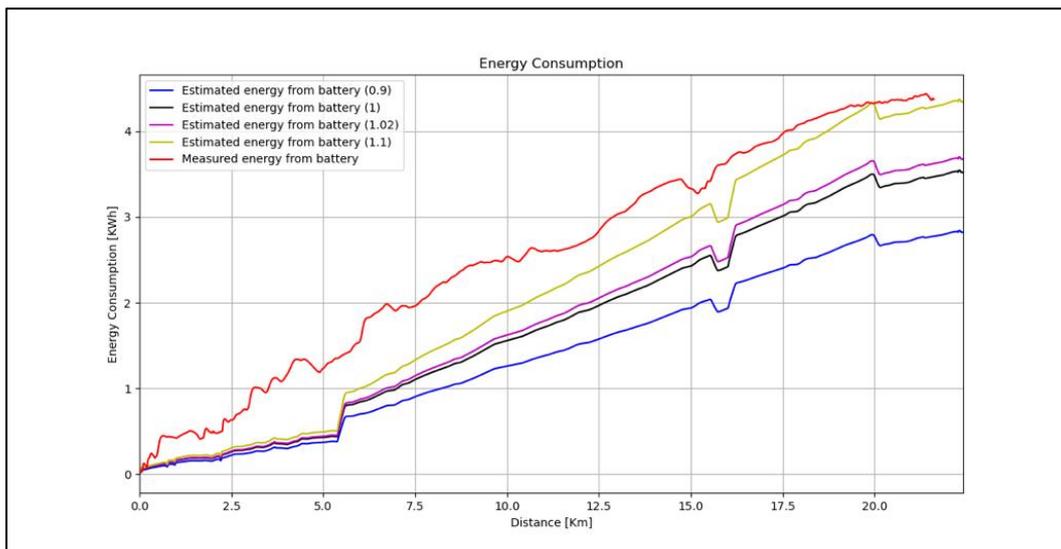


Figure 91. Comparison of different energy consumption trend, associated to different BVF, with respect the measured one (red line), relative to trip A.

From the analysis on Figure 92, and by the observation of the results in Table 9, it is possible to observe a discrepancy between the real energy consumption and the estimated ones, at different BVFs; in particular, a huge error there is in the first 5 kilometers. This is caused by a real velocity profile, characterized by high values, far from the ones imposed by legal speed limit of the Italian law, the same used by the algorithm.

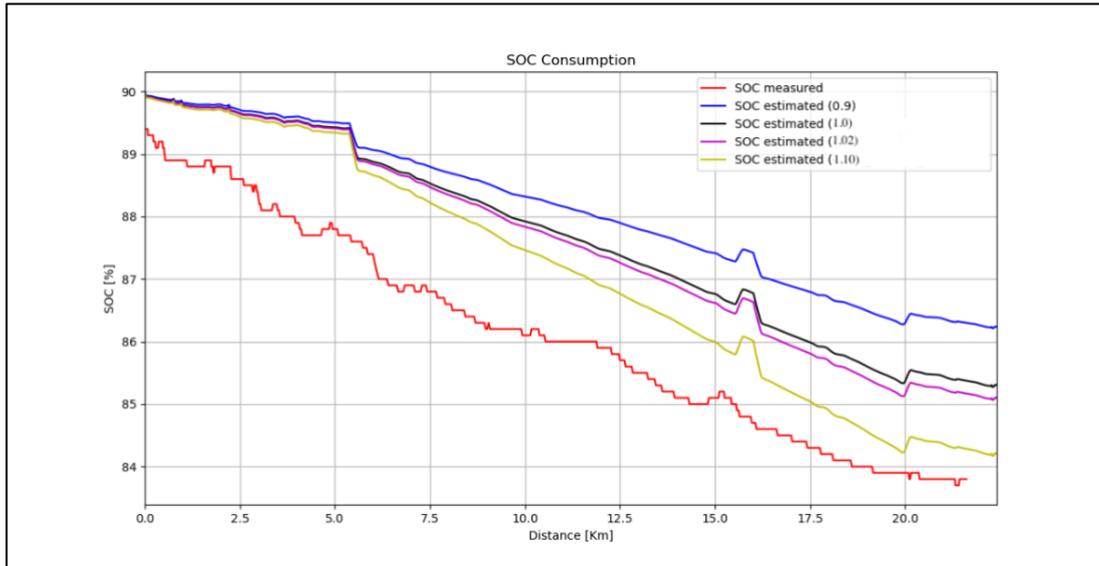


Figure 92. Comparison of different SOC consumption levels, associated to different BVF, with respect the measured one (red line), relative to trip A.

	Energy cons. [kWh/100Km]	FEC [kWh]	SOC final [%]	\Delta E [kWh]	\Delta SOC [%]
CASE 1 WITH BVF = 0.90					
Real Measurement	20.23	4.37	83.80	1.57	2.45
Estimated measurement	12.62	2.82	86.22		
CASE 2 WITH BVF = 1					
Real Measurement	20.23	4.37	83.80	0.84	1.91
Estimated measurement	15.75	3.53	85.29		
CASE 3 WITH BVF = 1.02					
Real Measurement	20.23	4.37	83.80	0.71	1.28
Estimated measurement	16.45	3.69	85.08		
CASE 4 WITH BVF = 1.10					
Real Measurement	20.23	4.37	83.80	0.02	0.39
Estimated measurement	19.45	4.35	84.19		

Table 9. Table of comparison of the consumption cases, associated to different BVF values, relative to trip A. Column named 'FEC' means Final Energy consumption value. SOC(t0) = 90.

3.9 CONCLUSIONS

The designed model-based algorithm, obtained following an approach based on the knowledge of the physical laws that regulate the power and therefore the energy consumption of an electric vehicle, returned good results. From the analysis on the RMSE computed on the SOC estimated values, from the drive cycles test set, can be seen that the general trend is to have value, around 0.1, that is very fine.

Even the absolute error computed, in terms of different between real and measured SOC, returns small values. These two facts are a proof that the algorithm works well.

A comment on the implementation of the algorithm in a real scenario is necessary to be done. How can be observed from the result in paragraph 3.8, the main drawback is associated to the function designed for the generation of the predicted velocity profile. In fact, the speed is among the inputs that have a majorly influence in the derivation of the power battery consumption of an electric vehicle and so, it influences a lot the estimation of state of charge consumption.

Therefore, among the future developments on this algorithm, there could be the design of a new function that better predicts the vehicle velocity profile, taking into consideration other factors than those considered here, such as: the presence of traffic, roundabouts, stop signals and traffic lights.

Other studies can be done in the analysis of the different electric vehicle systems (electric motor, battery and so on) that here were not considered; making this it could be reached better results in the offline estimation of the state of charge consumption.

A MACHINE LEARNING TECHNIQUE FOR SOC ESTIMATION

In the next paragraphs of this chapter, it will be discussed which algorithms, belonging to the category of Machine Learning methods, but in particular to the Artificial Neural Network group, could be used for the purpose of this thesis, therefore for the estimation of the SOC associated to a fully electric vehicle. Consequently, greater attention will be placed on the development of a regressive neural network, leaving out those used for classification or for other purposes. For this work's section, a large number of data were used, about 120 thousand, all provided by the company Bylogix S.r.l and acquired directly from a Tesla Model 3 with a 75 kWh battery pack.

The classical approach used by the researchers and engineers, in this particular field, is based on the usage as inputs of: voltage, current and temperature; all these are battery values normally used to estimate the SOC. Nevertheless, the developed model was designed in a totally different way. The variables used as inputs were: the velocity, the acceleration, and the road slope angle profile, while the output of the network was the battery power. Once the estimation of this quantity was obtained, by integrating it, the energy provided by the battery was computed, and so the SOC could be derived.

All these functions were implemented in the so-called ANN algorithm, whose design schema is illustrated in Figure 93. The main reason behind the choice of these input data was associated to the fact that the algorithm had to work offline, so the input data had to be predicted easily before the trip began and in addition, had to be related to the driving condition.

A brief overview about what the paragraphs describe in the chapter is listed here.

In the first paragraph there is an overview on how the machine learning world has changed the manner of approaching to some regression problems. In paragraph 4.2, a description about the neural networks' theory is done, focusing on the neuron element and its function on the network behavior, the main architectures, the learning process, the overfitting and underfitting problems and to conclude, a list of the main activation functions is provided.

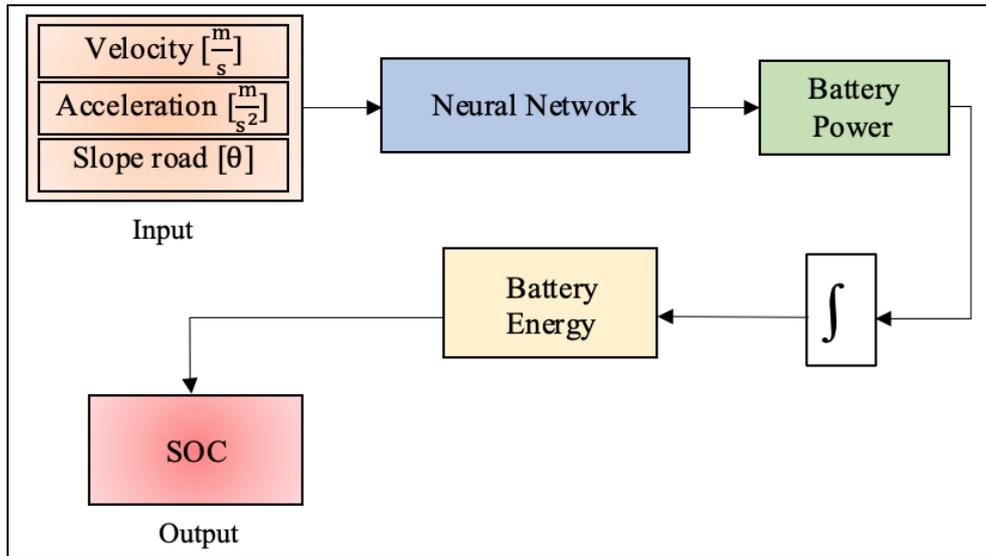


Figure 93. Design schema of the ANN algorithm to SOC estimation.

In the paragraph 4.3 a description on the neural network designed to solve the SOC estimation problem is provided, Figure 93.

Paragraph 4.4 presents an analysis of the results obtained using the developed algorithm. The chapter ends with the conclusions.

4.1 MACHINE LEARNING, A NEW FRONTIER FOR THE ESTIMATION PROBLEM

The digitization of processes has been achievable thanks to the exponential increase of the data available to engineers and scientists, to supervise and analyze processes and systems. If the amount of such data is very large, then it will talk of Big Data.

In addition to the growth of available number of data, new powerful Graphic Processing Unit (GPU) and enhanced algorithms have led to the development of machine learning (ML) methods, which today are among the main used tools to rely on for estimation processes, for the forecasting of future values or, in general, for the system identification problems. [56]

When there are problems that try to explain the input-output relationship, it can be applied three main methods: the first is a model-based, the second a black-box, the third is a combination of these two, said grey-box. The first type is the one analyzed in chapter 3; to the second method, instead, belong all those models where, to establish the input-output relationship, are used only the measured input and output data and are not taken in consideration the physical knowledges of the system under investigation. Machine

learning algorithms, among which it can be found the artificial neural networks, belong to this category. In many cases, the deep learning (DL) term is mistaken with the artificial neural network one, but to be clear, the first one is a particular architecture's form, that is deeper and more complex than the neural network standard one.

Talking about machine learning, could be easy to drop in the wrong comparison between machine learning and artificial intelligence.

By definition, Artificial Intelligence usually refers to the context of developing computers, hardware or software, able to emulate human thought and perform tasks in real-world environment, by using data that give them the possibility to automatically trigger actions without the human interface. [57]

Instead, Machine Learning is a subcategory of the AI, and it uses algorithms that automatically learn from data and adapt the reactions to the input data they receive. [57]

Inside the ML learning algorithm, there are the Artificial Neural Networks (ANN), that without going into the details, are models designed with the idea of being able to autonomously capture the relationships, especially non-linear, of also complex systems. They use only the input x and the output y data, as entry of the model, and provide, as output, an estimation \hat{y} of the true output y , without asking the physical explanation of the problem, Figure 94.

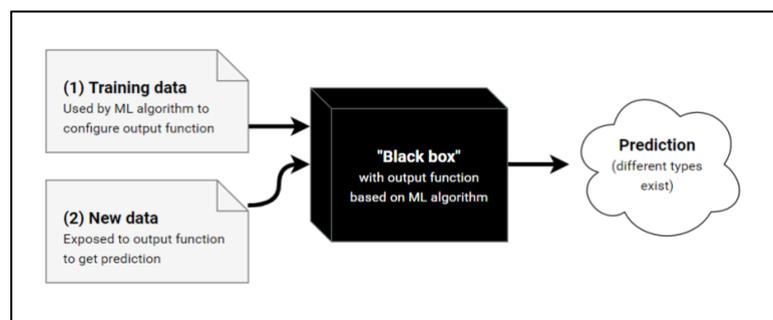


Figure 94. Black-box model. [58]

The approach, or better the idea, behind the development of machine learning (ML) algorithms has radically changed with respect to the canonical modus operandi of the classic software programmer. While the latter is used to developing a software, a program, or even a function, looking for some mathematical rules that allow to associate the inputs with the outputs, the ML programmer uses only the input and output data to train an algorithm, which autonomously learns the relationship sought, and returns, as output, the rules, that explain it, Figure 95. [59]

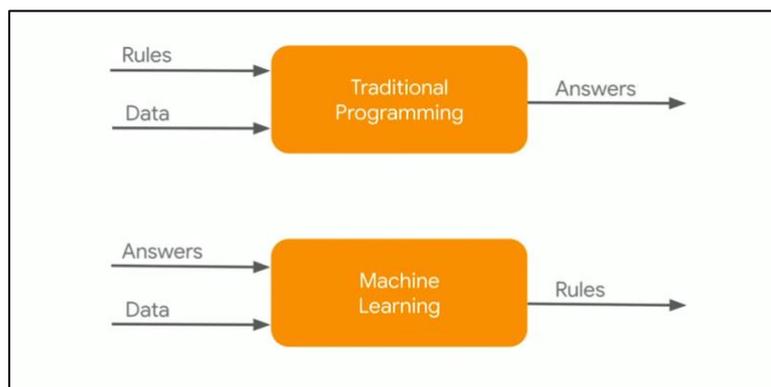


Figure 95. Machine learning and classical programming differences. [59]

Nevertheless, the neural network is not beautiful as it can be appeared, and several drawbacks are hidden behind this world. For example, to manage a huge number of data, or to create a very complex network, are necessary very powerful computers, equipped with expensive and high-performance GPU, in order to reduce the computational time. In addition, the science of the deep learning is far from the classical mathematics and science, where everything can be explained and demonstrated; here there is not yet a mathematical formalism. All the common rules, that can be found on internet, are obtained by experiments and so, it much more closer to trial-and-error approach. [56]

Among the negative aspects of these models, they require a large amount of data, and not always it is possible to have these amount of information; in fact, in general, the more data you have, the better the learning will be and, therefore, the final output provided by the algorithm. In addition, current ANN architectures do not have the possibility to track statistical changes of the training data in real time. [56]

However, in many application, the results found by the neural network algorithms are impressive and so, all the negative sides are overcome by these brilliant results.

Machine learning algorithms can be divided into four categories:

- **Supervised Learning:** in this case the algorithm developed return a set of rules starting from the knowledge of the input data (the training set) and of the true output (label data). The rules found will be then tested and used for totally new dataset. This category is applied for classification and regression problems. [56]
- **Unsupervised Learning:** in this case the labels are not provided to the model in the training phase. The model can be used to recognize patterns and in general is more complex than the Supervised typology. Typical applications are: detection of anomalies in patterns of data, clustering, split of the data into groups based on their similarities. [54]
- **Self-Supervised Learning:** with respect to the Supervised case, here the output data (labels) can be generated directly from the input data, using, for example, a heuristic algorithm. [56]

- Reinforcement Learning: the train task of this algorithm is to take a series of decisions that are dependent with each other to reach a certain goal, into a specific and, also, complex environment. To achieve the goal, the algorithm uses a system of either reward or penalties for the action it performs. The goal is to maximize the total award. [56], [60]

In this work it will be used the Supervised Learning, since the goal is to solve a regression problem, that is the estimation of the SOC.

4.2 THEORY ABOUT NEURAL NETWORKS

In this paragraph, the theory about the neural networks, starting with the mathematical description of what the neuron is, will be illustrated. Then a briefly description on the main architectures characterizing the neural networks (with a focus on the feed-forward typology), the mathematical description of the learning algorithms, the exposition of the main activation functions and the metrics that can be used to analyze the network is done.

4.2.1 THE NEURON

The Artificial Neural Network (ANN) is an example of computational model, that belongs to the autonomous learning field, characterized by artificial neurons. Its name derives by the human brain model, that is made by several biological neurons, that are fully connected one with each other, and are able to learn, by exchanging information, and adapt their behavior to new input stimuli. The learning and adaptation operations are performed by themselves, autonomously. The neural network can be used for solving several problems, as classification, prediction, estimation, and categorization. [26]

The core of the neural network is the node, also called neuron, that represents the equivalent mathematical model of the biological neuron, Figure 96. [26]

Each node is connected to the others using weights. The first model of artificial neuron was implemented by McCulloch and Pitts, in 1943.

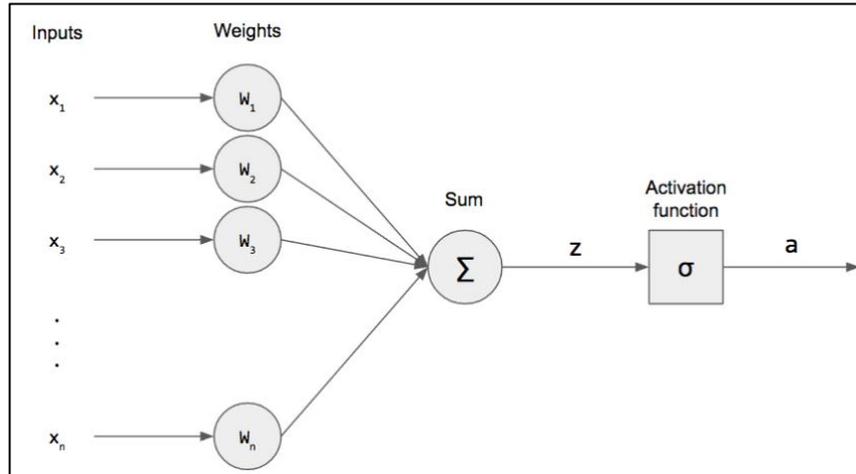


Figure 96. Function implemented in a single artificial neuron. [61]

Considering a model with only one layer, containing a single node and n inputs, as in Figure 96, the general functions implemented in that node are showed in equations 4.1 and 4.2. [26]

$$z = \sum_{i=1}^n (x_i * w_i) + \beta \quad (4.1)$$

$$a = \sigma(z) \quad (4.2)$$

where:

- x_i , is i -th node's input.
- w_i , is i -th node's weight.
- β , is the node's bias, a constant term (in Figure 96 it is 0).
- z , is the pre-activation function of the node (that is a number).
- σ , is the activation function of the node.
- a , is the output of the node.

Supposing to have n inputs, placed inside a row vector $\mathbf{X} = [x_1, \dots, x_n] \in \mathbb{R}^{1 \times n}$, the number of weights associated to the single node are n and, can be stored in a row vector $\mathbf{W} = [w_1, \dots, w_n] \in \mathbb{R}^{1 \times n}$. [61]

The equations 4.1 can be written in a vectorized version, as showed by the following equation:

$$z = \sum_{i=1}^n (x_i * w_i) + \beta = \mathbf{X} \cdot \mathbf{W}^T + \beta \quad (4.3)$$

Another important aspect of the neuron is the activation function σ , which represents the crucial point of the neuron and that regulates its functional behavior and so the type of relationship that will be between the inputs and the output of the node.

In fact, the powerful of these networks is that the activation functions, being mostly non-linear, are able to capture any variation of the data simply by changing the nodes' weights and biases values during the learning period, according to the training data flow. [26]

More information about the activation functions will be given in sub-paragraph 4.2.8.

4.2.2 NEURAL NETWORK ARCHITECTURE

In the previous sub-paragraph was described a single neuron and how it works, in mathematical sense. In a real neural network model, there is a combination of the nodes that, putted together, make the layer. In standard ANN, can be distinguished three different main layers:

- Input layer, that contains only the input features of the model; in practical cases, it is not considered a layer like the other one (no neurons are present).
- Hidden layer.
- Output layer, that gives as output the prediction \hat{y} .

