Università Degli Studi di Modena e Reggio Emilia

DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"
Doctor of Philosophy in Industrial and Environmental Engineering
Cycle XXXVII

PhD Thesis in Applied Physics

Data-driven Methodology for Battery Health Estimation

Candidate:
Matteo Ermini

Advisor:
Chiar.Mo Prof. Enrico Stalio

Co-Advisor:
PhD. Federico Brusiani

Coordinator:
Chiar.Mo Prof. Alberto Muscio

Academic Year 2023-2024

A chi c'è sempre stato

Abstract

With the environmental pollution affecting global warming year-by-year, decarbonisation of the transportation sector is seen as a key factor to contrast the CO_2 emissions and fighting climate change. In this direction, regulations about CO_2 emissions are tightening year after year reducing the allowed emissions for the fleet of vehicles. These strict regulations push carmakers to invest in hybrid electric vehicles and full electric vehicles which makes use of a battery as an additional or primary source of power. Formula 1, is a motorsport category at the forefront of automotive engineering and has always reflected these trends in terms of performance, safety and sustainability. In line with this, in 2014 with a major regulation change, the regulatory entity introduced the hybrid power unit-a complex propulsion system where different components cooperate to propel the car. The power unit relies on a battery as an additional power source. In such a highly competitive context, where every millisecond can make a difference, it is extremely important to have a clear and precise idea about the expected performance of any component of the car, to reduce as much as possible the uncertainties during races. This is true also for the battery whose performance affects the power delivered to the tyres and thus the lap time.

The works presented in this thesis have to be collocated within this context. They involve two distinct approaches aimed at accurately estimating battery degradation. The first study focuses in estimating the SOH of a complete battery. The problem is faced by implementing a data-driven reduced order model (ROM) of a cell. This model predicts voltage based on input parameters such as current, SoC and temperature. To achieve this, a deep long short-term memory (LSTM) network with a many-to-many architecture is employed. The voltage predicted is then used to calculate the expected voltaic efficiency for a new battery-modelled as a series/parallel arrangement of cells-under known working conditions. The efficiency is then compared with the voltaic efficiency derived from real battery measurements under the same working profile. To quantify battery health, a new coefficient is introduced, defined as the difference between the two voltaic efficiencies. The methodology is then to three different real batteries, showing the effectiveness of the proposed approach. Applying this methodology to three real batteries, each operating under different conditions, reveals that battery degradation varies depending on usage patterns. Consequently, two batteries with the same mileage can exhibit different efficiency losses, highlighting the necessity of tailored SOH assessments.

The second study proposes a methodology to estimate the capacity of the cells composing the battery based on the measurement of the voltage

relaxation. The key innovation lies in triggering voltage relaxation using a current spike. The resulting curve is processed to extract four statistical indicators, identified in maximum, minimum, average and variance. It is shown that the proposed indicators are correlated to the actual cell capacity. Eventually, they are used to implement a machine learning model to estimate the capacity. To optimise the approach, three different current spike intensities are evaluated in order to find out which is the smallest value that can be applied to provide enough information properly describing the capacity degradation of the cell. Moreover, three different machine learning models are implemented in order to find out the best combination (model, current spike) which results in the lower RMSE and the best prediction accuracy. The models implemented are multivariate linear regression, gradient-boosting decision tree and random forest regression. The results show that the best model is the gradient-boosting decision tree with the highest current spike, it results in an RMSE of 0.0172 Ah over the training dataset and 0.0184 Ah over the test dataset, showing that the proposed approach is effective in estimating cell capacity with a low error.

Contents

1 Introduction **3**

2 Literature Review **9**
2.1 Experimental methods 10
2.2 Physics-based methods 14
2.3 Data-driven methods 16

3 Fundamentals of AI **21**
3.1 Artificial intelligence, Machine learning and Deep learning 21
3.1.1 Supervised Learning 24
3.1.2 Errors in function estimation 26
3.1.3 Linear regression 28
3.1.4 Ensemble methods 29
3.2 Deep learning 39
3.2.1 Technical notes of Neural Networks 39
3.2.2 Optimisers 47
3.2.3 Sequence modelling 50
3.2.4 Long Short-Term Memory 54

4 Voltaic Efficiency for battery health monitoring **57**
4.1 Introduction 57
4.2 Battery efficiency 58
4.3 Dataset 59
4.4 LSTM architecture 63
4.5 Battery health monitoring 65
4.5.1 ROM performance 65
4.5.2 Health monitoring 67

5 Data-driven battery degradation modelling **73**
5.1 Introduction 73
5.2 Dataset 74
5.3 Models definition & Training 78
5.4 Results 81

6 Conclusions **85**

List of Figures

1.1 Representative schematisation of the PU of a Formula 1 (F1) car completed with technical regulation about the energy capacity and energy flows 5

2.1 Ragone plot for different chemistries. Updated to include Li-ION chemistries 9
2.2 Example of a DV analysis which shows the effect of the capacity fade on the differential curves. 12
2.3 Schematic diagram of an EM model of a Lithium-Ion battery (LIB) . . 15
2.4 Schematic representation of an ECM of a LIB 15

3.1 Relationship between artificial intelligence, machine learning and deep learning 22
3.2 Schematic representation of the three main paradigms of machine learning: a) Supervised Learning, b) Unsupervised Learning and c) Reinforcement Learning 24
3.3 Example of how a dataset can be subdivided into training, validation and test subsets to properly train an ML model 26
3.4 Representation of the Bias, Variance and total error as a function of the model complexity 27
3.5 Representation of a simple univariate linear regression 29
3.6 General representation of a Decision Tree structure 30
3.7 Regression tree corresponding to the partition represented in Fig. 3.8 . 32
3.8 Representation of the partition made by DT in the bi-dimensional feature space. 33
3.9 Example of a boosted gradient for GBDT implementation. 36
3.10 Example of bagging for random forest implementation. 37
3.11 General scheme of a feed-forward neural network. This particular architecture uses one input layer $\in \mathbb{R}^3$, two hidden layers $\in \mathbb{R}^5$ and one output layer$\in \mathbb{R}^1$ 39
3.12 Plot of the ReLU 42
3.13 Plot of the sigmoid function 43
3.14 Plot of the hyperbolic tangent function 44
3.15 Example of the possibility of accepting local minimum points instead of global minima 47
3.16 Representation of a function with a highlighted saddle point 48
3.17 Schematic comparison between the SGD and the momentum algorithm 48
3.18 General scheme of a recurrent neural network 51
3.20 Representation of an LSTM cell 54