The number of the hidden layers can be equal or greater than one, and, in this last case, the network is defined: Deep Neural Network (DNN). The term Deep is associated to the presence of the multiple hidden layers and it is the standard choice for the modern networks.

The standard nomenclature associated to an ANN established that a network with ℓ layers has 1 output layer and $\ell - 1$ hidden layers. For example, a network with 2 layers, is made by one hidden layer and a single output layer, so the input layer is not considered in the count; however, this is not a rule, but only a convention.

It can be possible to distinguish two main type of neural network architectures, the Feed-Forward (FFNN) and the Recurrent (RNN), Figure 97.

The first one, is the classical type of network and the most used too. Here, the layers are fully connected and the connections are established from the back to the forward and not backwards, or with itself, connections are allowed. [62]

The recurrent neural networks instead are characterized by feedback connections. Since feedback connection are used, they are advice for problems characterized by temporal sequence of data (natural language problems, audio/video problems, forecasting of stocks' price and so on.) In fact, the main feature of these networks is that they contain a

memory effect, that could be short in time, but also long as in the case of LSTM (Long Short Term Memory) networks. Obviously, this second architecture is more complex than the feed-forward one and so, it needs more computational power to be solved. [62]

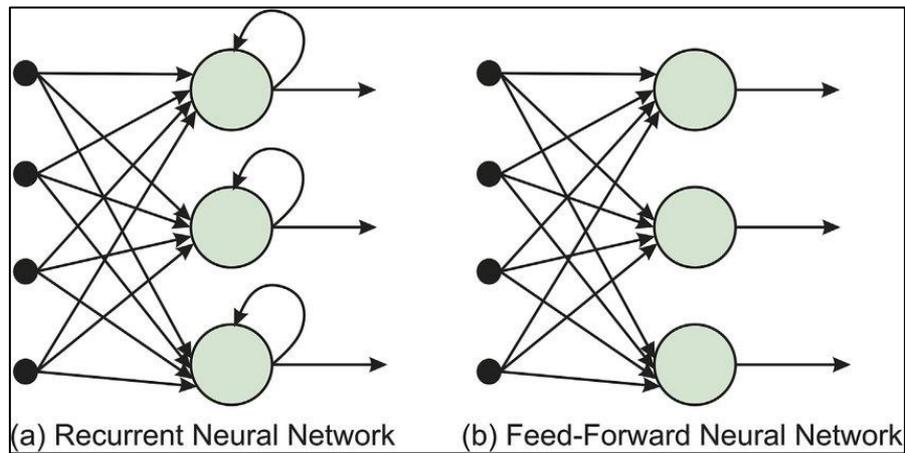


Figure 97. Recurrent Neural Network and Feed-Forward Neural Network.

However, since in the algorithm to implement it was choice to use the FFNN, the working principles and the mathematical formulations of these type of network will be described.

4.2.3 TRAINING WORKFLOW

In this sub-paragraph all the main mathematical formulations that characterize the feed-forward neural network models will be provided and evaluated.

In Figure 98, it is possible to have an overview of the main steps that regulate the training process. This is the classical workflow followed by the network to learn, from the given data, and adapt weights and biases values to return an estimated output as close as possible to the real one.

In the sub-paragraph 4.2.4 the forward step, whose goal is to find the output of each node of each layer is described. Then, in 4.2.5, the backward step, that includes the calculation of the loss function derivative, the backpropagation, and the weights update process, is described.

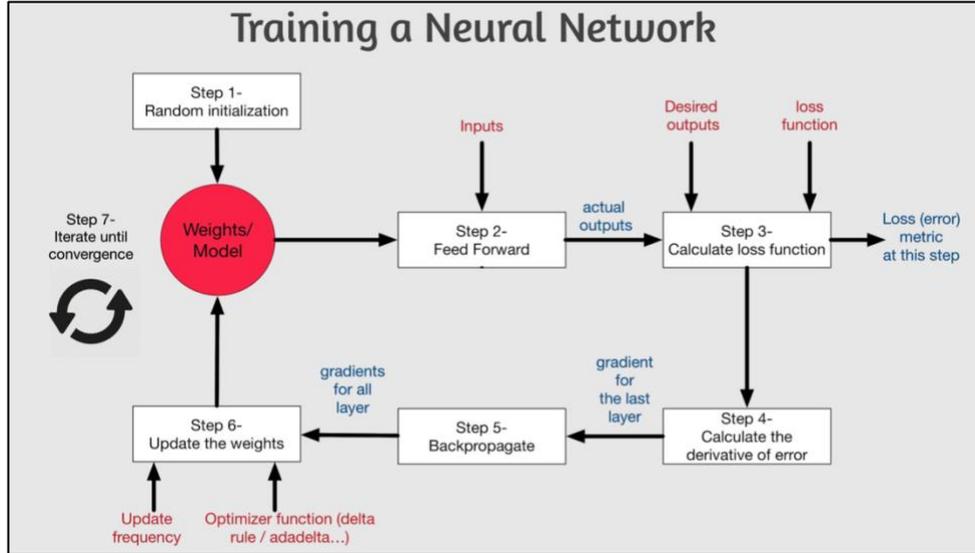


Figure 98. Workflow of the main step characterizing the training process in a neural network.

4.2.4 FEED-FORWARD STEP

The main calculus implemented in the neural network model to achieve the estimated output \hat{y} , starting from the input layer, are described here.

Since the aim of the developed algorithm is the estimation of the SOC, the network will be categorized as a regression problem. Being a regression problem, with only one output, the number of nodes present in the output layer will be always one. Instead, for each of the hidden layers, it can be possible to have g nodes.

In Figure 99, it can be observed a Feed-Forward Neural Network with only 2 layers: one hidden layer and one output layer (the input layer is not considered). Considering the hidden layer, the generic pre-activation function $z_j^{[1]}$ and the output $a_j^{[1]}$ of the generic hidden layer node j , can be obtained according to the following equations [12]:

$$z_j^{[1]} = \sum_{i=1}^3 x_i \cdot w_{ji}^{[1]} + \beta_j^{[1]} \quad (4.4)$$

$$a_j^{[1]} = \sigma_j^{[1]}(z_j^{[1]}) \quad (4.5)$$

where:

- $z_j^{[1]}$, is the pre-activation function of the hidden layer node j .
- x_i , is the input i of the hidden layer node.
- $w_{ji}^{[1]}$, is the weight between the input i and the hidden layer node j .
- $\beta_j^{[1]}$, is the bias term of the hidden layer node j .

- $a_j^{[1]}$, is the output of the hidden layer node j .
- $\sigma_j^{[1]}$, is the activation function of the hidden layer node j .

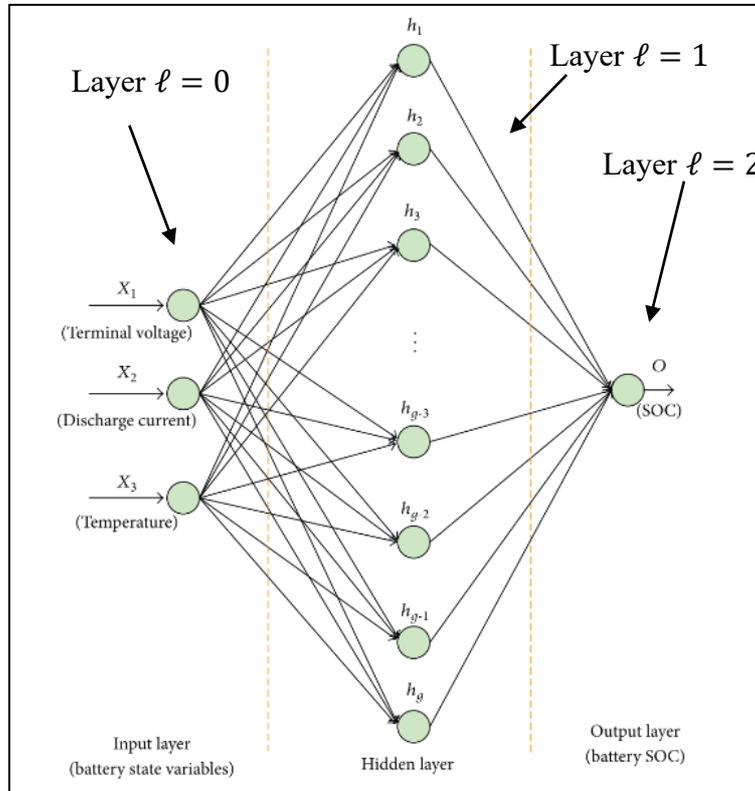


Figure 99. Example of ANN with 2 layers used to estimate the SOC. [12]

To follow a more general notation, x_i indicates the generic input i that goes forward in the hidden layer node j ; it can be also written as $a_i^{[0]}$, where: the i denotes the i -th input and the $[0]$ the input layer. After having computed all the outputs of the hidden layer neurons, it is possible to pass to the next layer, that in this case is the output one, and calculate the network output, following the equation below:

$$\hat{y} = a_1^{[2]} = \sigma_1^{[2]} \left(\sum_{i=1}^g a_i^{[1]} \cdot w_{1i}^{[2]} + \beta_1^{[2]} \right) \quad (4.6)$$

where:

- $a_i^{[1]}$, is the input i of the single output layer node ($j = 1$), and so, it is also the output of the hidden layer node i .
- $w_{1i}^{[2]}$, is the weight between the hidden layer node i and the output layer node.
- $\beta_1^{[2]}$, is the bias term of the output layer node.
- $a_1^{[2]}$, is the output of the output layer node.
- $\sigma_1^{[2]}$, is the activation function of the output layer node.

The equations 4.4, 4.5 can be written in vectorized version [63]. Supposing to have n_i input features and n_h hidden layer nodes, then those can be substituted by these new equations:

$$\mathbf{Z}^{[1]} = \sum_{i=1}^{n_i} x_i \cdot w_{ji}^{[1]} + \beta_j^{[1]} = \mathbf{X} \cdot (\mathbf{W}^{[1]})^T + \boldsymbol{\beta}^{[1]} \quad (4.7)$$

$$\mathbf{A}^{[1]} = \boldsymbol{\sigma}^{[1]}(\mathbf{Z}^{[1]}) \quad (4.8)$$

where:

- $\mathbf{X} = [x_1, \dots, x_{n_i}] \in \mathbb{R}^{1 \times n_i}$, is the input vector of features.
- $\mathbf{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & \dots & w_{n_h 1}^{[1]} \\ \vdots & \ddots & \vdots \\ w_{1 n_i}^{[1]} & \dots & w_{n_h n_i}^{[1]} \end{bmatrix} \in \mathbb{R}^{n_h \times n_i}$, is the weight matrix of the hidden layer.
- $\boldsymbol{\beta}^{[1]} = [\beta_1^{[1]}, \dots, \beta_{n_h}^{[1]}] \in \mathbb{R}^{1 \times n_h}$, is the bias vector of the hidden layer.
- $\mathbf{Z}^{[1]} = [z_1^{[1]}, \dots, z_{n_h}^{[1]}] \in \mathbb{R}^{1 \times n_h}$, is the pre-activation function vector of the hidden layer.
- $\boldsymbol{\sigma}^{[1]} = [\sigma_1^{[1]}, \dots, \sigma_{n_h}^{[1]}] \in \mathbb{R}^{1 \times n_h}$, is the activation function vector of the hidden layer.
- $\mathbf{A}^{[1]} = [a_1^{[1]}, \dots, a_{n_h}^{[1]}] \in \mathbb{R}^{1 \times n_h}$, is the output vector of the hidden layer.

The same can be done for the output layer, obtaining a similar result but with a weight matrix with different dimension and scalar elements, due to the presence of a single output layer node. The previous equations describe the mathematical formulation of the forward propagation process in the FFNN described in Figure 99.

From the previous example, more general results can be derived, starting from the notation used there. For example, $a_j^{[\ell]}$, is referred to the output of the node j of the layer ℓ ; $w_{ji}^{[\ell]}$, is the weight between the node j of the layer ℓ , and the node i of the layer $\ell - 1$. The first layer is indicated with $\ell = 0$.

Note that, according to the standard used, the input layer is not considered in the count of the total number of layers of the networks ℓ , so:

$$\ell = \#hidden_layers + output_layer \quad (4.9)$$

Given a network made by three layers (1 input layer, 1 hidden layer and 1 output layer), with n_i input layer nodes, n_h hidden layer nodes and n_o output layer node, the total number of trainable parameters (weights and bias terms) are:

- in the hidden layers: $n_h \times n_i + n_h$.
- in the output layer: $n_o \times n_h + n_o$.

In the example showed in Figure 99, $n_i = 3$, $n_h = g$, $n_o = 1$.

The implementation of the previous equations, relative to the simple network in Figure 99, justifies the name attributed to the Feed-Forward Neural Network; in fact, the information flow moves in forward direction, from the input layer to the output one.

4.2.5 LEARNING PROCESS

Introduced the two main architectures of the neural networks and described the equations that regulate the feed-forward propagation for solving a regression problem, in this sub-paragraph it will be described the classical procedure used by the FFNN to learn from the training data. This important step, belongs to the backward section of the general neural network working process, described in Figure 98.

The training data are made by input data x , and the output data y , where the latter is the true target of the problem. After set the typology of the network, so the number of layers and the number of nodes for each layer, the input data will go inside the network, according to the forward training process, as described in the previous sub-paragraph.

Suppose to have N input-output pairs of data, $\{(x_1, y_1), \dots, (x_N, y_N)\}$, the target is to obtain $\hat{y}_i \approx y_i$. To evaluate the accuracy of the model, in a single training sample, it is used the loss function \mathcal{L} ; instead, for the evaluation of the model over the total number of samples, it is defined the cost function J . In equation, $J = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\hat{y}_i, y_i)$. [63]

However, in this description it will be used only the cost function J .

A classical J function, normally employed to introduce to the updating parameter phase, is the Mean-Squared Error, defined as the equation 4.10. [64]

$$J(\vartheta) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2. \quad (4.10)$$

where:

- N is the number of data samples in the training set.
- ϑ , is a compact way to define all the trainable parameters of the model, so weights w and biases b of the layers.
- y_i , \hat{y}_i are, respectively, the real and the estimated output associated to the i^{th} training sample.

According to the value of the J , the network will use a certain optimizer criterion, or optimization algorithm, to update the nodes' weights and biases values of the layers with the goal of trying to minimize the loss function, and so, to have an estimated output closer to the real one, Figure 100. The process described works cyclically for a number of iterations equal to the training samples N . The update of the weights and of the biases are also called neural network learning process, and they are performed according to the optimizer algorithm chosen. [56]

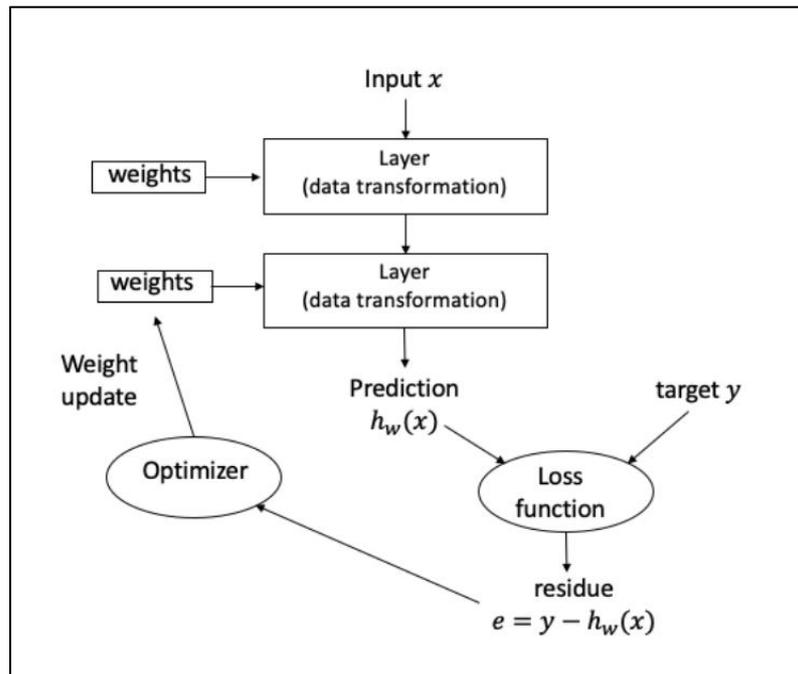


Figure 100. Working principle of the DNN. [54]

To give a simple idea on how the training process works, it is here described the gradient descent learning algorithm. The starting point is the definition of the derivate, and its usage in the computation of the minimum of a function. As known from calculus, the derivate is a measure on how fast the function is changing taken a very small step in positive direction, $\nabla y = f' \nabla x$. [65] [66]

Suppose that the goal is to find the parameter value x that minimizes the function. For simplicity, the function to minimize is $f(x) = x^2$, Figure 101. To reach the minimum, placed in $x = 0$, it is necessary to compute the function derivate with respect to the variable x , so $f(x)' = \frac{df(x)}{dx} = 2x$.

Since the algorithm reaches the solution iteratively, step by step, supposing that the starting value assumed by x is 3, then, the derivate of $f(3)' = 6$. This means that if it is taken a step towards the positive direction, the function will change proportionally to 2, that means to go far from the minimum; instead, it is necessary to move in the opposite

direction. The same thing happens in the opposite case, for the negative initial choice of the variable x .

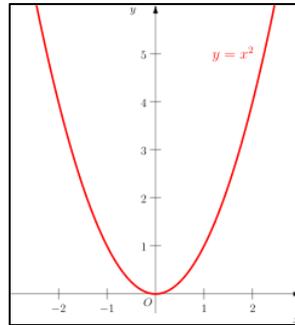


Figure 101. $f(x) = x^2$.

A general behavior can be observed, and so, it can be concluded that, to reach the minimum, the step must be done in the opposite direction than the derivate sign. The problem is to understand how big has to be this step, that is called learning rate α . This will be one of the several hyperparameters of the neural network architecture. [66] In general, if α is chosen too high, the main problem could be that the minimum point jumps far from the target and the algorithm will diverge; instead, if it is chosen too small, the convergence process could take a lot of time, Figure 102.

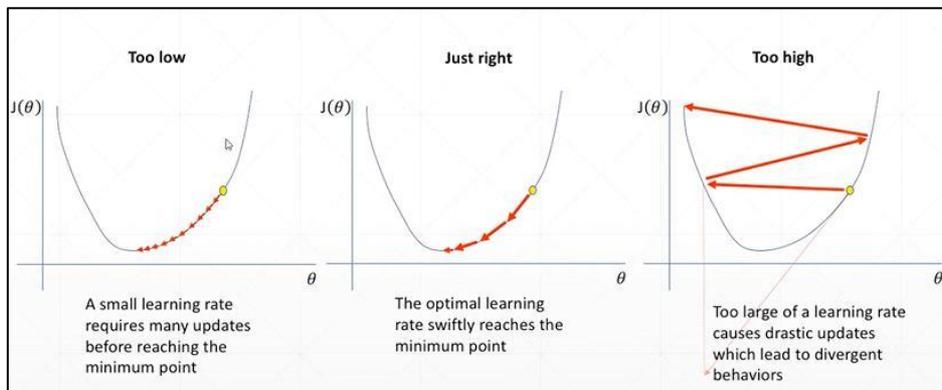


Figure 102. Example of GD algorithm with different learning rate.

In the problematic case where the loss function to minimize depends by more than one parameter it is better to talk about gradient, instead of derivate. Returning to the neural network problem, during the several iterations, the weights values have to be updated trying to reach the local or, if lucky, the global minima of the J function, Figure 103.

The update process of a single weight occurs by computing the gradient of the cost function with respect to that weight, multiplying it for a certain learning rate α and adding the negative sign; then this quantity is summed to the initial weight value. This

is done for each of all the trainable parameters $\vartheta_{ji}^{[\ell]}$ relative to the layer ℓ and thus, for all the weights $w_{ji}^{[\ell]}$ and biases $b_j^{[\ell]}$ of the networks' nodes.

The equation that represents the update for weights and biases, at each iteration of the Gradient Descend algorithm, is [67] :

$$\vartheta_{ji}^{[\ell]} := \vartheta_{ji}^{[\ell]} - \alpha \frac{\partial J(\vartheta)}{\partial \vartheta_{ji}^{[\ell]}} \quad (4.11)$$

To a single update of the weights in a particular iteration, all the training samples have to be run in the network since the cost function J has to be computed and this is the average of the N loss functions \mathcal{L} .