4.1 Representation of the different phases of dataset manipulation to properly arrange data and create the tensor. a) Representation of the dataset before data preprocessing b) Representation of the windowed dataset, each colour representing a different window c) Representation of the tensor to be fed to the LSTM. 62
4.2 Portion of 10 s representing the scaled cycle used to implement the cell model. a) Voltage vs time, b) Current vs time, c) state of charge (SoC) vs time, d) Temperature vs time. 64
4.3 Portion of 10 s representing the comparison between the ground truth and the model outcome over the training dataset. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time. 66
4.4 Portion of 10 s representing the comparison between the ground truth and the model outcome over the test dataset. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time. . 67
4.5 Comparison between the ground truth and the model outcome on a CC discharging cycle. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time. 68
4.6 Comparison between the ground truth and the model outcome on a CC discharging cycle. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time. 69
4.7 VE of the actual battery and voltaic efficiency (VE) calculated by predicting the voltage with the model and the difference between them. a) Results for battery number 1, b) Results for battery number 2, c) Results for battery number 3. 70

5.1 Linear correlation matrix of the features extracted from the voltage relaxation curves and the capacity. a) $I_s = 1$, b) $I_s = \frac{2}{3}$ and c) $I_s = \frac{1}{3}$. . . 76
5.2 Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = 1$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test. 77
5.3 Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = \frac{2}{3}$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test. 78
5.4 Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = \frac{1}{3}$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test. 79

5.5 Comparison between the ground truth and the prediction obtained with the GBDT model for the spike at $I = 1$ over the training dataset, Figure (a) capacity against the variance, (b) capacity against maximum, (c) capacity against minimum, (d) capacity against the mean. 82
5.6 Comparison between the ground truth and the prediction obtained with the GBDT model for the spike at $I = 1$ over the test dataset, Figure (a) capacity against the variance, (b) capacity against maximum, (c) capacity against minimum, (d) capacity against the mean. 83

List of Tables

4.1 Parameters defining the Adam optimiser 65
4.2 Model performance evaluation through RMSE calculated over the training, the test datasets and the two CC discharging cycles. 65

5.1 Hyperparameters used for the implementation of the LR 80
5.2 Hyperparameters used for the implementation of the GBDT 80
5.3 Hyperparameters used for the implementation of the Random Forest . . 80
5.4 RMSE obtained with each model over the train and test dataset 81

Acronyms

AC alternate current.

AI Artificial Intelligence.

BMS battery management system.

BOL beginning of life.

BPTT back-propagation through time.

CART Classification and Regression Tree.

CC constant current.

CU control unit.

CV constant voltage.

DA differential analysis.

DL Deep Learning.

DoD depth of discharge.

DT decision tree.

DV differential voltage.

DVA differential voltage analysis.

ECM equivalent circuit model.

EIS electrochemical impedance spectroscopy.

EM electrochemical model.

EOL end of life.

ERS energy recovery system.

ES energy storage.

F1 Formula 1.

FFNN Feedforward Neural Network.

FIA Fédération Internationale de l'Automobile.

GBDT Gradient Boosting Decision Tree.

GRU Gated Recurrent Unit.

IC incremental capacity.

ICA incremental capacity analysis.

ICE internal combustion engine.

ID3 Iterative Dichotomiser 3.

IG information gain.

IT information theory.

KERS Kinetic Energy Recovery System.

LIB Lithium-Ion battery.

LR linear regression.

LSTM Long Short-Term Memory.

MAE mean absolute error.

MAPE mean absolute percentage error.

MGU-H Motor Generator Unit - Heat.

MGU-K Motor Generator Unit - Kinetic.

ML Machine Learning.

MSE min squared error.

NDA Non-Disclosure Agreement.

NN neural network.

OCV open circuit voltage.

P2D pseudo-two dimensional.

PU Power Unit.

ReLU rectified linear unit.

RF Random Forest.

RMSE root mean squared error.

RNN Recurrent Neural Network.

ROM reduced order model.

RSS residual sum of squares.

SEI solid electrolyte interface.

SGD stochastic gradient descent.

SoC state of charge.

SOH state of health.

SVM Support Vector Machine.

TF TensorFlow.

VE voltaic efficiency.

VR voltage relaxation.

Chapter 1

Introduction

"La vittoria più bella è quella che deve ancora arrivare".
Enzo Ferrari

The citation is representative of the philosophy behind the development of each Ferrari F1 car. F1 is a highly competitive context where every millisecond can change the outcome of a race, determining the winning or the losing of a GP. This awareness drives the engineers in the scrupulous research of perfection in any detail, with the aim of pushing the boundaries of physical laws to reduce the lap time and crossing for first the line below the chequered flag.

Since its beginning in 1948, the F1 races have been regulated by Fédération Internationale de l'Automobile (FIA), an international organisation with two primary functions: the mobility division advocates the interest of the automotive industry, and the sport division is a governing body for many international motorsport championships, including F1. Over the years, FIA made several changes to properly regulate any part of the car including its heart: the propulsion system.