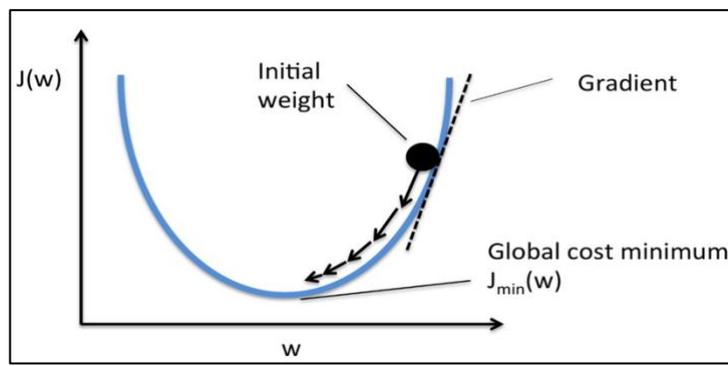


Figure 103. Gradient Descent process.

Concerning the optimization algorithms for regression problems, it is possible to choose among: Gradient Descent (GD), Stochastic Gradient Descent (SGD), SGD with momentum, Adam and many more. The simplest algorithm is the first one, but at the same time the training process is very time consuming for a huge datasets. However, for these cases it exists an alternative version, called SGD. This algorithm has the possibility to choose the dimension of the batch size that contains the training samples to run into the model before the weights are updated; the SGD has a classical version and a Minibatch one. In both cases, the computational cost is reduced, but also the accuracy. [66]

The most common algorithm adopted to solve a huge number of deep learning problems is the Adam, that is faster in convergence rather than the SGD, but requires more computational power. [56]

In all the algorithms, however, a common point is the computation of the gradient of the cost function with respect to all the several trainable parameters ϑ_{ji} . This operation is performed using the so called backpropagation algorithm. The backpropagation method was initiated in 1970s as a general optimization algorithm for solving differentiation of nested functions' problem. In 1986s, it became important also for machine learning

problems, with the paper published by Rumelhart, Hilton, and Williams, titled ‘Learning Representation by Back-Propagation Errors’. [64]

For more information about the backpropagation algorithm, it is advised read the ‘Deep Learning’ book. [68]

The training algorithm ends when all the iterations are completed with the goal of minimizing the cost function. To be clear, the term iteration is not correct, but it would be better to use the term epoch, since iteration is more appropriate to be employed talking about the batch size concept. Another important point to observe is that since, in many cases, most of the problems are not linear and very complex, the algorithms are not able to find the global minimum, but only local minima points. Moreover, the neural network training process, by construction, is stochastic and so, if the network is trained many times the results achieved will not be the same.

4.2.6 PROBLEMS DURING THE TRAINING PROCESS

During the training it is important to check the model learning process and how it changes with respect to the epochs. Its monitoring is made possible evaluating a curve, that is called learning curve. It is a plot that shows the progress of the specific metric with respect to the number of epochs and gives the idea about the training performances. So, the curve is characterized: in the x-axis, by the number of epochs, in the y-axis, by the error or metric chosen. A typical learning curve is the one that represents the loss over time, where the loss gives an idea about how bad the neural network model is doing. Another one, is the curve that represents the accuracy that, in contrast with the loss curve, the higher is the better is the model, Figure 104. [69]

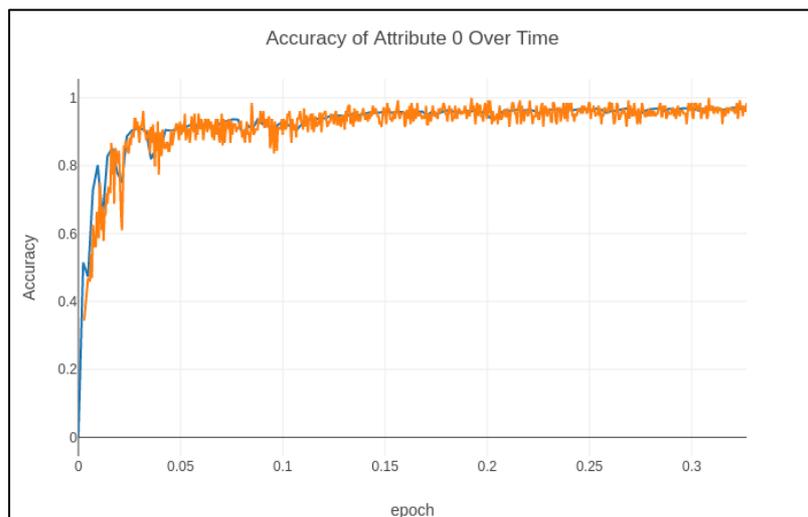


Figure 104. Example of model with high accuracy. The blue plot represents the accuracy of the training set, the orange the one for the validation set. [63]

From Figure 104, it can be seen that the accuracy curve comes higher in time, and this means that the model is improving its experience and is learning well; then, when it reaches a plateau the learning process is concluded. In Figure 104 are plotted two lines, one that represents the accuracy trend of the training set (blue line), and the other the accuracy trend of the validation set (orange line). In general, the training loss indicates how well the model fits to the training data, while the validation loss, how well the model fits to the new unseen data, always during the training process. [69]

Another important curve to check during the training process of a neural network is the one obtained by combining the performance metrics (e.g., RMSE or MAE) of training and validation sets, with respect to the epochs elapsed.

The reason to check these curves is to prevent from overfitting or underfitting problems. The training set and the validation set have been nominated so, it is better to explain what they are. The training set, as it can be understood by the name itself, is a set of samples that is used to perform the learning process and to proceed with the training of the neural network. The validation set, instead, is a partition of the training set, used to test the model during its training. In general, its dimension is about 20-30% of the total training set.

In real cases, it is employed another dataset, called testing set. The testing set is defined to test and evaluate the developed model on new data and so, to check if it benefits of the generalizability property.

OVERFITTING

The overfitting is a scenario that characterizes a model able to fit very well the training data, but has a poor fitting behavior in new unseen data during the training phase. So, an overfitting model does not enjoy of the generalizations property. [69]

In other terms, the model has memorized the training data instead of learning the input-output relationship. In statistical terms, the overfitting corresponds to high variance problem. High variance happens if the model is too complex and it is not able to represent the simpler patterns present in the data. [69]

A model affected by overfitting it cannot be used, since cannot be able to represent new unseen data.

UNDERFITTING

The underfitting problem is the opposite situation than the previous case so, here, the neural network found is too simple for describing the complexity of the true model.

In this case, the network is not able to model either the training data or the new one and, as consequence, the error is very high and does not decrease with the increment of the epochs. [69]

The underfitting situation is identical to the statistical one when model has a high bias. High bias happens when the developed model is not able to take into account all the relevant information and so, it cannot be able to represent the complexity of the real model. [69]

In Figure 105, an example of function, that is modeled by three different models, can be observed. The model on the left is affected by underfitting: it can be noticed from the fact that the blue line of the model is totally different with respect to the green one of the original function.

The one on the right, on the opposite, presents overfitting, in fact the blue line is too much complex that the green one, particularly in the first part. The optimal choice is the picture in the center.

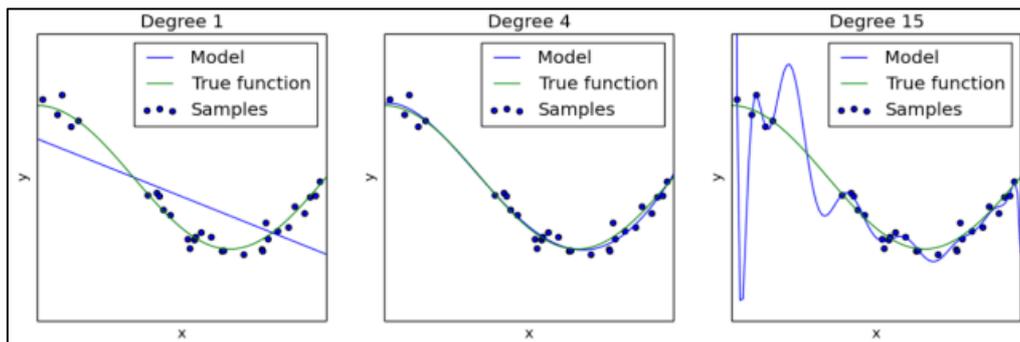


Figure 105. Example of function represented by a model with underfitting (left), overfitting (right), and optimal model (center).

The overfitting and underfitting problems can be checked also in a simpler way, by monitoring the learning curves that represent together the training and the validation losses over the number of epochs spent. A general example is illustrated in Figure 106.

A small comment on the Figure 106 is necessary to describe case by case. Starting with the first one, the underfitting model, it can be observed that the classical shape is to have the validation and the training loss, or error, curves closer with each other, but in both cases the error values are higher and does not decrease over the number of epochs.

The second scenario, the overfitting case, is characterized by a training loss curve that decreases in time, while the validation one, after starting to decrease, reaches a turning point and starts to go up again. [69]

The optimal choice is a combination of the good points of the two previous cases. The results found can be merged together as showed in Table 10.

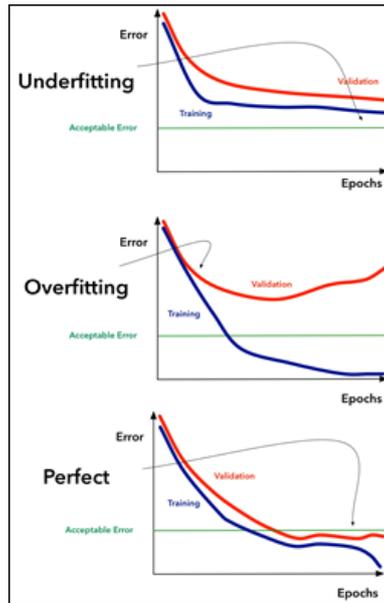


Figure 106. Example of underfitting, overfitting and optimal model in learning curves.

Training Error	Validation Error	Model
High	High	Underfitted
High	Low	Unlikely
Low	High	Overfitted
Low	Low	Optimal

Table 10. Table to check underfitting and overfitting of the model by the error values.

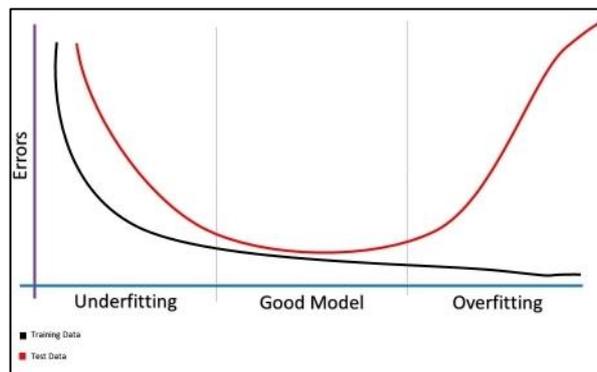


Figure 107. Example of possible learning curves scenario.

Image to have a situation like the one showed in Figure 107. The optimal choice is to stop the learning process of the algorithm in correspondence of the epochs' range denoted with "Good Model" label. In fact, there, both the curves are closer with each other, and the error have a low value. This is a practical example that demonstrates why the learning curves are fundamental in the training process of a neural network. With this example,

it has been possible to put in evidence how the user has the possibility to interfere, during the training process, and prevent from underfitting or overfitting situations.

How it can be solved overfitting and underfitting problems?

Starting with the overfitting problem, the possible solutions are: adding more data, data augmentation, regularization, and removing features from input data. [70]

Without going into details, it can be explained what the previous solutions mean. If the dataset chosen is poor, it contains few sample data, and these do not give to the model the possibility to learn the input-output relationships. Adding new data to the algorithm could give the chance to improve its performances, but in many cases, the collection of the data is not easy or could be very expensive and so, this is not always an actual solution. [70]

For these reasons, if possible, it is recommended to use the data augmentation.

The regularization is a technique that introduces some penalties to the model, with the aim to prompt the model to try to avoid these. The last possibility is to use less features as inputs of the model: in fact, if the overfitting is present, this means that the model is not able to have a generalized behavior and it is too specific. [70]

For regression problems, the most common choices among these are: the first one and the last two.

The underfitting situation, on the contrary, can be solved applying one or more of the following possible solutions: increasing the complexity of the model, reducing regularization, and adding features to the training set. All of these have the opposite effect that the ones described before. Note that, the solution that adds more data the training set is not considered for the underfitting situation. [70]

To Find, solve, or correct the overfitting or underfitting problem is not an easy task and, in many situations, the listed solutions here could not be sufficient.

4.2.7 EVALUATION METRICS

The choice of the metric is an important step, since it provides an idea about how good the model is working. It can be used both for checking the presence of overfitting or underfitting, evaluating the training and the validation dataset performances in term of loss functions, and to check the accuracy of the model analyzing the testing samples of data.

A lot of metrics exists and, all of these, can be used for any type of problem, but to be more precise, each problem has a certain number of metrics that are suitable to be applied to it. It is possible distinguish among 5 categories: the regression metrics, the probabilistic metrics, the accuracy metrics, the classification metrics, and the one used for image segmentation problems. Since the model to be developed in this work is a regression

problem, it will be described some of the main important metrics belongs to this category. Among these, there are: the Mean Absolute Error, Mean Squared Error, and Root Mean Squared Error.

MEAN ABSOLUTE ERROR

The Mean Absolute Error, or MAE, is defined as the average of the absolute difference between the predicted output and the real one. In practical terms, it gives an idea about how far the predictions are with respect to the true outputs. The negative point is that it does not give any information if the model is under-predicting or over-predicting the data, since it does not consider the direction of the error but only its magnitude. [71]

The lower the MAE's values are, the better the model is and so, better the estimation will be. The characteristic equation of the MAE is the following one [71] :

$$MAE = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i| \quad (4.11)$$

where:

- y_i , is the true value.
- \hat{y}_i , is the predicted value.
- N , is the number of sample data.

MEAN SQUARED ERROR

The Mean Squared Error, also called MSE, is a metric like the MAE. The difference is that, while the latter one computes the average absolute error, this one calculates the average of the square of the difference between the predicted output values and the real ones. This formulation puts in evidence larger errors rather than the smaller ones. [71]

The quantity is always positive with a values that tends to decrease as the error approaches zero value. The goal in a regression problem is to minimize this quantity, that means having an accurate model. The equation of the MSE is [71] :

$$MSE = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 \quad (4.12)$$

where:

- y_i , is the true value.

- \hat{y}_i , is the predicted value.
- N , is the number of sample data.

ROOT MEAN SQUARED ERROR

The Root Mean Squared Error, or RMSE, is the standard deviation of the residuals. In practical terms, it indicates the spread out of the residuals than the best fit line. The residual is the measure of how far the predicted point is with respect to the true one. [72] The mathematical formulation is given by the equation 4.13.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2} \quad (4.13)$$

where:

- y_i , is the true value.
- \hat{y}_i , is the predicted value.
- N , is the number of sample data.

4.2.8 ACTIVATION FUNCTIONS

As described in previous paragraphs, inside the nodes it is implemented an important function whose objective is to compute the node output, given a certain input value, and it is called activation function. There are several typologies of activation functions, that can be used only for regression problems, classification problems or both. The most common are listed here:

- Binary Step Function.
- Linear.
- Sigmoid.
- Softmax.
- Hyperbolic Tangent.
- ReLU.
- Leaky ReLU.

Choosing the right activation function is not easy since are necessary a lot of studies and in many cases, the right choice is moved without following a guideline, but with a trial-

and-error approach; so, a rule of thumb does not exist. However, some common and practical advances to follow are described here. [73]

- 1) There are some activation functions that works better for regression problems, and others that are more performed in classification problems. For example, in the binary classification problem, the activation function recommended for the output layer node is the Sigmoid, instead, for the multiclass classification problem, the Softmax.
- 2) To avoid the vanishing gradient problem, try to not use the Tanh and the Sigmoid function.
- 3) If there are some dead nodes, then try to use the Leaky ReLU.
- 4) A common choice for the activation functions in the hidden layers nodes are the ReLU, whose performances are better than the others.
- 5) In regression problems, it has to be used, as output layer node, the linear function.

BINARY STEP FUNCTION

Also called Threshold, or Heaviside function, this is the simplest activation function, and it is commonly used in binary classification problems, where the output could be among two different alternatives. As the name itself implies, this function has a binary output (0 or 1) and consequently, there will be an adjustable threshold that establishes whether the output is 0 or 1. In general, if the threshold is overcome, the output will be set to 1, otherwise to 0, Figure 108.

With a threshold set to 0, the equation of this function is:

$$\sigma(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \quad (4.14)$$

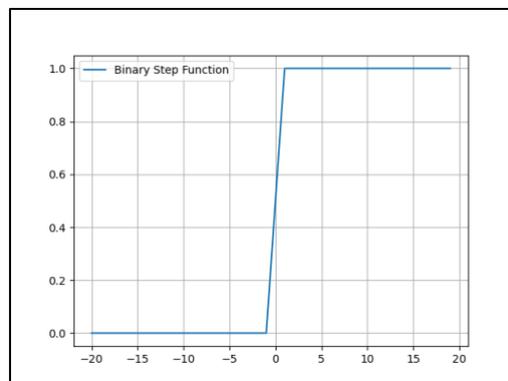


Figure 108. Binary Step function.

If this function is implemented in a neuron present in the hidden layer, it is able to establish if the neuron output will be used as input of the next layer nodes or not; in fact, if the output is 1, the neuron is activated and its output will be used as the input of the next layer nodes; otherwise, it is turned off and it will never be used again. [73]

A backwards of this function is that it can cause a hinderance in backpropagation step, since the gradient of the Binary Step Function is zero. [73]

LINEAR

As it can be seen from Figure 109, the linear activation function is directly proportional to the input, unless a multiplication constant factor k . With this activation function, the backpropagation can be realized since the gradient of the linear function is the constant term k . Nevertheless, this function is not so common in machine learning algorithms, due to the fact that the gradient does not change during the several epochs and, as consequence, the error is not improved. [73]

Common applications in which it is employed are linear regression problems. It is defined as equation 4.15.

$$\sigma(z) = kz \tag{4.15}$$

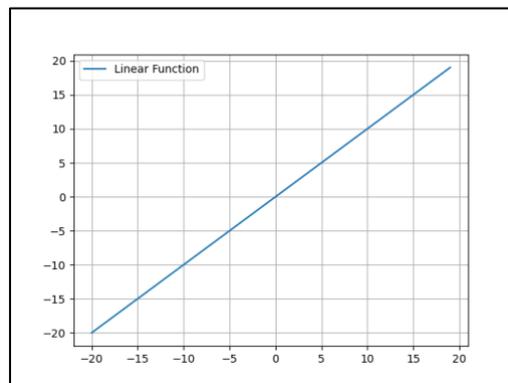


Figure 109. Linear function.

SIGMOID

The Sigmoid is the classical activation function used in neural networks problems, since it is a non-linear function. According to equation 4.16, this function maps the input into

an output, whose range is [0-1], as showed in Figure 110. Obviously, if the user wants a different output range, the function has to be scaled. [73]

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (4.16)$$

The reason why this function is among the most popular in the NN world is that it is differentiable everywhere and so, the backpropagation algorithm works fine. The Sigmoid derivative is:

$$\sigma'(z) = 1 - \frac{1}{e^{-z}} = \text{sigmoid}(z) \cdot (1 - \text{sigmoid}(z)) \quad (4.17)$$

Studies have demonstrated that Sigmoid function is not suitable for hidden layers, due to the gradient function that becomes very little as the inputs become very large or very small: the consequence is that the gradient descent algorithm can slow down.

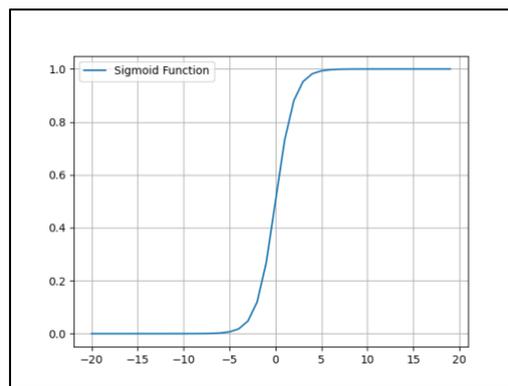


Figure 110. Sigmoid function.

SOFTMAX

The Softmax function is obtained by the combination of multiple Sigmoid functions. It can be used in such cases of probabilities data analysis. While the Sigmoid was used for binary classification problems, this function is normally employed for multiclass classification problems and so, when the possible outcome varies among several choices. [73]

Obviously, in multiclass classification problems it is used as activation function for the output layer node; the number of output layer nodes will be set equal to the number of classes of the problem. [73]

In equation, it is expressed according to the following formula:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}} \quad \text{for } j = 1, \dots, N \quad (4.18)$$

HYPERBOLIC TANGENT

The Hyperbolic Tangent function, also called Tanh, is closer to the Sigmoid function, but it is symmetric with respect to the origin, according to Figure 111. As result of this symmetry, the output of the node can have both positive and negative values and so, it can assume a value inside the range $[-1,1]$. As the Sigmoid function, it is continuous and differentiable everywhere.

In general, the Tanh function is preferred to the Sigmoid one, since its derivate is steeper. In addition, it is able to move in all the directions (positive and negative) and it is centered in zero. [73]

However, it is not a good practice to use in the hidden layers nodes, for the same Sigmoid function's reasons. In formula, the Tanh is described according to equation 4.19.

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \left(\frac{2}{1 + e^{-2z}} \right) - 1 = 2\text{sigmoid}(2z) - 1 \quad (4.19)$$

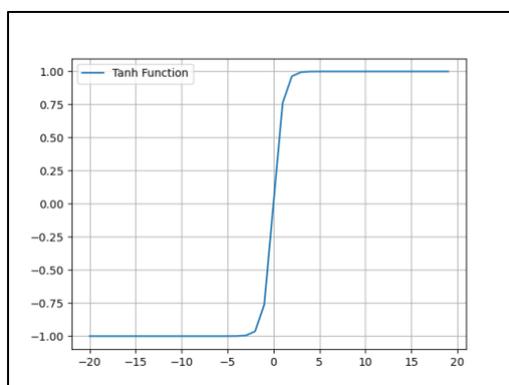


Figure 111. Hyperbolic tangent function.

ReLU & Leaky ReLU

The Rectified Linear Unit, or ReLU is a common non-linear function used in neural networks, and particularly, for the hidden layers nodes.

Figure 112 shows an example of ReLU function, that in formula can be expressed according to equation 4.20.

$$\sigma(z) = \max\{0, z\} \quad (4.20)$$

The reason why it is very popular, is related to its mathematical expression; in fact, not all the nodes of the layer will be activated at same time. A bad point of this function is that often the gradient is zero and so, this implies that the weights and the biases are not updated during the backpropagation process. A solution to this problem is given by the usage of two different versions of the ReLU function, that are: the Leaky and the Parametrized Leaky ReLU. [73]

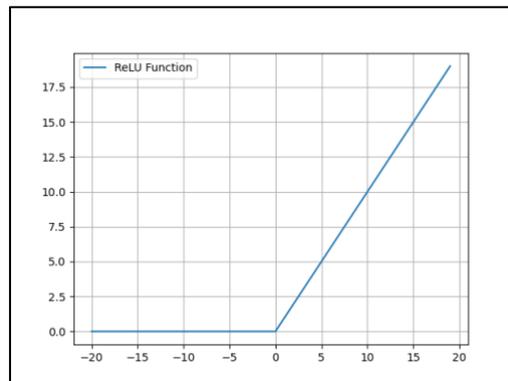


Figure 112. ReLU function.

The Leaky ReLU, is an activation function that derives from the ReLU but, for negative inputs, the function does not assume zero value but very small negative values, as it can be shown in the equation 4.21. [73]

$$\sigma(z) = \begin{cases} z, & z \geq 0 \\ 0.01z, & z < 0 \end{cases} \quad (4.21)$$

In Figure 113, it can be observed the general behavior of the Leaky ReLU.

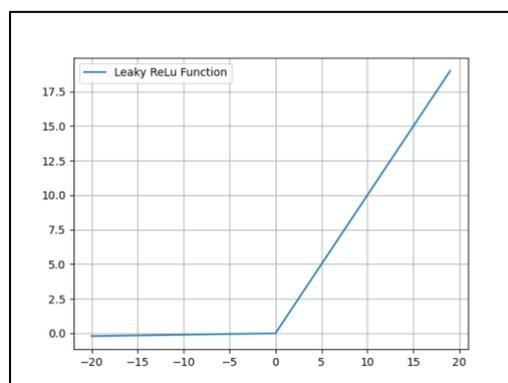


Figure 113. Leaky ReLU function.

In many cases, neither the Leaky version leads to have good results.

A possible solution is to use the Parametrized ReLU function, that is expressed according to equation 4.22. [73]

This version of the function can be showed in Figure 114.

$$\sigma(z) = \begin{cases} z, & z \geq 0 \\ kz, & z < 0 \end{cases} \quad (4.22)$$

where:

- k is a parameter chosen by the user. If $k = 0.01$ the Leaky function is obtained.

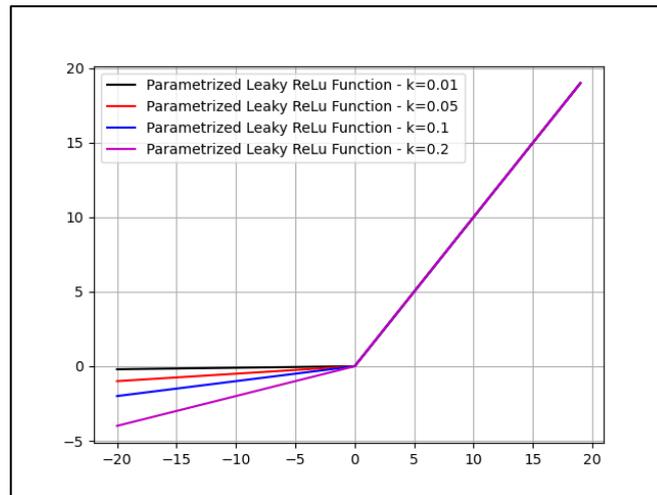


Figure 114. Parametrized Leaky ReLU function, with several parameters' values.

4.3 DESIGN OF A NEURAL NETWORK FOR THE SOC ESTIMATION PROBLEM

In this paragraph, the neural network design process to solve the state of charge offline estimation problem, in a fully electric vehicle, will be illustrated. In addition, the framework used, the data chosen, the selected model and the results achieved will be described.

The algorithm was written totally in Python language using a desktop graphical user interface (GUI), called Anaconda-Navigator, belonging to the Anaconda distribution group. It was chosen for simplicity in running the application, manage packages and work environments, without using the command lines. [74]

The application used to code and prototype the neural network algorithm was Spyder IDE (Python 3.7), a scientific python development environment, free and open-source available from Anaconda package manager. [75]

Among the several open-source external packages needed to easily work with the deep learning framework and used in this work, there were: Keras API [76], TensorFlow 2 [77], and Scikit-learn [78].

The computer used to train and test the neural network model was a MacBook Pro, with the following features:

- Processor, 2.7 GHz Intel Core i5.
- Memory, 8GB 1867 MHz DDR3.
- GPU, Intel Iris Graphics 6100 1536 MB.

The selected data used for this problem were taken from the dataset provided by the Bylogix S.r.l company. It contains about 120 thousands of data, acquired from a Tesla Model 3 with a 75 kWh battery pack, with a frequency of 1 Hz. The dataset features were 14, as it was well described in the paragraph 1.4 of the chapter 1. Note that, since the frequency is not so high, it is expected that several high dynamic behaviors were not recorded and so, could not be learned by the network.

The dataset contains several measurements that correspond to different drive cycles, collected in correspondence of different driving scenarios, so different weather conditions, different road typologies, different velocity and altitude profiles.

In Figure 115, it can be seen an example of data associated to a route, extracted from the original dataset and used for the estimation problem.

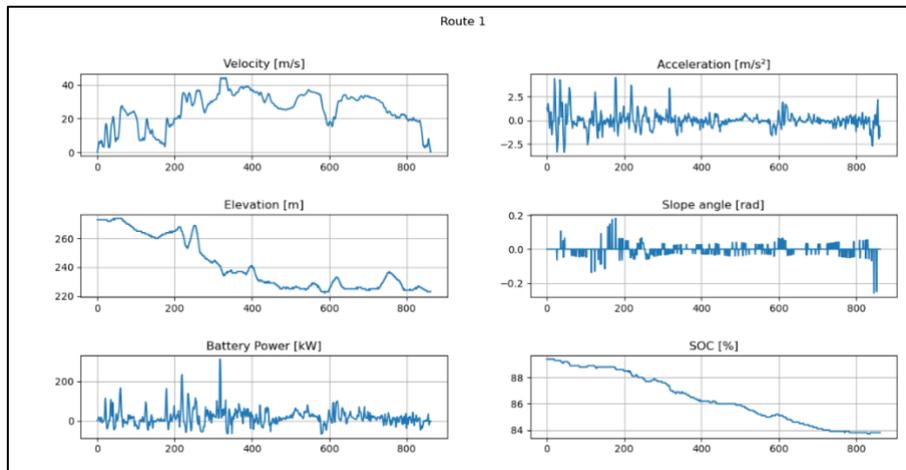


Figure 115. Example of data used in the neural network model design.

The workflow followed to develop the neural network is illustrated in Figure 116.

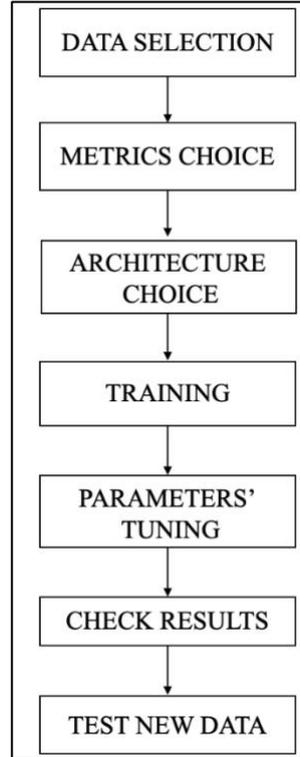


Figure 116. Workflow for neural network model design.

4.3.1 DATA SELECTION

Among all the data features provided in the dataset, it was selected as input set $X(t)$, the one containing these 3 features: the vehicle velocity, the acceleration, and the road slope angle. The output variable of the network $Y(t)$ was the battery power consumption. This quantity was possible to be computed using real data measurements of current (I) and voltage (V), presented inside the dataset.

$$X(t) = [v(t), \dot{v}(t), \theta(t)]$$

$$Y(t) = [P_{BATT}(t)]$$

where:

- $v \left[\frac{\text{m}}{\text{s}} \right]$, is the vehicle velocity.
- $\dot{v} \left[\frac{\text{m}}{\text{s}^2} \right]$, is the vehicle acceleration.
- θ [rad], is the slope road angle value.
- $P_{BATT} = \frac{(V \cdot I)}{1000}$ [kW], is the battery power consumption.
- t [s], is the time sample.

Comparing the input and the output data of the network (X, Y) with respect to the standard ones used, in other proposed neural network based solutions, for solving the SOC estimation problem, it is possible to observe some differences. In fact, in most of the cases, the classical inputs used were data coming from the battery pack's cells, like voltage (V), current (I) and ambient temperature (T), while, the classical output used was the SOC.

In this case, this was not possible to do since the model working scenario was offline and this implies to estimate the SOC consumption, associated to a certain route, before that the trip begins and, in addition, using data associated to the vehicle motion. Moreover, the voltage and current data were associated to the whole battery pack, and not to the single cell. So, in design phase, the possible paths to follow were two: the first one was to estimate the battery power consumption, using the selected input data ($X(t) = [v(t), \dot{v}(t), \theta(t)]$) and then, by integrating this quantity, to obtain the energy consumption and so, the SOC. The second approach was to compute the battery power consumption, using the methodology of the algorithm 1, as described in chapter 3 and then, to obtain the current as $I = \frac{P_{BATT}}{V}$, where V is the nominal voltage of the battery pack. In this second case, the input data and output of the network would be: $X(t) = [I(t), V(t), T(t)]$, $Y(t) = [SOC(t)]$.

Nonetheless, the second possibility had several negative points: the value of the nominal voltage of the battery pack is not constant as should be supposed, but changes as function of the velocity and other factors related to the motion of the vehicle, that are not easy to predict offline. In addition, since these measurements data would be related to the entire battery pack, it would be necessary to use conversion equations to scale them for a single cell, that have the drawback to introduce some errors.

However, it was chosen to follow the first idea, that explains, indeed, the attribute 'new' to the approach developed to solve the problem.

At this point, the dataset was divided into training, validation and testing set.

The validation set was extracted directly from the training set, and it was employed to check if the problem is suffered of overfitting or underfitting; instead, the testing set was used to evaluate the network performances to new and unseen data. Note that the testing dataset was made by different drive cycles extracted from the original dataset.

The training set was the 80% of the total dataset, the testing set was the remaining 20%, instead the validation set was extracted from the training set, by setting a portion of the 20%, Table 11.

Dataset	Samples
Total dataset	120.687
Training set	97.581
Validation set	19.516
Test set	23.106

Table 11. Dataset division.

The input data were pre-processed to obtain a more robust and better performance networks. A common technique used to obtain this result is the scaling of the input data to a standard range, and this can be achieved using the data normalization or standardization. In this work it was operated the normalization.

When the normalization is performed, the data are rescaled from the original range to a new one, that stays in [0,1]. [78]

The formula associated to the data normalization is the following one:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4.23)$$

where:

- x , is the input vector.
- x_{norm} , is the normalized vector.
- x_{max} , is the maximum value of the input vector.
- x_{min} , is the minimum value of the input vector.

A good practice used in machine learning when scaling techniques are applied, is to follow this procedure:

1. Use the training dataset to estimate the min and max observed values or, in other words, to fit the scaler function to the training samples.
2. Use the normalized data as input for training the model.
3. Transform all the future input data (e.g. testing data) by scaling these with the scaler function obtained in point 1.

The function employed was the `MinMaxScaler()`, implemented from `scikit-learn` library [78].

4.3.2 MODELS EVALUATION

After having divided the dataset, it was necessary to understand the typology of neural network to implement for solving the state of charge estimation problem in a fully electric vehicle.

In general, when it deals with machine learning problems and, particularly, with neural networks, the choice of a certain network rather than another one is difficult and in many case is not easy at all find the optimal one.

This is due mainly to the fact that a neural network design process is characterized by the choice of different variables, that can be grouped into two main classes. The first one has an influence on the final result, since it consists in the chosen of the input and output network's variables.

The second group, instead, contains the so called, hyperparameters. These are variables that influence the training process of the network, and they are: the number of hidden layers, the number of layers nodes, the activation functions, the optimization algorithm, the loss function to minimize, the choice of the metrics to evaluate the model estimation behavior, the batch size, and the number of epochs.

Each of these variables have multiple choices, thus, the selection of the optimal combination is very hard to find and so, one of the best procedure to follow is the trial-and-error one.

Some general aspects were selected to check the differences among the several models trained, for example: the architecture chosen was the feed-forward one with fully connected layers, the optimizer algorithm and the loss function selected were the Adam and the MSE, respectively, and the criteria used to select the best model was linked to the one with the best compromise in terms of computational cost, complexity, and accuracy. The accuracy was measured with the following metrics: MSE, RMSE, MAE.

Other values maintained fixed during the evaluation of different models were: the epochs and batch size value, imposed equal to 250 and 100. In addition, the hidden layers have the ReLU as activation function, instead, the output layer the linear one. Different networks, designed to have the same number of layers, but different layers' nodes are tested, and the performances are compared, in order to choose the best.

Three different set of data was used to test the accuracy and the generalized behavior of the network. These were extracted from the original dataset and obviously were not used for the training session, moreover, they correspond to three different and independent drive cycles, that will be called DC1, DC2, DC3. For each route, several data were extracted and used, as showed in Figure 117.

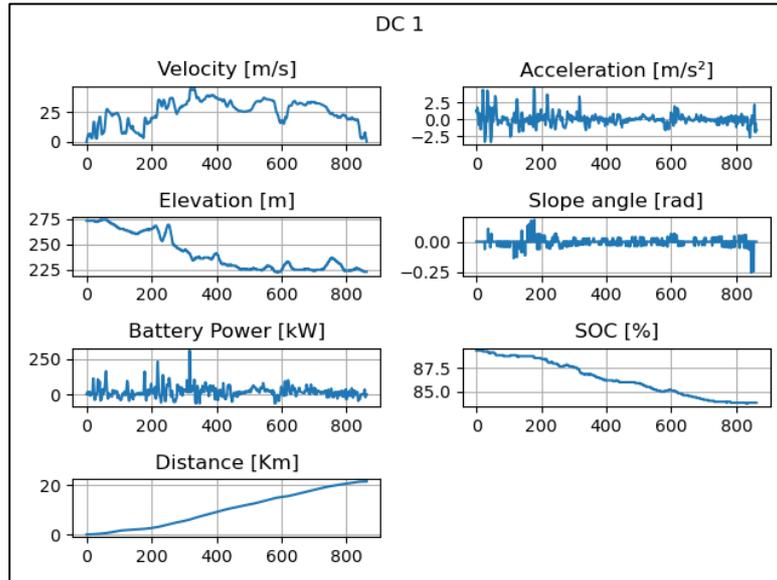


Figure 117. Example of data contained in a driven cycle, used a test set for the neural network. In all the subplot, the x-axis indicates the travel time duration [s].

In the following tables it is possible to observe the results, in terms of MAE and RMSE (approximative value), obtained by different implemented feed-forward neural networks. Note that the metrics' values are computed on the output of the network, so considering only the battery power consumption, estimated and measured. Associated to each network description, there are also the learning curves.

NN1	Value	MAE	RMSE
Layers (input, hidden output)	1, 2, 1		
Node's layers (input, hidden, output)	3, 8, 8, 1		
Training set		6.76	9.77
Validation set		6.67	9.58

Table 12. NN1, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).

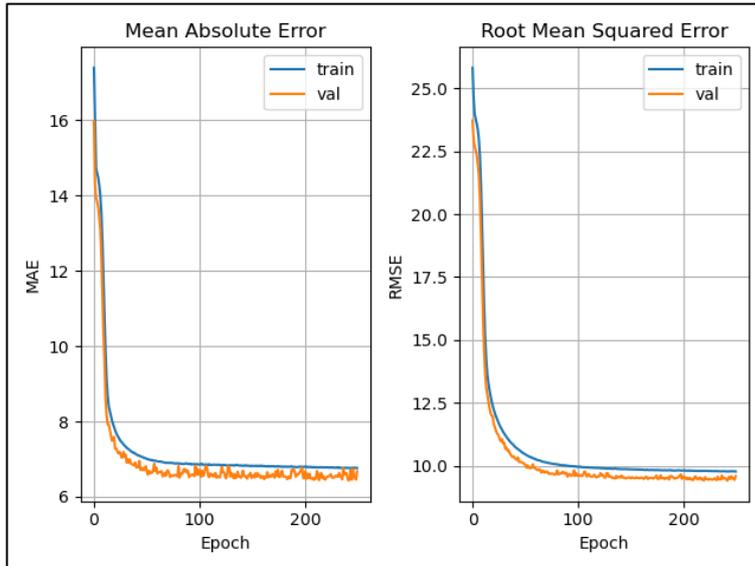


Figure 118. Learning curves for training and validation set associated to NN1.

NN2	Value	MAE	RMSE
Layers (input, hidden output)	1, 2, 1		
Node's layers (input, hidden, output)	3, 32, 32, 1		
Training set		4.21	6.14
Validation set		4.02	5.86

Table 13. NN2, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).

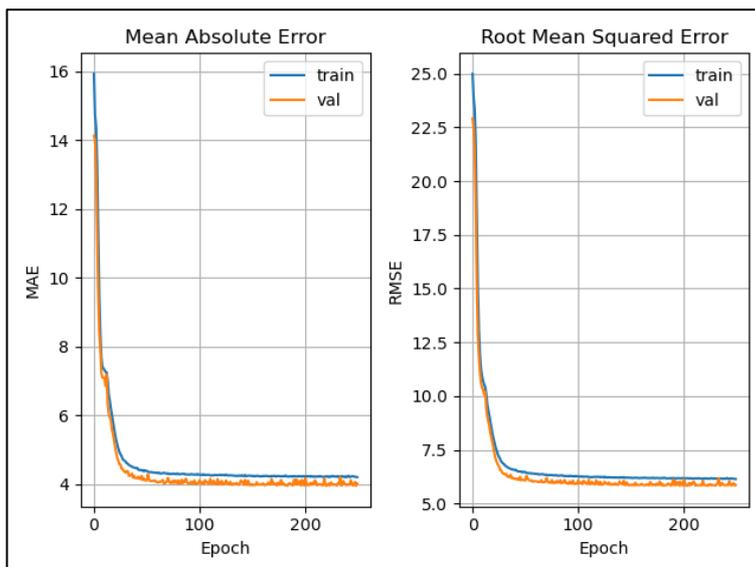


Figure 119. Learning curves for training and validation set associated to NN2.