At the very beginning, the regulations were defined by the engine capacity, trying to bring a balance between supercharged and normally aspirated engines. In particular, the teams had the possibility to choose between a 1.5 ℓ supercharged and 4.5 ℓ naturally aspirated internal combustion engine (ICE). No limits were imposed on the weight of the car The changes in regulation that occurred over the years, explored different ICE architectures to reflect the evolution trends in automotive engineering driven by the pursuit of performance, efficiency, sustainability and vehicle safety improvement. At the beginning of the 1960s, a regulation review removed the possibility of using a supercharged engine in favour of the allowed 1.5 ℓ naturally aspirated one, and the minimum weight of 450 kg for a car was introduced. This decision was pushed by the necessity of reducing the speed and thus improving safety. In the late 60s, regulation changed again, giving the teams the possibility to choose between a 3.0 ℓ naturally aspirated engine or a 1.5 ℓ turbocharged, and the weight of the car was limited to a minimum of 500 kg. In the 1970s, to reduce fuel consumption, a change in regulation forced the teams to use a 1.5 ℓ turbocharged ICE which can produce significantly more power with the same displacement compared to the naturally aspirated version. The minimum weight of the car was set to 530 kg. In this decade, the first movements to improve the safety of the sport began, and the regulations made the wholesale changes needed to bring it up towards the modern standards of safety which it enjoys today. The

first step has been that of removing long circuits (such as Nürburgring) and shorter and more safety-compliant circuits (such as Paul Ricard or Hockenheimring) that were built with safety features installed. With the safety improvement implemented during the late 70s and early 80s, F1 became overall much safer despite deaths still happening. The huge amount of downforce created by the ground effect became too dangerous, and thus the technology was banned. Regulations changed again in 1989, when turbocharged engines were banned in favour of a naturally aspirated ICE with 8 or 12 cylinders and a capacity of 3.5 ℓ, subsequently limited, in 1995, to 3.0 ℓ. From the early 2000s, FIA opted for a shift toward improving the efficiency of the F1 vehicles, controlling its costs and improving their safety. Concerning safety, the most important introduction has been the head and neck support (HANS), anyway, no major changes have been made until the introduction of the halo. Concerning efficiency and cost reduction instead, a big step was made in 2009 with the introduction of the Kinetic Energy Recovery System (KERS): a system that converts the kinetic energy of the car during braking, into mechanical or electrical energy which is then stored in a flywheel or a battery respectively. The harvested energy is available to be deployed later during acceleration phases. The most significant change in the F1 propulsion system came in 2014, with the introduction of hybrid Power Unit (PU). The propulsion system becomes a highly complex device composed of a 1.6 ℓ ICE and a sophisticated energy recovery system (ERS), which harvests kinetic energy during braking as well as energy, in the form of enthalpy, from the waste gas duct converting it in electrical energy to be stored in a LIB used as energy storage (ES). LIBs are ideal for this application because of their high power and high energy density characteristics [1].

The technical regulation defines the PU as the assembly of ICE and turbocharger, complete with its ancillaries, any ERS system, all the actuation systems and the control electronics to make them work at any time. The same regulations define the ERS as a system specifically designed to recover energy from the car, store it, and make it available to propel the car itself when required [2]. As for all the aspects of the F1 car development, also the PU operation is bounded by a strict set of rules. In particular, FIA ruled the energy capacity and its flow. A representative picture of the layout of the PU system and its regulations, can be seen in Fig. 1.1. There, all the components of the PU are schematised and are reported the information about the allowed energy storage capacity and the allowed energy flows between the components. The picture highlights that it is possible to think of the PU as composed of three main subgroups: ICE and its related parts (represented with red squares), ERS and its control unit (represented via green squares) and non-ERS energy storage systems and their ancillaries (represented in blue). The ERS system, as can be verified from Fig. 1.1, is composed of:

- Energy Storage (ES);
- control unit (CU);
- Motor Generator Unit - Kinetic (MGU-K);
- Motor Generator Unit - Heat (MGU-H).

Diving into each component, most of the time, the ES is a LIB whose function is that of storing the electrical energy (in the form of chemical energy) and providing that

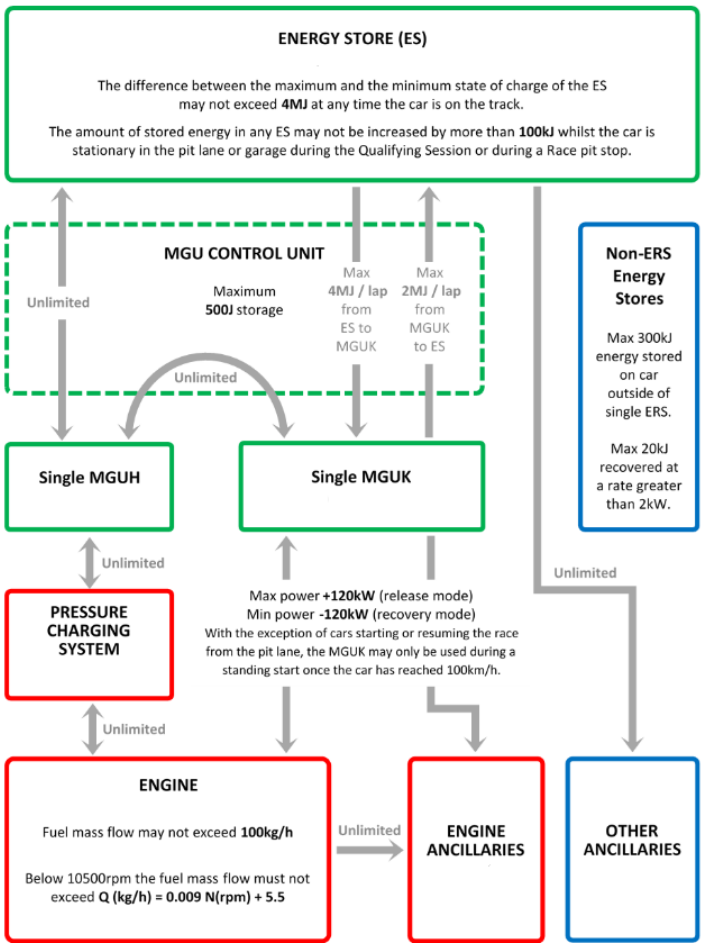


Figure 1.1: Representative schematisation of the PU of a F1 car completed with technical regulation about the energy capacity and energy flows

energy when required, such as in acceleration phases. The CU is the set of electronic devices which drives the electric motors and the current flow between them and the ES through the AC/DC conversion. The MGU-K is an electric motor paired to the ICE through one or more gears with a double purpose, on the one hand, it is used as a supplementary power supplier in acceleration phases, on the other hand, it works as a supplementary brake in braking phase, converting the car's kinetic energy in electrical energy which is stored in the ES. The MGU-H instead, is directly paired to the turbocharger through a coaxial shaft and has the double function of reducing the turbo lag driving the compressor when the engine works at low revolutions, and recovering energy from wasted gas by converting its enthalpy in electrical energy. The introduction of the ERS system marked a transformative era in the sport, aligning with broader trends in automotive technology and environmental sustainability. The presence of such strict regulations and the limited testing time forced by the FIA with the aim of cost reduction, pushes the teams to invest in developing high-fidelity virtual models of the car's components to be juxtaposed with the experiments. This way of working translates into reduced costs and environmental impact, in line with the philosophy behind the FIA goals.