NN3	Value	MAE	RMSE
Layers (input, hidden output)	1, 2, 1		
Node's layers (input, hidden, output)	3, 128, 128, 1		
Training set		3.62	5.36
Validation set		3.45	5.24

Table 14. NN3, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).

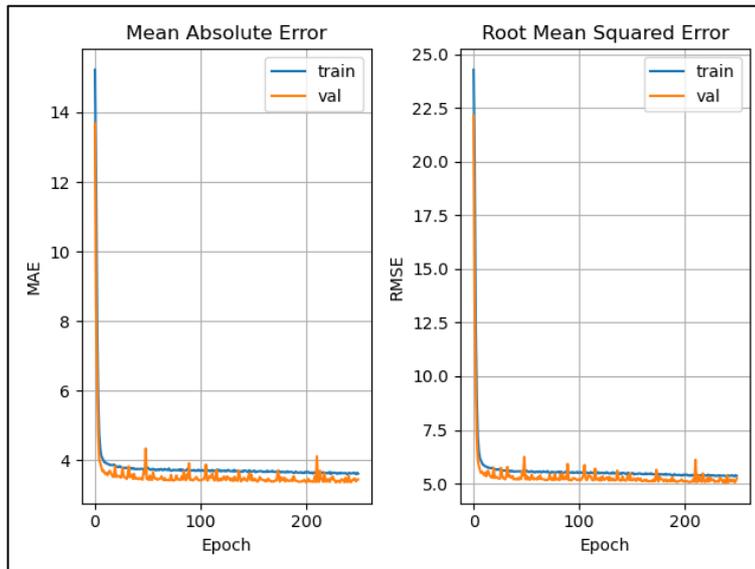


Figure 120. Learning curves for training and validation set associated to NN3.

NN4	Value	MAE	RMSE
Layers (input, hidden output)	1, 3, 1		
Node's layers (input, hidden, output)	3, 8, 8, 8, 1		
Training set		3.95	5.79
Validation set		3.67	5.44

Table 15. NN4, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).

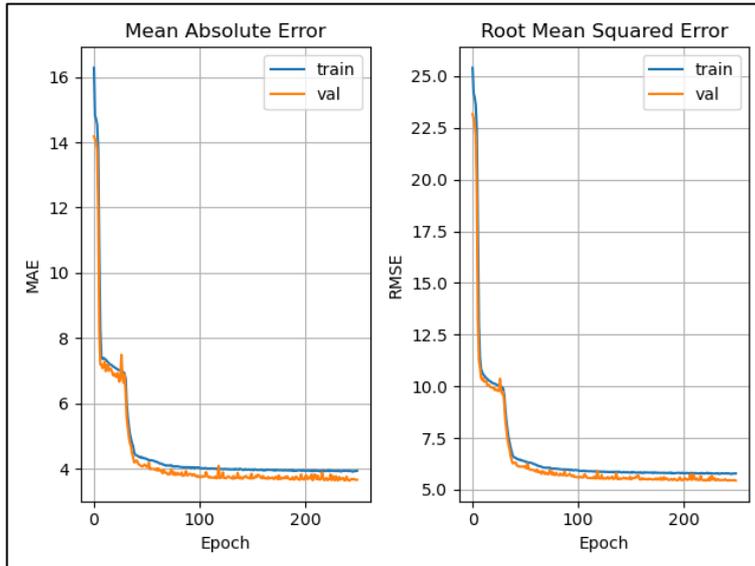


Figure 121. Learning curves for training and validation set associated to NN4.

NN5	Value	MAE	RMSE
Layers (input, hidden output)	1, 3, 1		
Node's layers (input, hidden, output)	3, 32, 32, 32, 1		
Training set		3.64	5.32
Validation set		3.49	5.09

Table 16. NN5, information about layers, nodes and evaluation metrics associated to the neural network output (battery power).

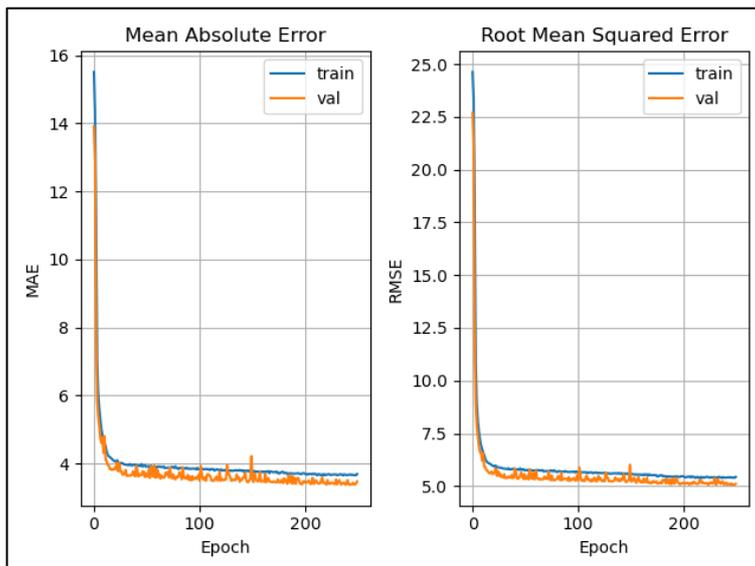


Figure 122. Learning curves for training and validation set associated to NN5.

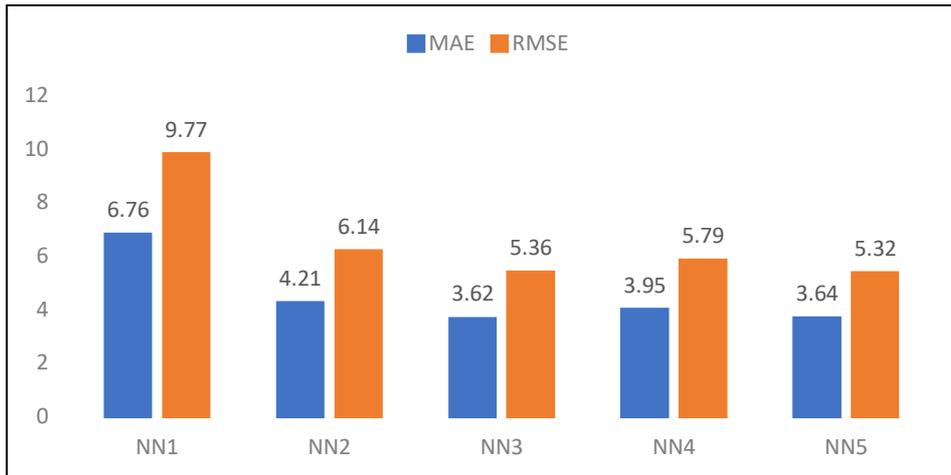


Figure 123. MAE and RMSE on the training dataset, for the networks developed.

How it can be possible to observe from the Figure 123, all the models, except the first implemented, are trained quite well. In particular, the last 3 models have very close MAE and RMSE values. Observing the previous learning curves, computed for each network developed, the two lines, representing the training and validation error, have the same behavior and after a certain number of epochs both converge to a similar value; this means that the networks were not affected by overfitting problem. From these figures, in addition, it can be observed as the orange line, associated to the validation dataset, presents a bit of oscillations, these are due to the small number of samples in the validation dataset, however this is not a problem.

In order to choose the best model, it was decided to analyze the RMSE and MAE values, obtained comparing the predicted battery power consumption and the real one. These are evaluated for each network analyzed and with respect to three datasets DC1, DC2, DC3. The results are resumed in Table 17.

NN	Layers (I,H,O)	Nodes	DC1		DC2		DC3	
			MAE	RMSE	MAE	RMSE	MAE	RMSE
1	1,2,1	3,8,8,1	9.51	13.73	8.95	12.20	6.82	9.81
2	1,2,1	3,32,32,1	5.07	7.19	5.11	7.02	5.03	6.98
3	1,2,1	3,128,128,1	4.85	6.63	5.04	7.02	4.64	6.58
4	1,3,1	3,8,8,8,1	4.80	7.08	4.87	6.96	4.53	6.34
5	1,3,1	3,32,32,32,1	4.37	6.02	4.41	6.06	3.73	5.22

Table 17. Sum-up of the RMSE and MAE values, relative to the output of the neural network, computed in the 3 test sets. In the column ‘Layers’, the letters I, H, O indicate Input, Hidden, and Output, respectively.

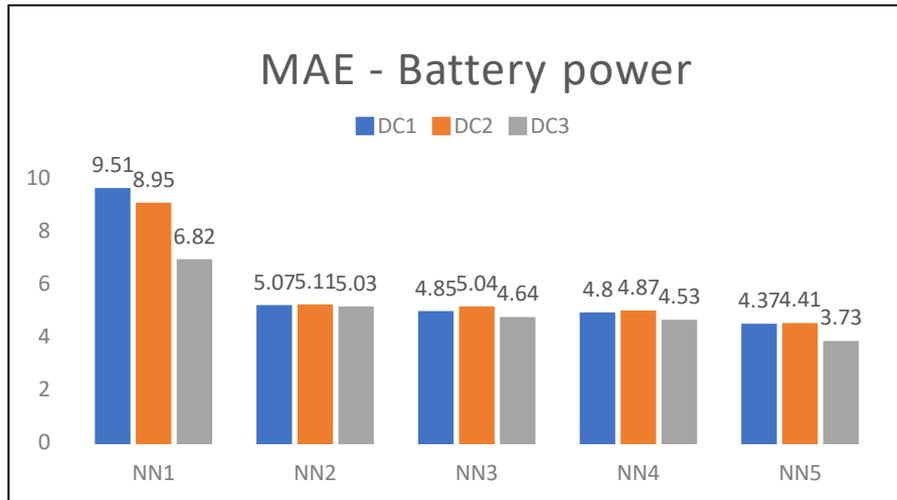


Figure 124. Evaluation of the MAE error on the battery power consumption, for the developed networks, in the three test sets, DC1, DC2, DC3.

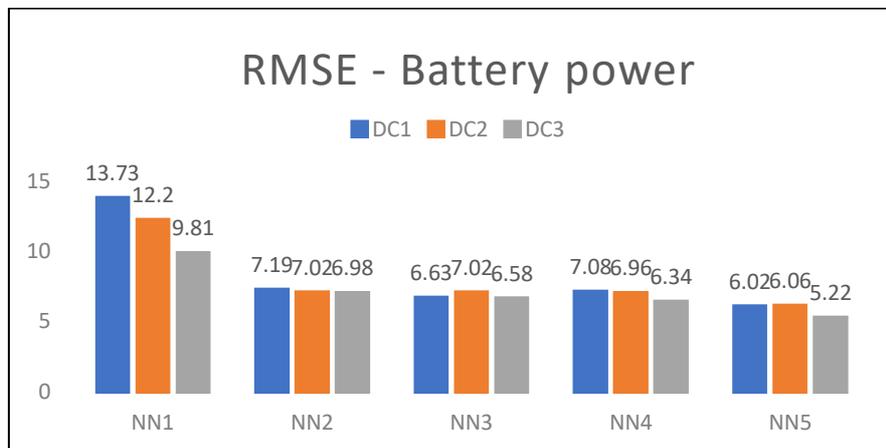


Figure 125. Evaluation of the MAE error on the battery power consumption, for the developed networks, in the three test sets, DC1, DC2, DC3.

The RMSE and MAE values, related to the estimation of the battery power consumption, that is the output of the neural network, are good; the fact that these numbers are not close to zero is related to the fact that the output assumes values that varies from -80 to +400 [kW].

To select the best model, among the ones developed, it was necessary to evaluate the performances on the error metrics, as RMSE and MAE, relative the test sets. According to Figure 124, 125, the best possible choice, in terms of complexity, time consuming and accuracy, is the network denoted as NN5. Even if it was the most complex among the five analyzed, this is not a very elaborate network, since it has only 3 hidden layers with 32 nodes for each. Other more complex networks were evaluated and the obtained results were, more or less, closer to the selected one.

4.3.3 AN OVERVIEW OF THE DEVELOPED NEURAL NETWORK MODEL

Regarding the architecture of the network, the choice was a fully connected neural network, also called FFNN, with a single input layer, 3 hidden layers and a single output layer. Since the developed network was used for a regression problem, the output layer was made by a single node, whose output was the estimated variable. The input layer contains 3 nodes, as the number of input variables. Instead, the hidden layers have 32 nodes per each. Figure 126 shows the architecture developed.

Concerning the activation functions used: for hidden layers, the ReLU functions while, for the output node the linear function.

The optimizer algorithm chosen was the Adam, that returns better results than the standards Stochastic Descend or Gradient Stochastic Descend algorithm; instead, as loss function the mean squared error was used.

The metrics elected for validation process were: the mean absolute error (MAE), the mean squared error (MSE) and the root mean squared error (RMSE).

Once set the neural network architecture and all the training settings, the values relative to the number of epochs and to the batch size were imposed.

The latter one is a parameter used when the dataset dimensions are very high and want to reduce the complexity cost required for the training session. The batch size is defined as the number of data samples that are propagated into the network at each iteration, during the training process.

For example, if a dataset is made by 1000 samples and the batch size of 100, the algorithm, instead of taking a single input sample for each iteration, takes the first 100s input samples, engages the training process and, at the end, the first iteration is concluded. Then, the second group of 100s samples are taken and the procedure is repeated, ending the second iteration. This process is done until the last batch of samples is sent to the network; so, in concomitance of the last iteration, that corresponds to the training of the last batch of data, the epoch is concluded. Thus, these series of steps are repeated until the last epoch is reached, so after 10 iterations.

The epoch can also be defined as the number of iterations to perform the forward and backward processes of all the training data.

Since the training dataset used is quite huge, it was imposed a batch size of 100. A resume of all the parameters imposed for the neural network developed, called NN5, is showed in Table 18.

Neural Network characteristics	Value
Input data	[velocity, acceleration, slope angle]
Output data	[power battery]
Layers (input, hidden, output)	1,3,1
Nodes' layers (input, hidden, output)	3,32,32,32,1
Activation function (input, hidden layer)	ReLU
Activation function (output layer)	Linear
Optimizer	Adam
Loss function	MSE
Metrics	MAE, RMSE
Batch size	100
Epochs	250

Table 18. Resume of the hyperparameters setting in the developed neural network NN5.

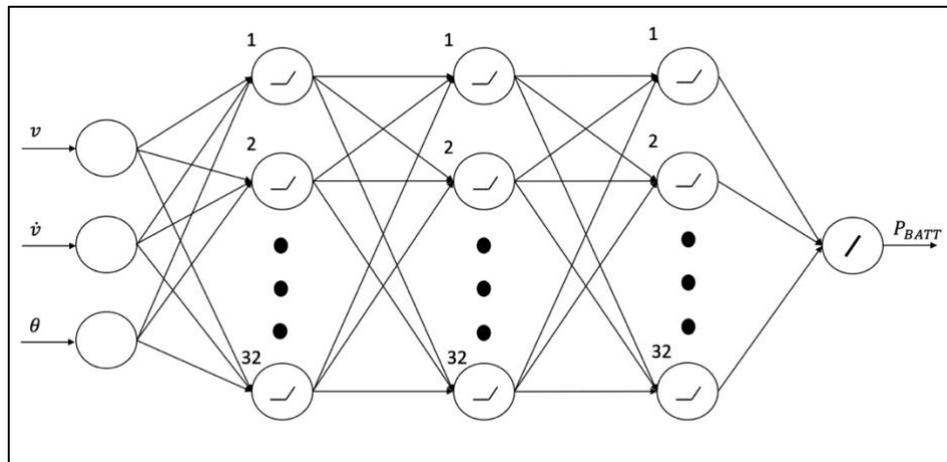


Figure 126. Architecture of the neural network NN5. In the hidden layer the node's activation functions are ReLU, instead, in the output layer is linear.

4.4 ANALYSIS OF THE RESULTS USING ANN ALGORITHM

In this sub-paragraph, the results obtained by using the developed ANN algorithm, Figure 93, choosing the neural network NN5, are showed and commented. In particular, the results in terms of energy and state of charge consumption estimated by the algorithm will be analyzed and compared with respect to real measured ones. The analysis is done for the three drive cycles: DC1, DC2, DC3.

Before illustrating the obtained results, a briefly description about the equations used to obtain the SOC estimation is done.

As said in the previous paragraph of this chapter, the neural network takes as input the vector of velocity, acceleration and slope road angle and returns as output, the estimation of the battery power consumption. By integrating this quantity, it is possible to obtain the energy, and thus, the SOC consumption, according to the equation 4.24 and 4.25.

$$E_{BATT}(t_f) = E_{BATT}(0) + \int_0^{t_f} P_{BATT} \cdot dt \quad [\text{kWh}] \quad (4.24)$$

$$SOC(t) = SOC(t_0) - \frac{E_{BATT}(t)}{C_{nominal}} \cdot 100 \quad [\%] \quad (4.25)$$

For the first equation, the initial energy value $E_{BATT}(0)$ is set equal to 0, since it is assumed that the vehicle starts in stationary conditions.

The second equation, instead, needs the knowledge of the nominal capacity $C_{nominal}$ and of the initial SOC value, $SOC(t_0)$. The first was imposed equal to 75 kWh, that is the nominal battery capacity value of the Tesla model 3, while the second one was set equal to the true initial value of the SOC.

The reason why the latter term is imposed equal to the initial state of charge value measured is purely associated to analysis purposes. In the real practical application, since it is not easy measure this quantity, it is assumed a $SOC(t_0) = 90 \%$. Note that the energy quantity was convert from J to kWh.

In the next three sub-paragraphs it is possible to check the results, obtained applying the ANN algorithm to three test sets (DC1, DC2, DC3), in terms of battery power, energy and SOC consumption. However, since for the thesis purposes the most important result is the state of charge estimation, a statistical analysis on the error made by the algorithm will be evaluated in the sub-paragraph 4.4.4.

4.4.1 DRIVE CYCLE, DC1

This first route extracted has a total length of about 22 Km, characterized by a change in the elevation profile of about -50 m, with an initial elevation value of 273 m and a final one of 223 m above the sea level. The measured SOC consumption goes from 89.4 % to 83.8 %. A series of measured data extracted from the dataset, associated to the DC1, are showed in Figure 127. Note that in Figure 127, the x-axis indicates the travel time, measured in seconds.

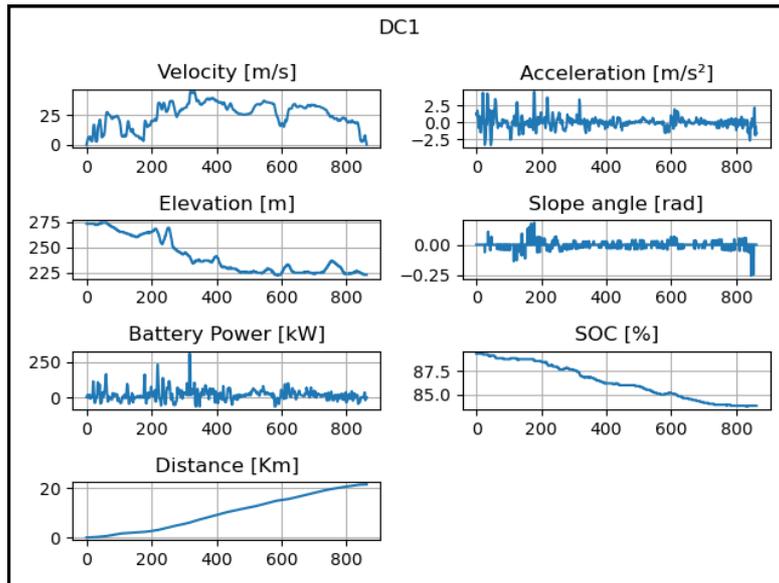


Figure 127. Drive cycle 1 data.

Given the velocity, acceleration, and slope road angle as inputs of the neural network, the battery power was estimated. A comparison among the real and the estimated quantity of battery power consumption is showed in Figure 128.