Formula 1 is a highly competitive championship where everything is pushed to the

limit and stressed, this is true for the drivers who have to withstand extremely high g-forces and make decisions at speeds above 300 km/h, but also for the components which have to be designed to be as much performant as possible in terms of energy efficiency but also in terms of weight and resistance. This makes each component extremely efficient but, at the same time, it might happen that the sustained stresses degrade it, reducing its performance and reliability. In this context, it becomes of primary importance to monitor the health of components during their whole lifetime. This monitoring becomes more important for such components for which FIA limits the number during a season such as ES or MGUs. One of the most complex components to take care of is the ES. As stated before, most of the time, a LIB is used as ES, bringing all the complexity of such a system. Ageing of a battery can be thought of as given by the contribution of two main phenomena such as calendar ageing and cycling ageing. Calendar ageing occurs when the battery is at rest, with no current flowing. It is a function of the storing average SoC, time t and temperature T [3, 4]. Cycle ageing instead occurs during battery usage, and it is related to charge and discharge current I, temperature T, number of cycles n_{cycles} and depth of discharge (DoD). The effects of ageing are mainly two: capacity reduction which manifests with a progressive reduction of the energy that can be stored, and a rise in internal impedance whose effect is reducing the available power extracted from the battery for the same conditions [5, 6]. In both cases, ageing negatively affects battery reliability and safety which, for a F1 car, translates successfully completing a race or retiring the car. Most of the processes causing the performance loss, occur at similar timescales and cannot be studied independently, making the investigation of the ageing mechanism significantly complex. The dominant ageing mechanism is the solid electrolyte interface (SEI) formation that causes an increase in the impedance, this effect manifests itself mainly at the beginning of the life. Later, loss of Lithium in the active material takes place and becomes the dominant effect, leading to self-discharge and capacity fade [7]. In general, capacity fade is caused by structural changes occurring in the cell due to cycle, chemical decomposition and surface film modification.

Given the exponential rise in the importance of batteries for different critical applications such as automotive, it becomes crucial to accurately understand, monitor and find a methodology to model these effects to estimate battery life status. In this direction, over the years, strong efforts have been made in the research and development of any aspect, to improve battery efficiency, capacity and safety. A wide methodology used by researchers to monitor the battery life status is by defining a parameter referred to as state of health (SOH). It is evaluated as the ratio between the actual capacity and the capacity at the beginning of life (BOL). However, accurate estimation of SOH still presents many challenges due to the complexity of the internal chemistry and the difficulty of accessing required information. Actually there is not a universally recognised methodology to accurately estimate its value.

The battery used in the context of F1 races, is considered to be a battery for high-power applications, and it's subjected to extremely high discharge and recharge currents compared to its capacity, for that reason, it is important that each cell would be designed to have a small internal resistance and impedance to maximise the efficiency and the power extracted. Moreover, given that the regulations allow the use of a maximum of two batteries in a season without incurring penalties, it is important to

have reliable and safe batteries for, at least, half of the races.

This thesis presents a detailed study conducted during the PhD period to estimate the health of an F1 car battery. Among the various methodologies reported in the literature, two distinct Machine Learning (ML)-based approaches are explored and presented here. While both leverage ML, they rely on different concepts and physical phenomena.

The first approach employs ML to develop a Reduced Order Model (ROM) of a battery cell at 0 km. This model takes current, State of Charge (SoC), and temperature as inputs and estimates the expected voltage (VE) of a brand-new battery under specific operating conditions. By comparing this predicted VE with the actual VE of a used battery following the same operational profile, the battery's health can be assessed. The second approach focuses on estimating the State of Health (SOH) of the battery cell using four statistical indicators derived from voltage relaxation (VR) after a current spike. This method provides insight into the battery's degradation based on its response to transient conditions. Both approaches offer valuable perspectives on battery health estimation, contributing to the broader field of battery diagnostics in high-performance motorsports applications.

For the first work, the cell model takes as input current, SoC and temperature and estimates the voltage. The estimated voltage represents the response that a brand-new cell should have for the working conditions. This is used to calculate the efficiency that a battery (seen as a series/parallel composition of cells) would have over a cycle if it were new, and its value is compared with the efficiency calculated using data measured for a real and aged battery. This comparison can be synthesised by the definition of a new health parameter ($\Delta\eta$) as the difference between the theoretical efficiency η_{U_m} calculated using the data-driven reduced order model (ROM) of an average LIB cell and the one calculated from direct measures (η_{U_b}). Here, the problem of modelling a cell is tackled as a regression of a time series where data are chronologically ordered, and thus Recurrent Neural Network (RNN) is used to model the function describing the cell behaviour. In particular, a Long Short-Term Memory (LSTM) is used here because of its high effectiveness in representing time series and sequential data and the possibility to account for long-term dependencies. The accuracy of the model is evaluated through root mean squared error (RMSE). It is calculated for the training dataset, for a test dataset extracted from the original dataset and for two sets of data representing a constant current (CC) discharging test, which consists of an off-design working condition. The best model shows an error of 2.17 mV on the test dataset and 5.47 mV on a CC discharging test. The results obtained through the application of the method proposed, are shown by reporting the monitoring process employed for three active batteries. For each of them, the parameter $\Delta\eta$ is evaluated 11 times at regular intervals during their lifespan. For each evaluation, both the real battery and the model are subjected to the same working conditions in terms of current, SoC and temperature, and thus are directly comparable. The comparison of VE shows an increase in the life-monitoring parameter $\Delta\eta$ for subsequent calculations, suggesting that such an approach can be effectively used to estimate age in batteries. Moreover, the monitoring shows also that each battery degrades differently.

The other work focuses on estimating the capacity of any single cell composing the battery. It consists of using 4 statistical indicators extracted from the voltage

relaxation (VR) curve collected after the battery is subjected to a current spike. These statistical indicators are shown to have a correlation with the actual capacity of the cell and thus can be used to build up an estimator to estimate the cell capacity. The functions used to manipulate data and extract the features for each cell are max, min, var and avg. In the context of the presented work, these statistical indicators are used to build up a capacity estimator through Machine Learning (ML). In particular, three different ML models are implemented, to compare them and find out the most effective one in estimating the cell capacity. Specifically, the models implemented in the work are the linear regression (LR), Random Forest (RF) and Gradient Boosting Decision Tree (GBDT). The metrics used to realise the comparison is RMSE. The proposed methodology requires on-purpose tests. One of the main limitations is represented by the current spike that may pose a danger to the cells and accelerate their degradation. To reduce such risk during tests, three different values of the current spike are tested to find the smallest one resulting in a VR which carries enough information to allow a proper capacity estimation. The results show that the most effective model is the GBDT with a VR given by the highest current tested, it shows a RMSE of 0.00184 Ah.

Despite the way of working and the purpose of the two methods are different, the ultimate goal is the same: to find a methodology that allows one to estimate the battery life status when it cannot be attached to a charging line.

Chapter 2

Literature Review

As briefly introduced in Chapter 1, LIB batteries are the preferred energy storage system for many applications. The reasons behind the success, are mainly justified by their characteristic of high energy density and higher power densities compared with the other chemistries, as shown in Fig.2.1. The picture shows a Ragone plot which is used to display the characteristics of a variety of batteries. There the specific power is plotted against the specific energy in a logarithmic scale to allow the comparison of very different devices. Any class of device is represented by a curve, where each curve represents the envelope surrounding the performance of different batteries with the same chemistry [8]. From the plot, it is interesting to notice the wide range of energy and power densities LIBs batteries can ensure the same base chemistry. Together with

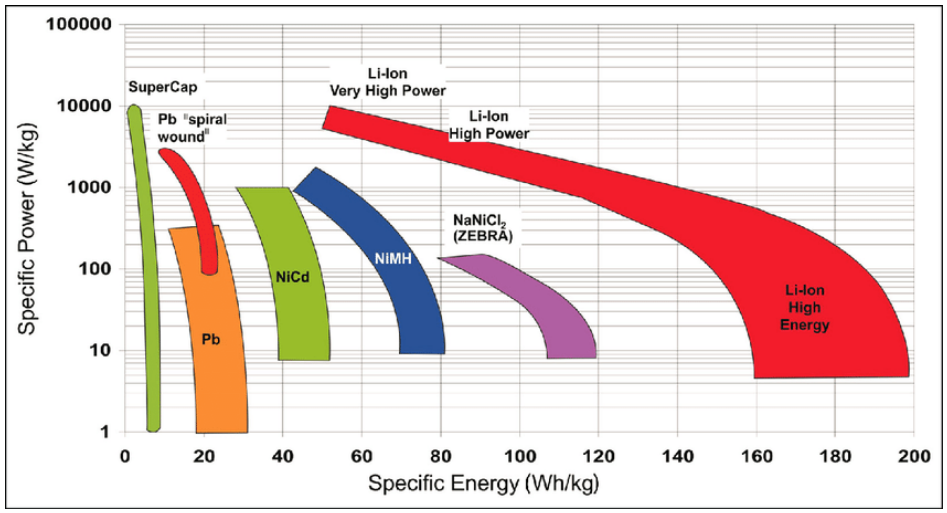


Figure 2.1: Ragone plot for different chemistries. Updated to include Li-ION chemistries

the properties of high power and energy densities, LIBs also have the properties of long cycle life with high safety, a continuously decreasing cost and a low self-discharge which allow charge/discharge cycling with high coulombic efficiencies. [9, 10, 11, 12]. All of these aspects, make the LIBs a very interesting candidate for automotive electric propulsion, but also allow LIBs to be fascinating for motorsport applications, making them the most used energy storage device in F1. One of the most critical aspects to take

care of in dealing with LIBs, is represented by the characteristic variation occurring while the cells age. This phenomenon is generally referred to as battery ageing and consists of battery health reduction during its lifespan, bringing to a capacity reduction and a resistance increase [13, 14]. These effects are directly related to the performance and safety of a battery and thus is it important to accurately determine the health of a battery, which can be addressed by the definition of a parameter referred to as state-of-health SOH [15, 16, 17, 18]. The SOH, is defined as the ratio between the actual battery capacity and the capacity at BOL, which can be translated mathematically as

$$SOH = \frac{Q}{Q_0} \times 100, \tag{2.1}$$

where Q is the actual capacity and Q_0 is the capacity at BOL. Given the importance of the topic, over the years a lot of effort has been made to find a methodology to accurately describe and assess the effects of ageing, anyway, given the complexity of the topic, there are still many challenges given by the complexity of internal chemistry and the difficulty in measuring the required parameters.

The purpose of this chapter is to provide a comprehensive overview of the existing literature about battery SOH estimation. Here, the literature is analysed with the aim of giving an overview of the existing methodologies and highlighting the strengths and the weaknesses of applying any proposed approach to the case under scrutiny.

2.1 Experimental methods

The first SOH estimation class analysed, is the set of experimental methods. They consist of performing direct measurements of the quantities of interest and using them to assess the battery SOH. Given the nature of the test, it is time-consuming and limited to laboratory tests.