How it can be possible to see, the estimated quantity covers well the real measured one, excepting in correspondence of certain peaks values where exceed.

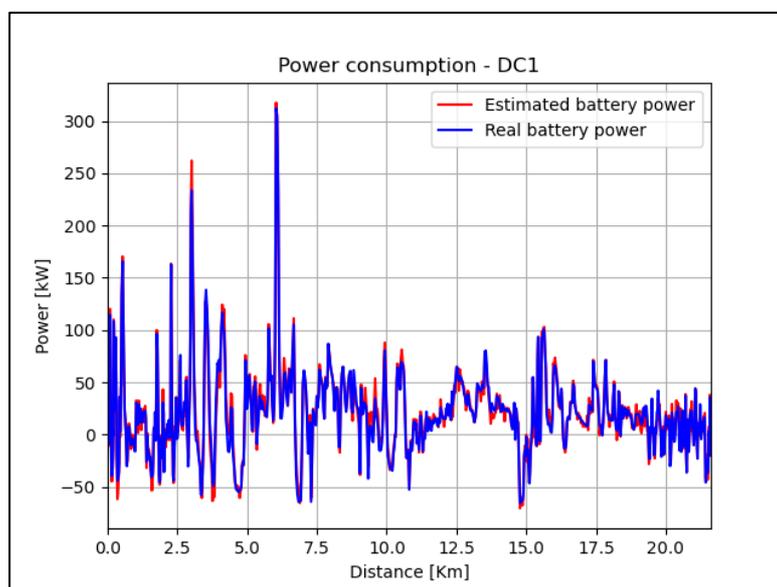


Figure 128. Comparison between real and estimated power consumption associate to DC1.

From this estimated quantity, the algorithm computes the energy and then, the SOC consumption. In Figure 129, it is showed a first subplot, where the measured energy consumption is matched with respect to the estimated one, instead, the second subplot, illustrates the associated absolute error computed among the two previous quantity,

according to the following equation: $|e| = |\hat{y} - y|$, where \hat{y} is the estimated quantity and y is the true one. The same thing is done for the SOC, as it can be seen in Figure 130.

A recap of the obtained values is illustrated in Table 19.

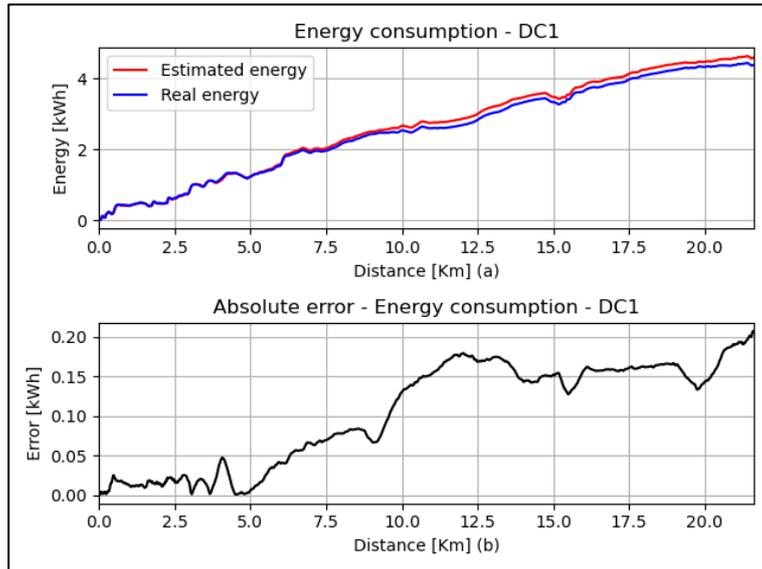


Figure 129. Comparison between real and estimated energy consumption (a) and the associated absolute error (b), relative to DC1.

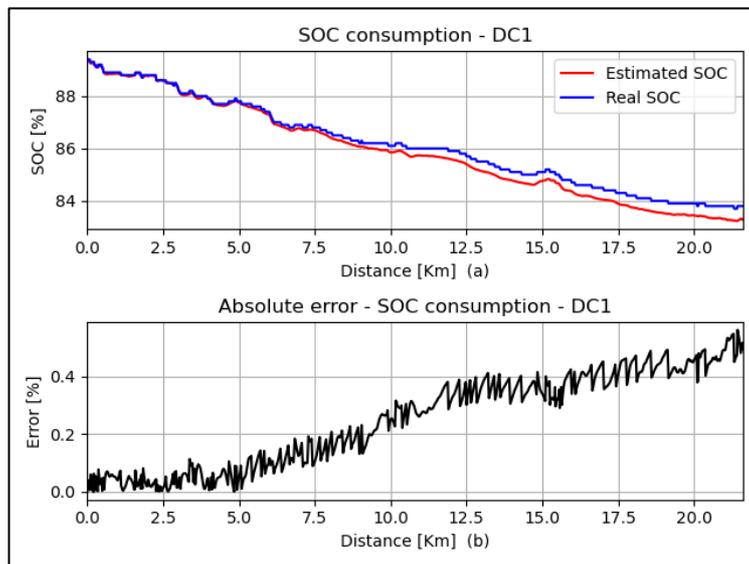


Figure 130. Comparison between real and estimated SOC consumption (a) and the associated absolute error (b), relative to DC1.

	Real Measurements	Estimated Measurements
Energy consumption1	4.37 [kWh]	4.58 [kWh]
Energy consumption2	20.22 [kWh/100Km]	21.18 [kWh/100Km]
SOC final value	83.8 [%]	83.29 [%]

Table 19. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC1.

Apart from these plots, also the RMSE and the MAE values were computed to evaluate the algorithm performances in the estimation of these three quantities: the battery power, the energy and the SOC consumption, as shown the histogram in Figure 131.

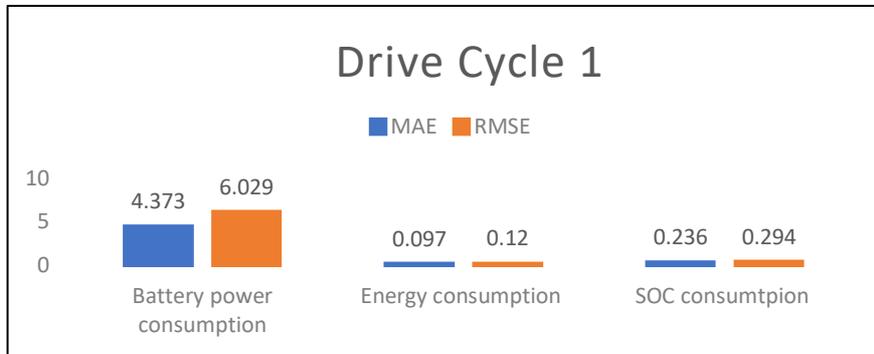


Figure 131. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC1.

According to the results found and to the fact that the absolute error computed was low, it can be concluded that the neural network model chosen worked well on the first dataset.

4.4.2 DRIVE CYCLE, DC2

The second route of the test set has a total length of about 10 Km, characterized by an elevation profile that grows in time, passing from an elevation of 275 m to 462 m above the sea level, so it is an uphill. The expected SOC consumption would be consistent. In fact, for only 10 Km, the initial SOC measured level is of 60.5 % and at the end of the trip is diminished to 55.7 %. Others measured data extracted from the dataset, associated to the DC2, are showed in Figure 132. Note that in Figure 132, the x-axis indicates the travel time, measured in seconds.

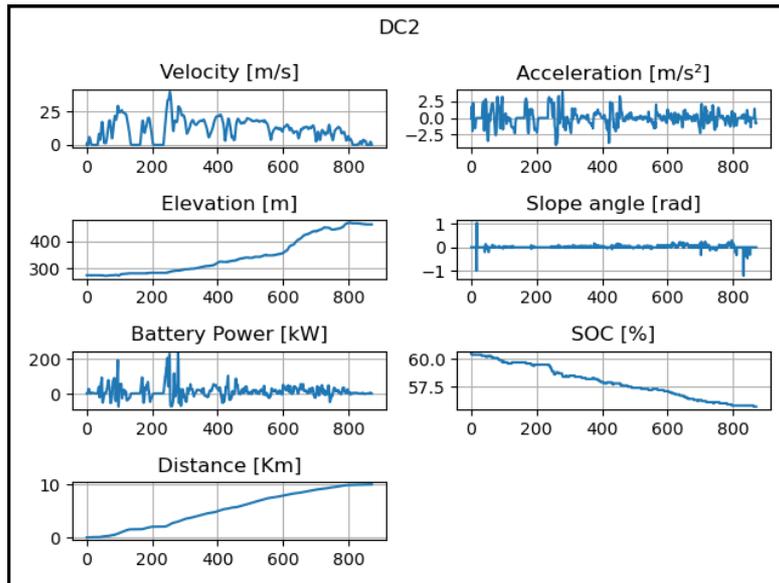


Figure 132. Drive cycle 2 data.

Figure 133 shown the estimated quantity that was obtained as output of the neural network. Instead, the estimated energy and SOC consumption are displayed in Figure 134 and Figure 135. In numerical terms, instead, the results were resumed in Table 20. In Figure 136, it is possible to have a view of the estimation performances obtained by the algorithm.

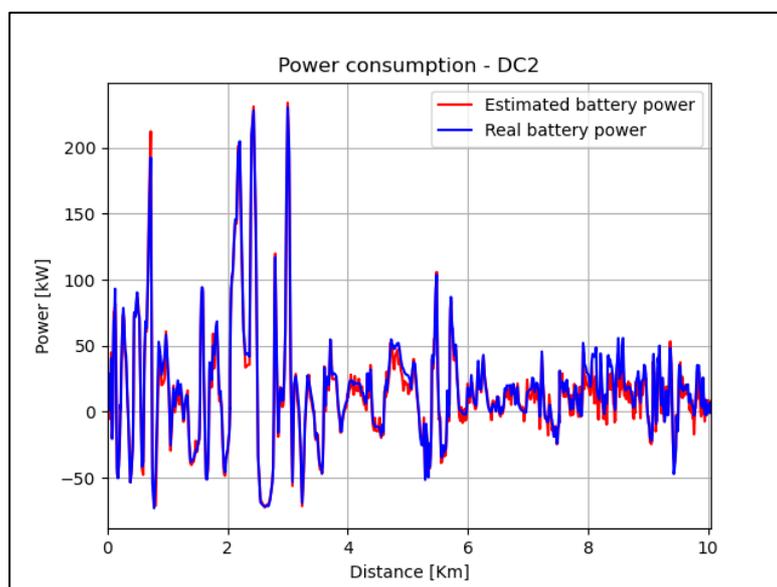


Figure 133. Power consumption associate to DC2.

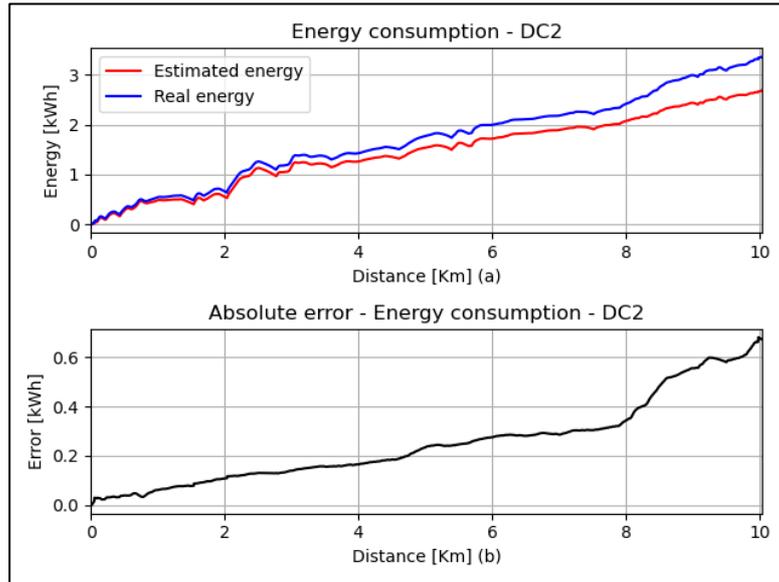


Figure 134. Energy consumption (a) and the associated absolute error (b), relative to DC2.

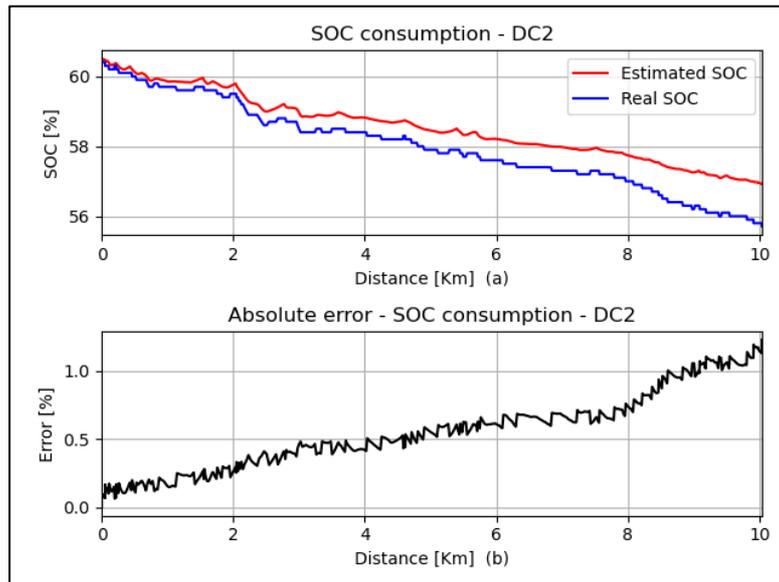


Figure 135. SOC consumption (a) and the associated absolute error (b), relative to DC2.

	Real Measurements	Estimated Measurements
Energy consumption1	3.36 [kWh]	2.68 [kWh]
Energy consumption2	33.45 [kWh/100Km]	26.73 [kWh/100Km]
SOC final value	55.7 [%]	56.91 [%]

Table 20. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC2.

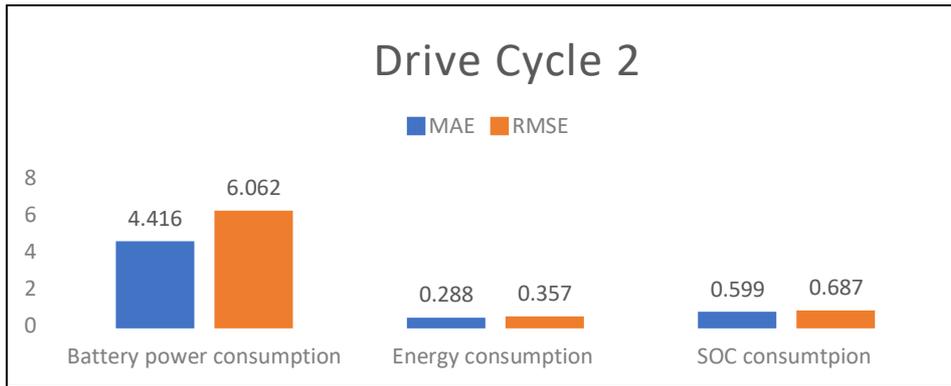


Figure 136. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC2.

Again, for the second test samples DC2, the results were good. The maximum absolute error reached in the estimation of the SOC value was about of the 1%.

4.4.3 DRIVE CYCLE, DC3

The third route used, called DC3, has a total length of 9.92 Km. From the elevation profile it can be deduced that the route was a downhill, so it is expected to not have a huge SOC consumption. The elevation at the beginning was 462 m and, with an excursion of -187 m, the route ends at 275 m above the sea level. The measured SOC varies from 56.9 % to 55.4 %. A series of measured data extracted from the dataset, associated to the DC3, are showed in Figure 137. The same analysis done for the two previous test drive cycles was made here.

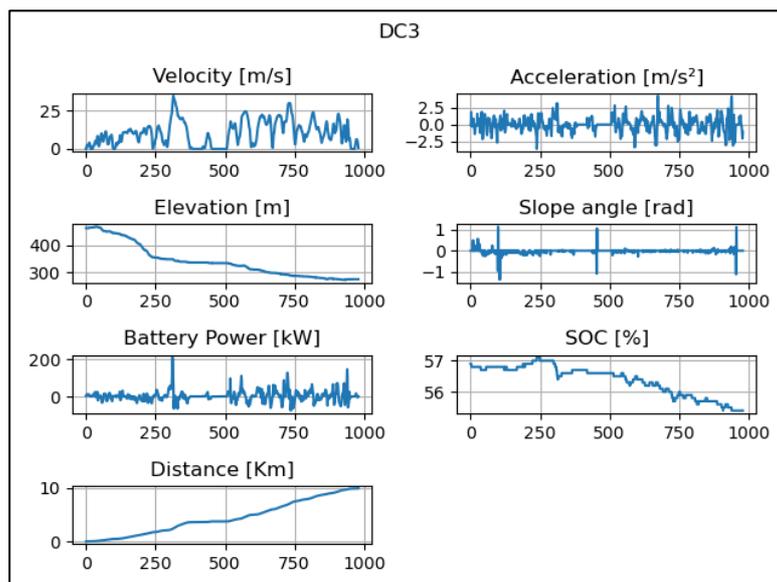


Figure 137. Drive cycle 3 data.

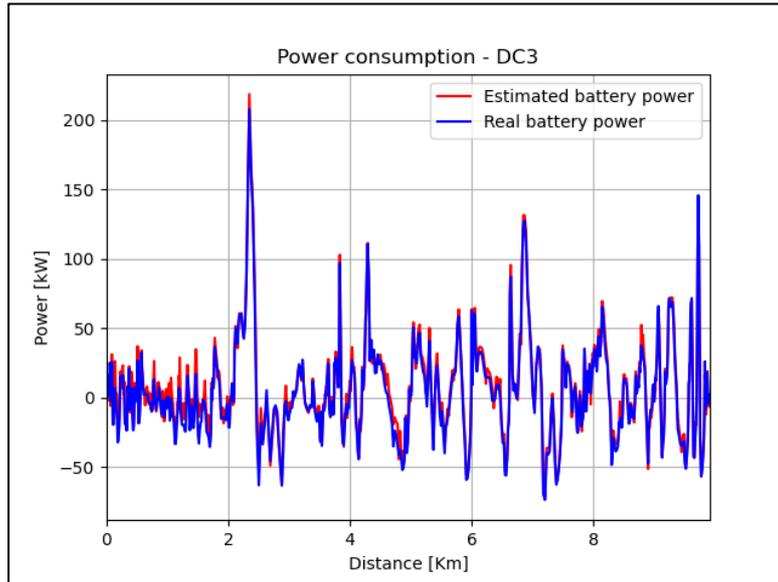


Figure 138. Power consumption associate to DC3.

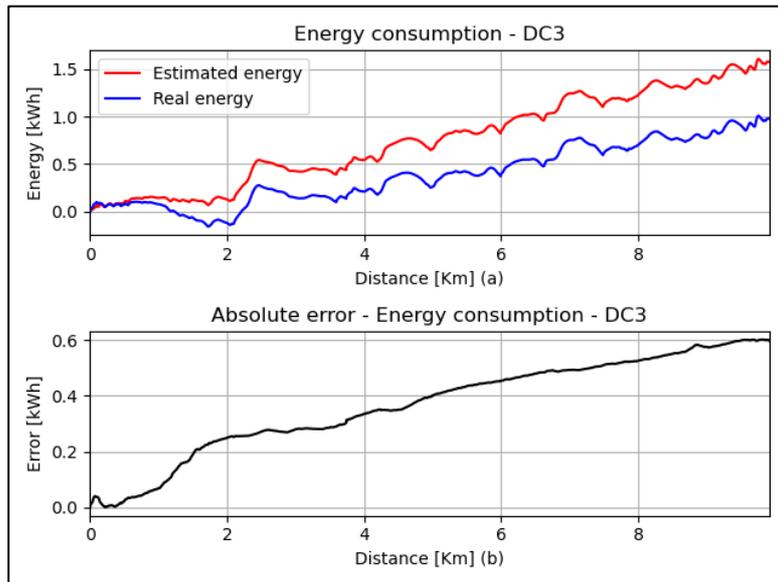


Figure 139. Energy consumption (a) and the associated absolute error (b), relative to DC3.