One of the most used techniques is the coulomb counting. As the name suggests, it consists of measuring the charges which flow in or out from the battery during a CC charging or discharging test respectively. This counting method is one of the easiest ways to estimate SOH due to the simplicity of the test. The procedure consists of completely discharging the battery with a known relatively low CC and with a controlled temperature. The low current is essential to ensure the smallest perturbation of equilibrium, and the controlled temperature is essential to keep the repeatability of subsequent tests. Since the capacity is the number of charges which can be stored in the battery, it can be calculated by measuring the test CC and integrating it in time, as in Eq. 2.2.

$$Q = \frac{1}{3600} \int_0^t I_{dis} dt, \tag{2.2}$$

where Q is the actual capacity in Ah of the battery, I is the discharging current in A and t is the time in s. Once the capacity is measured the SOH can be estimated through its definition as in Eq. 2.1. The procedure to carry out the test makes this procedure time-consuming and not applicable to real-life scenarios. Moreover, the precision of the method highly depends on the precision of the measurement instruments since the error cumulates because of the integral [19, 7].

Increment of battery internal resistance is a good indicator for battery life status. Because of that, researchers developed methodologies to calculate the internal resistance from measured data. The test consists of applying a known current and measuring the voltage drop, then by using Ohm's law, it is possible to calculate the DC value of the internal resistance [20] as:

$$R = \frac{\Delta V}{\Delta I}, \tag{2.3}$$

where R_0 is the ohmic resistance, ΔV is the voltage drop and ΔI is the current pulse applied [21, 22].

Another methodology to estimate the battery SOH, is by estimating the actual value of its impedance which, as the resistance, increases with ageing. The measurement of the impedance is realised with a technique referred to as electrochemical impedance spectroscopy (EIS). EIS is a powerful technique to study the electrochemical process inside the cell. It is a non-destructive test used to analyse the dynamic processes occurring inside a battery [23, 24]. The test consists of applying a perturbation in the electrochemistry composing the cell through a small sinusoidal alternate current (AC) over a wide range of frequencies while monitoring the sinusoidal response. The strength of EIS lies in its ability to extract information about different electrical, chemical and physical parameters, each of them having a different time constant τ. Time constant is a measure of the temporal behaviour of a particular process, for the case under scrutiny it is defined as:

$$\tau = RC \tag{2.4}$$

where R is the resistance (in Ω) of a resistor and C is the capacitance (in F) of a capacitor. The test allows a high-quality characterisation of the battery health status [25, 26]. Anyway, its real-life application is complex given the nature of the test which is suitable only for laboratory. Generally, the measurements carried out through the EIS are generally used to characterise battery physical models represented via linear circuitry elements [27].

It is highly recognised that the battery charging curve changes with ageing. Some authors used the charging/discharge profiles to explore variations in electrochemical properties of the cell based on voltage measurement under constant current [28]. This procedure, anyway, does not account for the temperature effect on the charging curve. Another source of error is represented by the charging times: electric vehicle batteries are charged before they reach the full discharge, and thus it is impossible to access a full charging curve. An alternative might be using the constant voltage (CV) charging curve. Researchers find a correlation between CV charging curve with battery capacity, and thus with battery SOH [29, 30]. The main limitations of the methodologies using CV curves are represented by the fact that the CV charging tests are highly time-consuming, and there is no extensive comparative study on the features of interest. Another limitation to the applicability is the fact that to get CC-CV charging curves the battery must be attached to a charging line. This might not be possible for particular applications such as F1 batteries.

In the last years, differential analysis (DA) methodologies gained a lot of attention. These consist of applying differential calculations on voltage curves to extract features related to SOH. The two most effective methodologies are incremental capacity analysis

(ICA) [31, 32] and differential voltage analysis (DVA) [33, 34]. By differentiating the charged capacity over the voltage, incremental capacity (IC) curves can be obtained (see Fig. 2.2). The procedure consists of charging and discharging the battery at low CC to extract a relationship between capacity and cell voltage under quasi-steady conditions. The relationship is then analysed through differentiation. Both of these methodologies are a valuable technique to monitor the capacity loss of the battery. Mathematically, the incremental capacity (IC) is derived as the gradient of charged/discharged capacity with respect to the cell voltage (see Eq. 2.5) and differential voltage (DV) curve is computed as the gradient of voltage with respect to capacity (see Eq. 2.6):

$$\frac{dQ}{dV} = \frac{\Delta Q}{\Delta V} \tag{2.5}$$

$$\frac{dV}{dQ} = \frac{\Delta V}{\Delta Q}. \tag{2.6}$$

At different ageing phases, the ICA and DVA peaks have different amplitude, positions and shapes which is the main indication of the health of the battery, and thus can be considered as a sign of battery ageing and degradation [17]. An example of an IC curve is represented in Fig. 2.2.

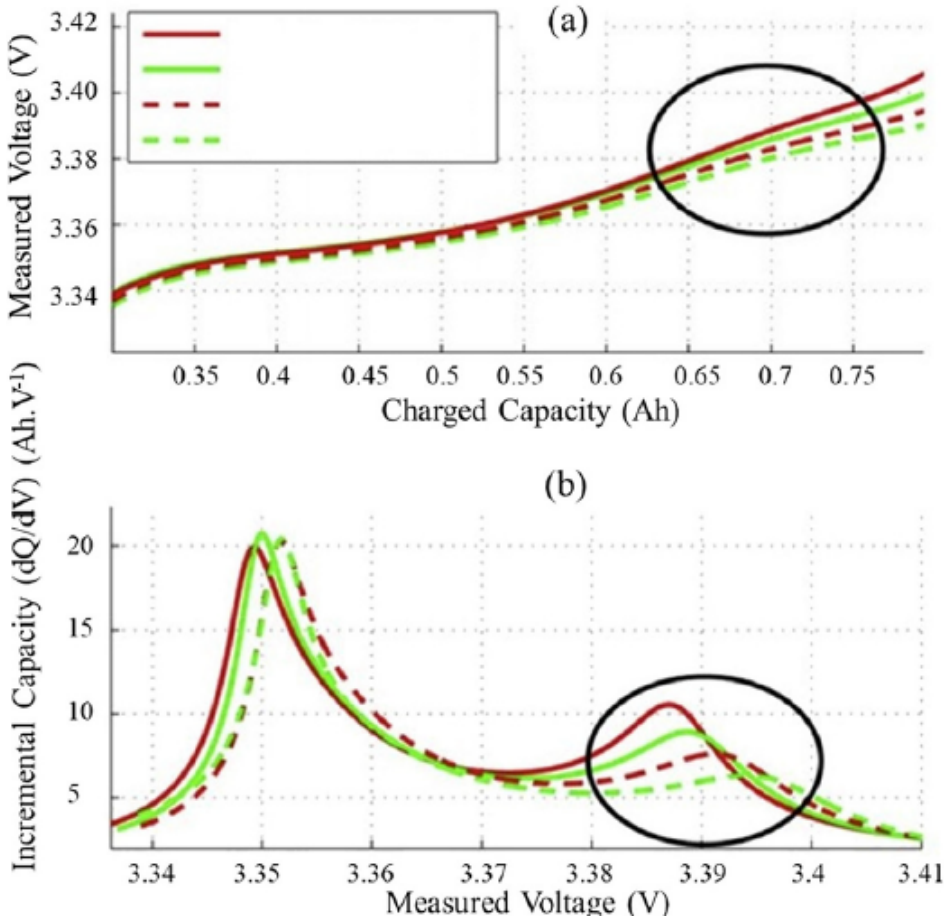


Figure 2.2: Example of a DV analysis which shows the effect of the capacity fade on the differential curves.

Battery degradation directly affects system efficiency. Most of the studies focus on capacity fade and impedance rise, making efficiency less frequently considered. Efficiency is defined as the ratio of the discharged energy to the recharged energy. For batteries, three different efficiencies can be defined: Coulombic efficiency, Voltaic efficiency and Energy efficiency [35, 36]. The first work defining a direction to calculate the efficiency of a battery is from Kang *et al.* [37]. They proposed a novel methodology to calculate the efficiency of rechargeable batteries based on electrochemical theory. The methodology is tested on $LiFePo_4$ power cells. Redondo-Iglesias *et al.* [38], analyse the energy efficiency degradation of NMC and LFP Li-ION cells under calendar ageing. They find out that the efficiency of cells reduces while ageing goes on. Moreover, they implemented an empirical model for each chemistry to estimate the efficiency degradation. Another contribution in this direction is given by Zhu *et al.* [39]. They examine coulombic and energy efficiency for Ni-MH batteries at different charge/discharge rates, showing that both are a function of the charge/discharge rate itself and of the SoC. Stroe *et al.* [40] made a comparison between the performance of a 13 Ah high-power lithium battery at its BOL and at two different degradation levels. The quantities considered are capacity, internal resistance, open circuit voltage (OCV) and energy efficiency. Their work highlights that efficiency is a parameter that can be used in assessing the performance of a battery. Moreover, they find out that the efficiency decreases while the battery ages, and that the ageing has a huge impact on energy efficiency. Yang *et al.* [41], explore the behaviour of Coulombic Efficiency and its relation with capacity fade for cylindrical $LiFePo_4$ cells, and $Li(NiMnCo)O_2$ cells. They used three cells for each type. They find out that for most of the cells, there is a sharp coulombic efficiency increase in initial cycles, and then it is relatively stable for several hundreds of cycles. Moreover, they identified a relation between coulombic efficiency and capacity degradation for each cell type. In particular, they observed that a reduction in coulombic efficiency corresponds to an increase in the degradation rate. Eventually, they proposed a relationship between efficiency and capacity degradation which can help in developing battery models, estimate battery SOH and provide early warnings for battery failure. Despite different typologies of efficiency that can be defined to assess battery performance, most researchers focused their attention on coulombic and energy efficiencies, making voltaic efficiency not considered for health estimation. To the best of the author's knowledge, there isn't any contribution in literature considering voltaic efficiency as a parameter to estimate battery SOH.

Given the importance of the topic, researchers developed different methodologies to assess the battery SOH in a simple way. Researchers showed that an important characteristic which can be easily obtained through voltage measurements is Voltage Relaxation (VR). Voltage relaxation behaviour contains information about the lithium ions and electrons transportation, and thus it can be used as an indicator on SOH evaluation since it is correlated to the capacity. VR can be identified in the potential of the battery immediately after the current interruption following a charging or discharging procedure. There, the voltage will continue to vary and return to the equilibrium OCV. It has been proven that the relaxation process is related to the battery SOH. The correlation is given by the information about the Li-ions and electrons transportation in the voltage relaxation process [42, 43, 44]. Based on this phenomenon, researchers developed different ways to properly estimate battery SOH. Uhlmann *et al.* [44] and

Lüders *et al.* [45] used the correlation between VR and lithium plating effect to develop models to estimate the cell capacity. Baghdadi *et al.* [43] monitored VR over ageing for a Li-ion battery. To assess the VR, they measured the OCV of the battery after 30 min at rest, this quantity shows a linear correlation with the actual capacity of the battery and the authors developed a linear model to estimate SOH. Their best model shows a mean error less than the 2%. [46] *et al.* developed a non-linear structural model to analyse the VR of lithium batteries. In [47], the authors developed a voltage relaxation method to estimate the OCV from the experimental investigation of the relationship between the open-circuit time and the time constant of an RC branch of a model. The work is then used by Fang *et al.* [48] as a base to develop an SOH estimation methodology based on two model parameters. In particular, they obtained a linear relationship between the capacity and the model parameters identified in open-circuit time and time constant. The voltage relaxation data are collected after a CC discharge at 1C for 180 s and a rest of 2 hours.

The methods presented in this Paragraph, are shown to be effective in detecting capacity fade and ageing mechanisms. However, effective experimental procedures are challenging to apply and replicate in real-life scenarios due to the extremely precise procedures that require sophisticated experimental equipment and the environmental differences occurring between the laboratory and real life (such as the computational power available in laboratories compared with that of BMS). Anyway, experiments can provide insights and data to further develop model-based techniques and different methodologies such as those based on voltage relaxation.