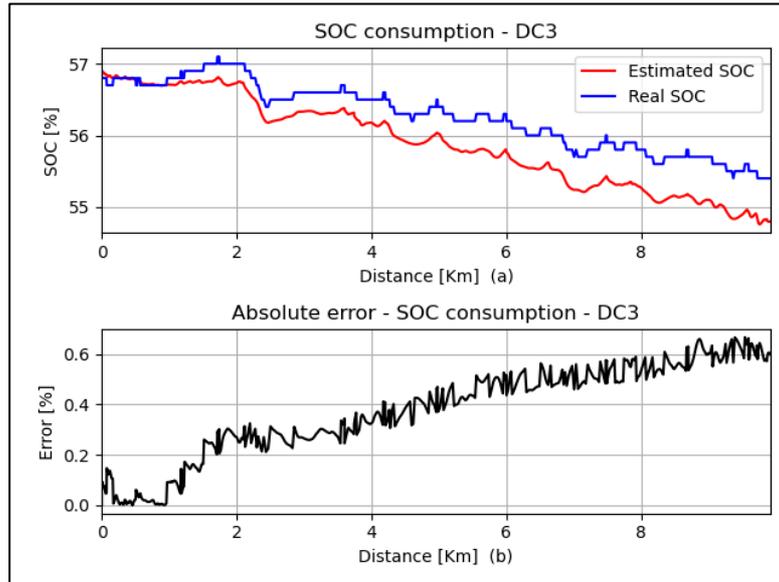


Figure 140. SOC consumption (a) and the associated absolute error (b), relative to DC3.

	Real Measurements	Estimated Measurements
Energy consumption1	0.98 [kWh]	1.57 [kWh]
Energy consumption2	9.86 [kWh/100Km]	15.87 [kWh/100Km]
SOC final value	55.4 [%]	54.80 [%]

Table 21. Resume of the real and estimated values obtained in terms of energy consumption and SOC final value for DC3.

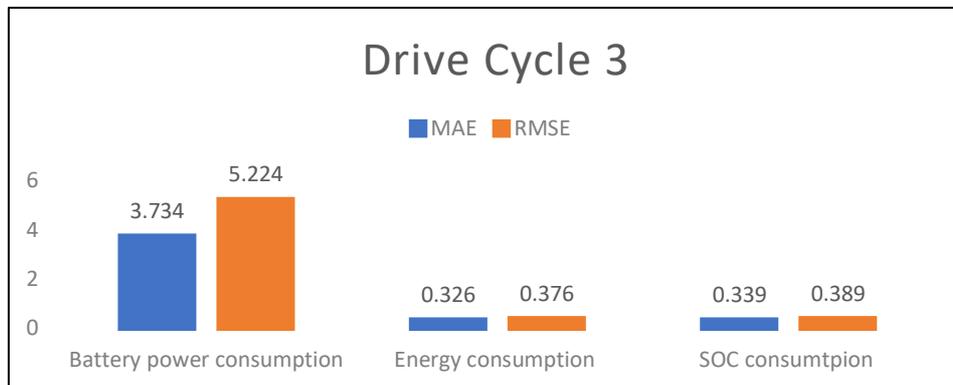


Figure 141. Metrics performances for estimation of battery power, energy and SOC consumption, related to DC3.

For this third test, DC3, the obtained results were aligned with the previous ones obtained evaluating the drive cycles DC1 and DC2. The maximum absolute error reached in the estimation of the SOC value was lower the 0.7%.

4.4.4 STATISTICAL ANALYSIS

After tested the ANN algorithm with only three routes, it was performed a similar analysis for 22 different routes that was possible to extract from the testing dataset. Focalizing only on the results relative to algorithm output and so, to the SOC, in order to check the estimation accuracy of the algorithm, the RMSE and MAE were computed. Considering as data the real SOC values and the estimated ones, the obtained results are showed in the histogram in Figure 142.

How it can be seen from this figure, all the values (both for RMSE and MAE) are low than 0.8, and this is a good indicator of the generalized behavior of the algorithm.

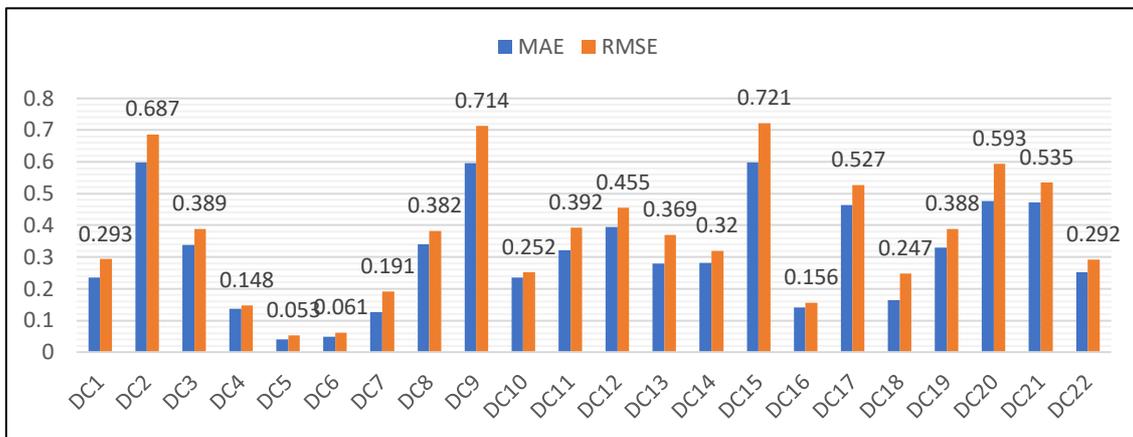


Figure 142. Metrics performances for estimation of SOC consumption, related to all the 22 drive cycles, used for testing.

It was also interesting to make a bit of analysis of the absolute error computed between the real and the estimated SOC consumption, $|e| = |\widehat{SOC} - SOC|$.

In Figure 143, it is shown the evolution of the absolute error over the time, for all the tested routes. The global trend of the error is to increase over the time, and this phenomenon was expected, due to the presence of an integrator function in the algorithm. In some cases, the error starts with a plateau and then increases, but this could be associated to the goodness of the network in the battery power consumption estimation process.

The integrator introduces a cumulative error, that is intrinsic, in physical sense, in the solution found, and cannot be removed, unless of changing the algorithm working process.

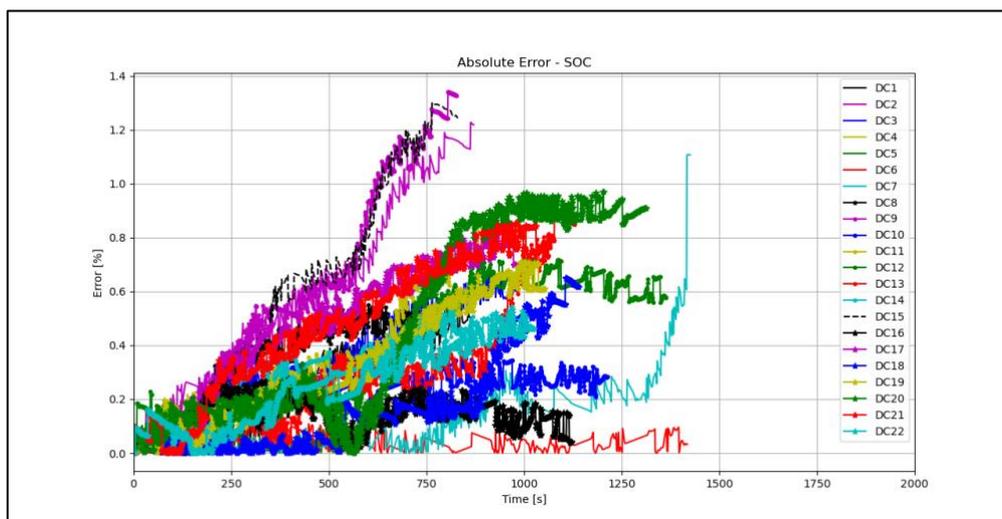


Figure 143. Absolute error relatives to the SOC computation over the time, for all the 22 drive cycles used for testing.

In addition, for each instant of time, the mean and standard deviation values, associated to the absolute error " e " of all the drive cycle tests used, were computed. In this way it was possible to obtain a general view of the mean error (\pm the standard deviation) behavior committed by the algorithm, in the SOC estimation, over the time. How it can be seen from Figure 144, both the mean and the standard deviation values associated to the error tend to increase over time. This is an effect of the integration function implemented in the algorithm and, as said before, it cannot be solved in any way. So, the bad point of the integrator presence can be observed from the occurrence of an error that increases over the time. This is due to the fact that the developed neural network, instead of estimating directly the SOC, estimates the battery power consumption and then, integrating this quantity, derives the energy and thus, the associated SOC.

Another important point to discuss is why the neural network does not directly estimate the SOC. By several tests done during the design phase of the algorithm, it was seen that the obtained results were bad: the estimated SOC values were deviated a lot from the real ones and moreover, were characterized by a lot of noises and oscillations.

A possible theoretical explanation as to why these poor results were achieved could be attributed to the non-observability of the SOC variable, in a problem like this; and also in this case, this is a direct consequence of the integrator function presence. For this reason, possible future research will be carried out in order to realize a theoretical study on the observability of the SOC variable in this system, trying to demonstrate why such poor results were obtained.

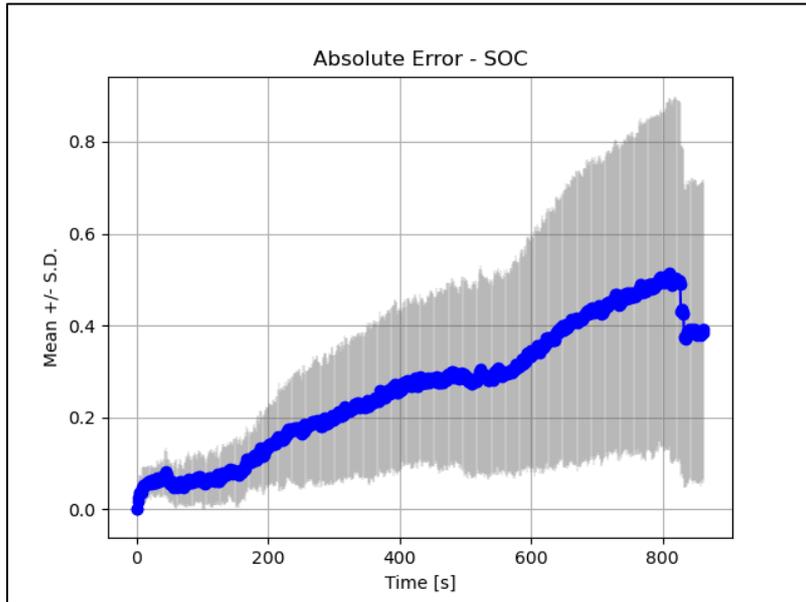


Figure 144. Plot with mean (blue) and standard deviation (gray) values relative to the absolute SOC error over time, computed considering all the 22 routes for testing.

After, it was created a vector containing the maximum absolute error computed by each route used for testing, in the total period of time and, another one, for the final absolute error value. From these, the mean and the standard deviation values were obtained as showed in Table 22. The chart with the results is shown in Figure 145.

	Final absolute error	Maximum absolute error
Mean [%]	0.631	0.703
Standard deviation (S.D.)	0.376	0.344

Table 22. Resume of mean and standard deviation values relative to final and maximum absolute error value for SOC, computed considering the 22 routes for testing.

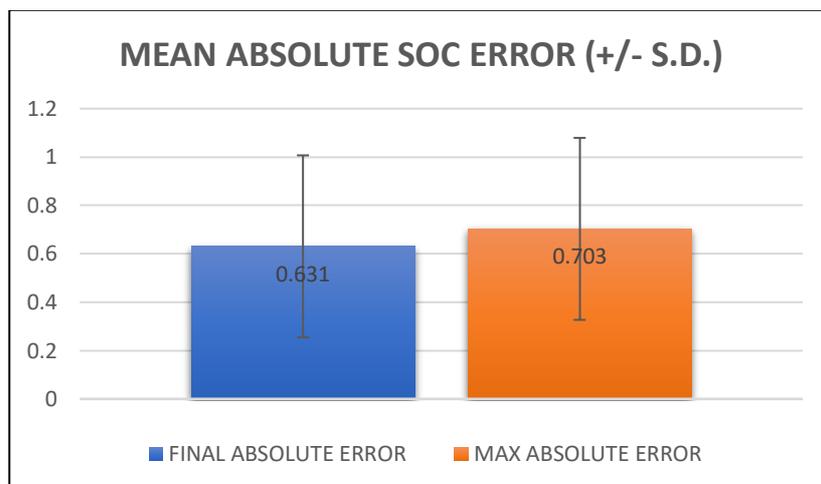


Figure 145. Mean value with standard deviation, related to the final and maximum absolute error value, associated to the SOC quantity, computed considering the 22 routes for testing.

4.5 CONCLUSIONS

The designed ANN algorithm is a novel way to solve the offline estimation of the state of charge problem, for a fully electric vehicle, including a neural network model. The inputs of the algorithm were: the velocity, the acceleration, and the slope road angle. Focalizing to the neural network implemented in the ANN algorithm, it is based on a feed-forward and, fully connected, architecture, with 3 hidden layers each containing 32 neurons. The output estimated by the network is the battery power consumption. This quantity is then integrated, to obtain the associated energy and so, the SOC level of consumption is derived. After a first evaluation of the results, obtained testing the algorithm with different drive cycles, extracted from the dataset, it was possible to conclude that the developed ANN model works well.

The absolute error computed by the model for SOC estimation was 0.703 [%], with a standard deviation of 0.34. The expected result could be better, and a possible reason can be attributed to the fact that the estimated variable by the neural network was not the SOC, but the battery power.

Obviously, also this second algorithm can be used in real life. The only difference with respect to the operations done before is that the input data have to be predicted, by using the same exactly functions implemented for algorithm 1, as described in chapter 3. In particular the functions that will be employed are: the one to obtain the elevation profile, that for velocity prediction profile, and for the travel distance. All these, need as input only the geographic coordinates of the route's points.

COMPARISON OF THE OBTAINED RESULTS AND CONCLUSIONS

In this last chapter of the work the differences between the results achieved by the two algorithms, analyzing both a single selected drive cycle and all the drive cycles together, will be showed. The estimation of the energy consumption and, obviously, of the state of charge will be compared, both in plots and in numerical terms. The discussion is divided into two paragraphs: the first is dedicated to the single drive cycle analysis, while, the second one to the whole set of drive cycles. In the following paragraphs, the model-based algorithm was indicated with algorithm 1, or ALG1, instead, the ANN algorithm with algorithm 2, or ALG2. The chapter ends with the work's conclusions and recommended future developments.

5.1 DRIVE CYCLE 1

As just said many times, in the previous two chapters, this drive cycle was used for testing processes and to compare the results obtained by the two developed algorithms. Velocity profile, acceleration profile and slope angle road profile, relative to the DC1 are showed in Figure 146. In addition, consumption values of the battery power and state of charge, acquired from the Tesla model 3 with a 75 kWh battery pack, are showed too. In addition, only for the model-based algorithm, also the weather data were used as entry. These data were obtained as response from an API, called OpenWeatherMap.

Without going into the details of the formulations and functions used to achieve the final estimated values, these are computed and compared. The estimated energy consumption, associated to the two algorithms, and compared with respect to the real one, is shown in Figure 147. Instead, the state of charge level of consumption is displayed in Figure 148.

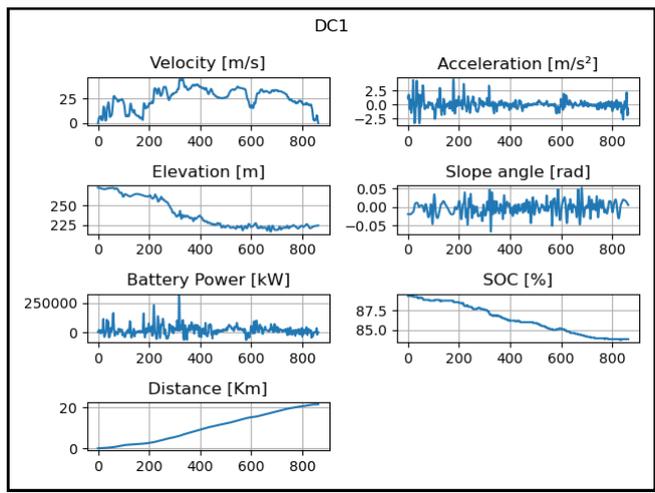


Figure 146. Main information about the DC1.

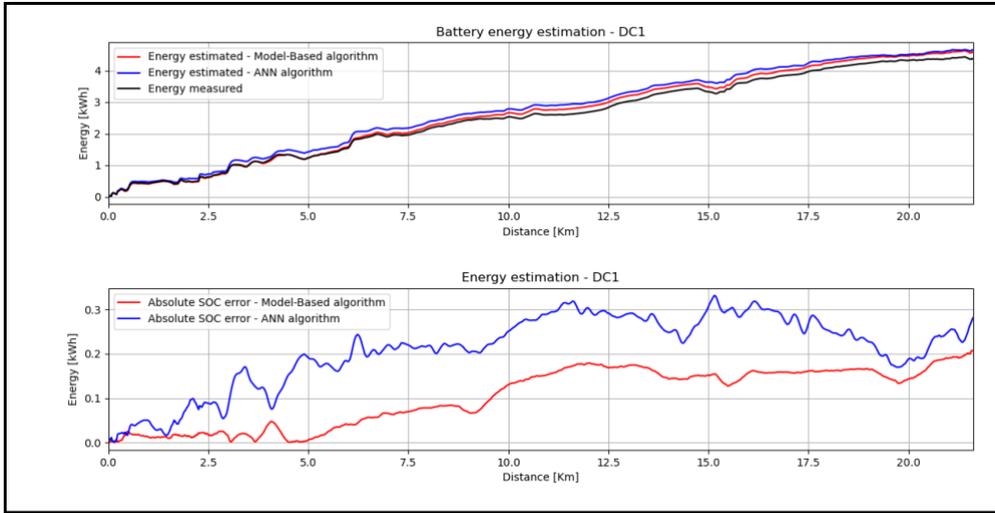


Figure 147. Battery energy consumption associated to DC1.

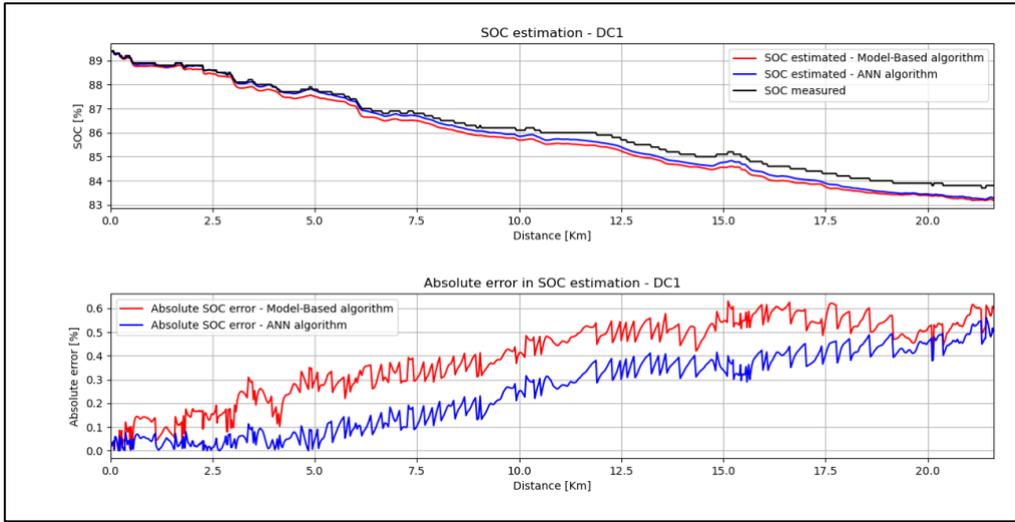


Figure 148. SOC consumption associated to DC1.

The consumption values, associated to the energy and state of charge are showed, in numerical terms, in Table 23. Comparing the results can be seen that both the algorithms worked fine and returned an estimated SOC level of consumption very close to the real one. The same is true for the energy. The absolute error committed by the two algorithms, in SOC estimation, is less that 0.7%.

	Real Measurements	Estimated Measurements ALG1	Estimated Measurements ALG2
Energy consumption1	4.37 [kWh]	4.65 [kWh]	4.58 [kWh]
Energy consumption2	20.22 [kWh/100Km]	21.52 [kWh/100Km]	21.18 [kWh/100Km]
SOC final value	83.8 [%]	83.2 [%]	83.29 [%]

Table 23. Comparison of the real and estimated values, obtained by the two algorithms, in terms of energy and SOC consumption, for DC1.

5.2 COMPARISON ON 22 DRIVE CYCLES SET

In this paragraph, a direct comparison of the results derived from the testing process of the two algorithms in the set of 22 different drive cycles will be made. In particular, the data here showed are taken from the previous two chapters and put together, to make a graphical matching. Precisely, the comparison is made in terms of algorithms' accuracy in the estimation of the SOC quantity, as can be seen from Figure 149. The metric chosen, among the MAE and RMSE, was the RMSE.

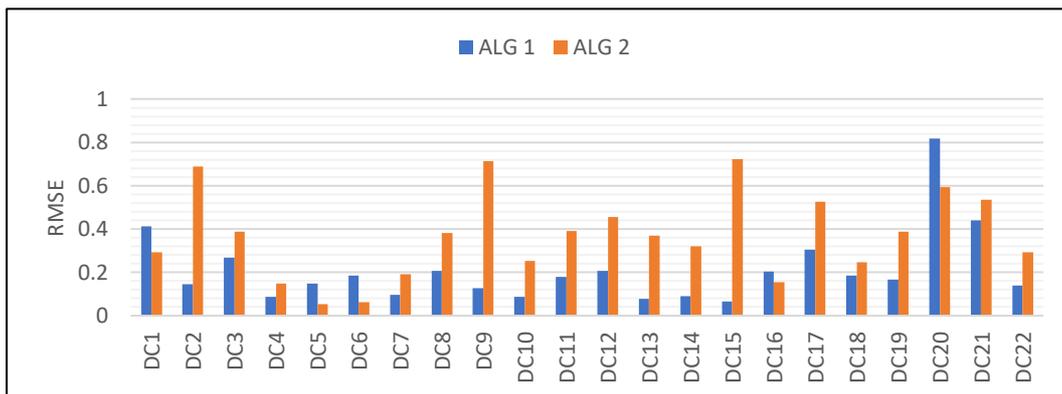


Figure 149. RMSE values associated to the SOC estimation, computed on 22 drive cycles, by the two algorithms. ALG1 = model-based algorithm. ALG2 = ANN algorithm.

Moreover, the absolute error computed by the algorithms in estimation of the SOC relative to the 22 drive cycles was calculated, and this was used to derive some interesting data for performing a statistical analysis. For example, the final and the maximum absolute errors computed by both the algorithms were measured and used to compute

the mean and the standard deviation. The values obtained are shown in Table 24 and, graphically, are illustrated in Figure 150.

	Final absolute error	Maximum absolute error
Algorithm 1		
Mean [%]	0.369	0.436
Standard deviation (S.D.)	0.266	0.261
Algorithm 2		
Mean [%]	0.631	0.703
Standard deviation (S.D.)	0.376	0.344

Table 24. Final and maximum absolute error, in terms of mean and standard deviation, computed for SOC estimation by both the algorithms on 22 drive cycles.

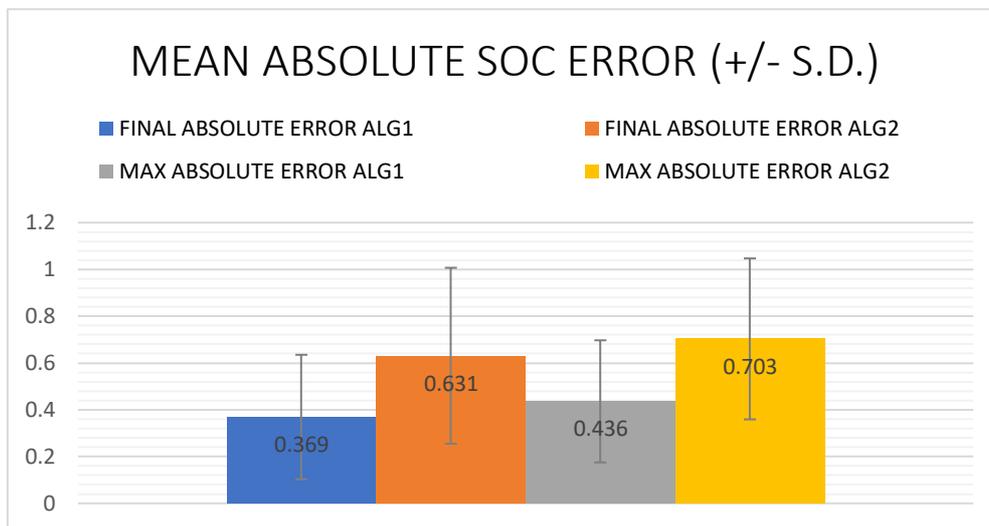


Figure 150. Comparison of mean absolute errors with S.D. for the SOC estimation on 22 drive cycles, relative to the algorithms developed. ALG1 = Model-Based algorithm. ALG2 = ANN algorithm.

As it can be figured out from the Figure 149 and Figure 150, the model-based algorithm estimates the SOC consumption a bit better than the ANN algorithm. Obviously, the data-driven model was derived from the availability of a good dataset, also in dimensional terms, to use; however, the possibility of having more real measured data to use for training, and even testing would help to achieve better results. Nevertheless, both the algorithms can be used as starting point for future developments in this direction.

5.3 THESIS CONCLUSIONS AND FUTURE DEVELOPMENTS

This thesis work presents two algorithms for the estimation of the state of charge in a fully electric vehicle. The strategies followed were different: one is based on a model-based approach, the other one, on a data-driven technique.

The first algorithm was based on the second Newton's law applied to the longitudinal vehicle dynamics equation, that allows to evaluate the energy consumption. Knowing the energy consumption, the SOC was derived.

The second algorithm, indicated as ANN algorithm, instead, follows a data-driven method, made possible by the availability of a dataset that contains thousands of data, acquired by a Tesla model 3 with a 75 kWh battery pack.

The algorithms were both tested on a set of 22 drive cycles in order to assess, in terms of accuracy, the SOC estimation process. As seen in the previous chapters, the two developed algorithms are able to return a good estimation of the discharge profile of the SOC. The final absolute errors, achieved by the two algorithms, after being tested with 22 different drive cycles, were: $0.369\% \pm 0.266$ for the model-based algorithm and $0.631\% \pm 0.376$ for the ANN algorithm. Then, the accuracy of the two algorithms, in the SOC estimation, expressed in RMSE is below than 0.85, and this is an indicator of the goodness of the algorithms. Among these two algorithms, the better seems to be the first one. A possible reason could be that the neural network does not estimate the SOC directly, but the battery power consumption that, once integrated, allows to obtain the SOC.

Despite these algorithms have achieved good results in the state of charge estimation, several future improvements can be realized. In this view, about the first algorithm, future studies could be realized to model the several systems presented in the EV's architecture (the electric motor, the inverter, the battery pack and so on). This aspect could be important for the estimation purposes, since it gives the possibility to calculate the power consumption from the battery side to the wheel side more precisely than was made in this work. Here, in fact, the conversion of power was realized using efficiency conversion coefficients η and obviously, this had involved a loss of accuracy.

Future developments on the second algorithm could be geared towards improving of the neural network 's overall performances. These could be achieved, for example, by using a larger training dataset. However, a more important result could be reached by realizing a theoretical study about the observability of the SOC variable, with the aim to explain why, with the inputs used in this work (velocity, acceleration, and road slope angle), it was not possible to realize a neural network that directly provides a fine and clear estimation of the state of charge consumption.

In addition, the first algorithm was also tested in a real scenario, thus simulating what the user would do to have an estimate of SOC consumption associated with a selected route. The results achieved by the algorithm 1, in its real application scenario, are described, in detail, in paragraph 3.8 of chapter 3.

Once again, it is necessary to repeat how, in a real application scenario, the input data used have to be predicted, before the journey begins. Among these data, the most arduous to predict is the velocity profile, which, by the way, is also the one that most influences the final result. This data, fundamental input in both the algorithms, is predicted using only information associated to the type of road and weather data; however, better description on how it is predicted is presented in paragraph 3.2.4 of chapter 3.

Among the possible future developments of this work, in this direction, there may be the improvement of the function designed to predict the speed profile. In order to have more accurately results, it could take into account other factors such as traffic conditions, the presence of traffic lights, stop signals, or roundabouts. A more complex point to achieve is the prediction of the driving style of the driver, that is maybe the most critical part, but also the one that mostly influences the consumption of energy. Bad driving style, characterized by sudden and severe accelerations, or decelerations, and by high speeds, is associated to major consumption. Vice versa, a smooth driving style, will lead to less energy consumption. This result could be achieved by considering more driving paths, made by several drivers that drive with different styles; then for the data elaboration machine learning techniques could be used.

Another possible point to develop is the realization of a GUI (Graphical User Interface) that gives to the user the possibility to set a route and automatically to provide the SOC level of consumption.

So, the contribution of this work is to provide two different offline algorithms for the SOC estimation of a BEV, that can be used both as a baseline for future works, but also for the development of a practical real application.

REFERENCES

- [1] MG MOTOR UK LTD, "The history of electric vehicles a timeline." [Online]. Available: <https://mg.co.uk/behind-the-wheel/electric/the-history-of-electric-vehicles-a-timeline/>.
- [2] U.S. DEPARTMENT OF ENERGY, "History electric car." [Online]. Available: <https://www.energy.gov/articles/history-electric-car>.
- [3] W. Contributors, "Gaston Planté," Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Gaston_Planté&oldid=1029418600.
- [4] F. L.-P. Environmental Protection Agency, "General Motors Urban Electric Car gets a battery charge from an outlet in the parking lot at the first symposium on low pollution power systems development held at the Marriott Motor Inn, Ann Arbor, Michigan.," The U.S. National Archives and Records Administration. https://commons.wikimedia.org/wiki/File:1973_GM_electric_car.png.
- [5] Alvolante, "Toyota Prius Image." <https://www.alvolante.it/news/toyota-prius-prima-ibrida-compie-20-anni-353813>.
- [6] Tesla, "Tesla model X." [Online]. Available: https://www.tesla.com/it_IT/blog/tesla-model-x-5-star-safety-rating.
- [7] J. Dedek, T. Docekal, S. Ozana, and T. Sikora, "BEV Remaining Range Estimation Based on Modern Control Theory - Initial Study," IFAC-PapersOnLine, vol. 52, no. 27, pp. 86–91, 2019, doi: 10.1016/j.ifacol.2019.12.738.
- [8] ACEA, "Car emissions testing facts." 2016, [Online]. Available: <https://www.caremissionstestingfacts.eu/#>.
- [9] A. S. and F. M. Bogdan Ovidiu Varga, "Prediction of Electric Vehicle Range: A Comprehensive Review of Current Issues and Challenges." 2019.
- [10] Energit, "Quante auto elettriche ci sono nel mondo." <https://energit.it/quante-auto-elettriche-ci-sono-nel-mondo/>.
- [11] Il sole 24 ore, "Ecco cosa rallenta diffusione dell' auto elettrica in europa." [Online]. Available: <https://www.ilssole24ore.com/art/ecco-cosa-rallenta-diffusione-dell-auto-elettrica-europa-AERFYIME>.
- [12] W.-Y. Chang, "The State of Charge Estimating Methods for Battery: A Review," ISRN Appl. Math., vol. 2013, no. 1, pp. 1–7, 2013, doi: 10.1155/2013/953792.
- [13] Z. Cai, G. Liu, and J. Luo, "Research state of charge estimation tactics of nickel-hydrogen battery," Proc. - 2010 Int. Symp. Intell. Inf. Process. Trust. Comput. IPTC 2010, pp. 184–187, 2010, doi: 10.1109/IPTC.2010.91.
- [14] "effectiveness-comparison-of-range-estimator-for-battery-electric-vehicles-2167-7670-1000128.pdf."
- [15] Tesla, "SUPERCHARGER TESLA." [Online]. Available: https://www.tesla.com/it_IT/supercharger.
- [16] Enelx, "ENELX JUICEBOX." [Online]. Available: <https://www.enelx.com/ar/en/electric-mobility/products/companies/juicebox>.
- [17] C. Li, F. Xiao, and Y. Fan, "An approach to state of charge estimation of lithium-ion batteries based on recurrent neural networks with gated recurrent unit," Energies, vol. 12, no. 9, 2019, doi: 10.3390/en12091592.
- [18] A. Massing, "Hamilton Kerr Bull 1 Orazio Gentileschi.pdf," Analog devices, 2017, [Online]. Available: <http://www.analog.com/media/en/technical-documentation/technical-articles/A-Closer-Look-at-State-Of-Charge-and-State-Health-Estimation-Techniques-....pdf>.
- [19] "POWERTECHSYSTEMS." [Online]. Available: <https://www.powertechsystems.eu/home/tech-corner/lithium-ion-state-of-charge-soc-measurement/>.

- [20] "Lead-Acid example." https://upload.wikimedia.org/wikipedia/commons/7/79/Lead-acid_voltage_vs_SOC.PNG.
- [21] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. 1998.
- [22] R. Bustos, A. R. M. Siddique, T. Cheema, S. A. Gadsden, and S. Mahmud, "State Of Charge And Parameter Estimation Of Electric Vehicle Batteries," 2018, doi: 10.25071/10315/35324.
- [23] S. Bittanti and P. E. Bologna, "Teoria della Predizione."
- [24] M. Taragna, "KALMAN FILTER THEORY Kalman filtering problem," 2011.
- [25] M. I. Ribeiro, "Kalman and Extended Kalman Filters : Concept , Derivation and Properties," *Inst. Syst. Robot. Lisboa Port.*, no. February, p. 42, 2004, [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.5088&rep=rep1&type=pdf>.
- [26] Y. Boujouadar, H. Elmoussaoui, and T. Lamhamdi, "Lithium-ion batteries modeling and state of charge estimation using artificial neural network," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 5, pp. 3415–3422, 2019, doi: 10.11591/ijece.v9i5.pp3415-3422.
- [27] J. Wang, I. Besselink, and H. Nijmeijer, "Battery electric vehicle energy consumption prediction for a trip based on route information," *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 232, no. 11, pp. 1528–1542, 2018, doi: 10.1177/0954407017729938.
- [28] C. J. J. Beckers, M. Paroha, I. J. M. Besselink, and H. Nijmeijer, "A Microscopic Energy Consumption Prediction Tool for Fully Electric Delivery Vans," *33rd World Electr. Veh. Symp. Expo. Peer Rev. Conf. Pap.*, pp. 1–12, 2020, doi: 10.5281/zenodo.4023302.
- [29] W. Contributors, "GPS Exchange Format," *Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/wiki/GPS_Exchange_Format.
- [30] W. Contributors, "GPS Exchange Format," *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/GPS_Exchange_Format.
- [31] "Project OSRM." <http://project-osrm.org/>.
- [32] "OpenStreetMap." <https://www.openstreetmap.org/about>.
- [33] "OSM NODES." <https://wiki.openstreetmap.org/wiki/Node>.
- [34] "OSM WAY." <https://wiki.openstreetmap.org/wiki/Way>.
- [35] "Overpass API." https://wiki.openstreetmap.org/wiki/Overpass_API.
- [36] "Limiti di velocità." <https://www.poliziadistato.it/articolo/limiti-di-velocita>.
- [37] S. J. et al. Harris, C.R., Millman, K.J., van der Walt, *Array programming with NumPy*, vol. 585. Springer Science and Business Media, 2020.
- [38] "Calculate distance, bearing and more between Latitude/Longitude points." <https://www.movable-type.co.uk/scripts/latlong.html>.
- [39] "Shuttle radar topography mission." <https://www2.jpl.nasa.gov/srtm/>.
- [40] T. Krajina, "Python module srtm.py." <https://github.com/tkrajina/srtm.py>.
- [41] W. Contributors, "Butterworth filter," *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Butterworth_filter&oldid=1052086135.
- [42] "SciPy." <https://www.scipy.org/>.
- [43] Scipy, "Butterworth filter in SciPy." <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html>.
- [44] "OpenWeather." <https://openweathermap.org/>.
- [45] National Programme On Technology Enhanced Learning, "Introduction to Hybrid and Electric Vehicles," pp. 1–28, 2013.
- [46] D. Baek, Y. Chen, A. Bocca, S. Di Cataldo, and N. Chang, "Estimation of the residual energy in battery electric vehicles," 2019 AEIT Int. Conf. Electr. Electron. Technol. Automotive, AEIT Automot. 2019, 2019, doi: 10.23919/EETA.2019.8804523.
- [47] "Tesla Regen System." https://www.tesla.com/it_IT/blog/magic-tesla-roadster-regenerative-braking.
- [48] W. Contributors, "Energy conversion efficiency," *Wikipedia, The Free Encyclopedia*. .

- [49] P. Rb and R. Universitario, "Politecnico di Torino Politecnico di Torino," p. 87316161, 2020.
- [50] Wikipedia contributors, "Tesla Model 3," Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Tesla_Model_3&oldid=1045247745.
- [51] W. Contributors, "Density of air," Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Density_of_air&oldid=1040808753.
- [52] Baker; Schlatter, "Algorithms, Comparisons and Source References by Schlatter and Baker," 1991. https://wahiduddin.net/calc/density_algorithms.htm.
- [53] S. E. Richard Shelquist, "An Introduction to Air Density and Density Altitude Calculations." https://wahiduddin.net/calc/density_altitude.htm.
- [54] "Compass diagram." Geography, Map, Compass, Rose, Plot, Travel, World - Wind Rose With Angles, HD Png Download , Transparent Png Image - PNGitem.
- [55] "MeteoBlue." <https://www.meteoblue.com/>.
- [56] A. B. de Lima, M. B. C. Salles, and J. R. Cardoso, "State-of-Charge Estimation of a Li-Ion Battery using Deep Forward Neural Networks," 2020, [Online]. Available: <http://arxiv.org/abs/2009.09543>.
- [57] C. Engineering, "AI vs Machine Learning." <https://ai.engineering.columbia.edu/ai-vs-machine-learning/>.
- [58] laptrinhx, "An introduction to machine learning interpretability amazing read for ml enthusiastic." [Online]. Available: <https://laptrinhx.com/an-introduction-to-machine-learning-interpretability-amazing-read-for-ml-enthusiastic-42753182/>.
- [59] ecyy, "Learning machine learning how to code without learning coding," medium. <https://ecyy.medium.com/learning-machine-learning-how-to-code-without-learning-coding-9fc4291902b>.
- [60] deepsense.ai, "what is reinforcement learning the complete guide." <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/>.
- [61] "Perceptrons the first neural networks." .
- [62] N. Biologici, N. Artificiali, and O. Backpropagation, "Reti Neurali Neuroni Biologici," pp. 1–37.
- [63] "coursera - machine learning." [Online]. Available: <https://www.coursera.org/learn/machine-learning>.
- [64] Brilliant, "Backpropagation." <https://brilliant.org/wiki/backpropagation/>.
- [65] Wikipedia contributors, "Derivate," Wikipedia, The Free Encyclopedia. <https://en.wikipedia.org/wiki/Derivative>.
- [66] V. Bushaev, "How do we train neural networks," towardsdatascience.
- [67] M. H. SAZLI, "A brief review of feed-forward neural networks murat h. sazli," vol. 50, no. 1, pp. 11–17, 2006.
- [68] I. G. and Y. B. and A. Courville, Deep Learning. MIT Press, 2016.
- [69] baeldung, "Learning curve ml." <https://www.baeldung.com/cs/learning-curve-ml>.
- [70] J. Chemama, "How to solve underfitting and overfitting data models," AllCloud. <https://allcloud.io/blog/how-to-solve-underfitting-and-overfitting-data-models/>.
- [71] Aditya Mishra, "metrics to evaluate your machine learning algorithm," Towards Data Science. <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.
- [72] A. G. Barnston, "Correspondence among the Correlation RMSE, and Heidke Forecast verification Measures; Refinement of the Heidke Score," Climate Analysis Center, 1992.
- [73] S. Sharma and S. Sharma, "ACTIVATION FUNCTIONS IN NEURAL NETWORKS," vol. 4, no. 12, pp. 310–316, 2020.
- [74] anaconda, "Anaconda." <https://docs.anaconda.com/anaconda/navigator/index.html>.
- [75] Spyder Website Contributors, "Spyder-IDE." <https://www.spyder-ide.org/>.
- [76] Keras, "Keras." <https://keras.io/>.

- [77] E. B. Martín Abadi, Ashish Agarwal, Paul Barham et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016, [Online]. Available: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>.
- [78] É. D. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011, [Online]. Available: <http://jmlr.org/papers/v12/pedregosa11a.html>.