2.2 Physics-based methods

Despite experimental methodologies that can provide precise degradation information, they are not suitable to be used in BMS because of the difference existing between laboratory and real-life environments or the possibility of collecting all the data needed to properly estimate the required quantities. An approach which can help in this direction is physics-based modelling. Physics-based methodologies consist of a set of differential equations describing the cell behaviour. The most studied modelling approaches in this direction are two: electrochemical models (EMs) and equivalent circuit model (ECM). These two categories represent the state-of-the-art of physics-based modelling for batteries.

In EMs, the cell is modelled by considering the chemical and physical laws governing the phenomenon. A EM model of a LIB, consists of a set of non-linear partial differential equations representing, with high fidelity, the internal chemical and physical dynamics governing the battery (see Fig. 2.3) and its degradation [49, 50]. The models in this category, are highly accurate, but the large amount of calculations required might represent a limit, mainly when used on battery management system (BMS). To overcome these issues, researchers developed a simplified version, referred to as pseudo-two dimensional (P2D) model. It can represent the cell dynamics with high accuracy but lower computational cost. The bottleneck of the EM, is given by the representation of the porous structure of the electrodes because of its complexity. In P2D model the representation, is substituted by the porous-electrode theory using a macroscopic approach [51]. Anyway, even P2D models are not suitable to be used in BMS because

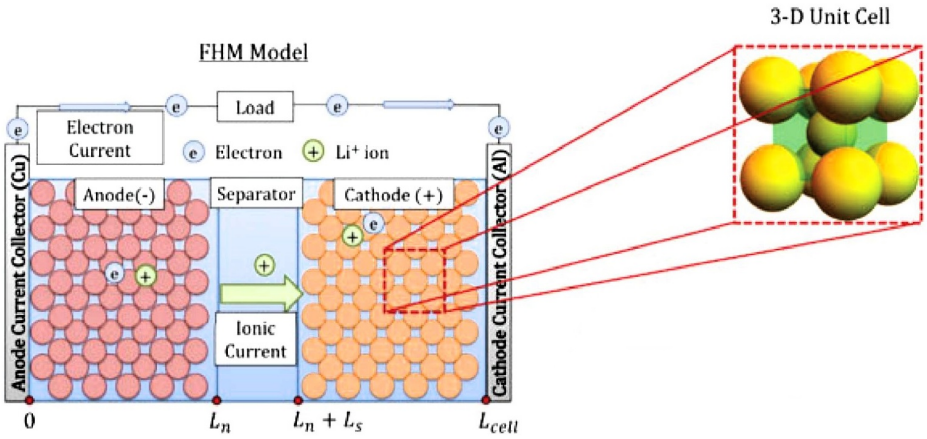


Figure 2.3: Schematic diagram of an EM model of a LIB

of the large amount of calculation required.

In ECMs, the battery is represented through linear circuit elements such as resistances, inductors, and capacitors [52, 53, 54] arranged in a series/parallel disposition, as represented in Fig. 2.4. Given the nature of these models, they are characterised by high computational efficiency which makes them suitable for real-time applications. Anyway, their accuracy strongly depends on the characterisation of the parameters which delineate the model itself. ECMs are characterised by a high computational

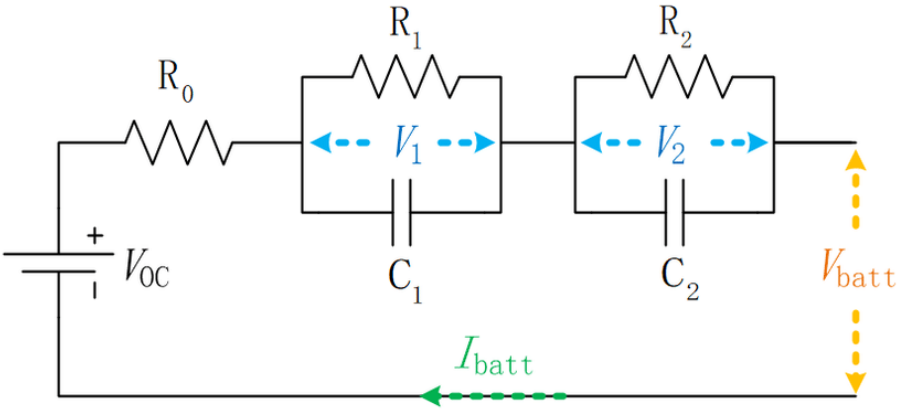


Figure 2.4: Schematic representation of an ECM of a LIB

efficiency, but their accuracy is strongly dependent upon the characterisation of the parameters driving the model itself. Most of the time, they are defined following an experimental approach, which means designing and realise on-purpose tests for parameter identification, and thus to accurately determine them might be expensive and time-consuming [55, 56].

In the last years, the availability of data has increased exponentially thanks to the advent of connected vehicles which record and share data continuously [57]. This scenario, together with the advancement of ML algorithms, encourages researchers to explore data-driven applications in battery modelling. The main difference between these two methods is given by the implementation methodology and the philosophical

approach: model-based methodologies consist of representing the system through a set of partial differential equations describing the physical and chemical phenomena driving the system itself, and thus, they require a deep knowledge of the system to be represented. Data-driven models build up a black-box model starting from data measured during on-purpose experiments or real usage and do not require a deep knowledge of the system to represent, on the other hand, these methodologies are complex to implement.

2.3 Data-driven methods

The traditional simulation and experiment methods in battery research usually require large research resources in combination with sophisticated domain knowledge or experience to enhance the effectiveness of trial-and-error approaches. Data-driven methodologies consist of using collected data to define a model which approximates a function describing a particular phenomenon. These methods, rely on the application of ML to gain insights from data [58].

Before introducing the applications of ML in battery research, it is necessary to introduce what ML is and how it is applied in scientific research. Here a brief introduction to ML and its history is reported, and the concept is developed in the next Chapter. Machine Learning is a subset of Artificial Intelligence. The Artificial Intelligence (AI) is a field of study referring to the technologies to build up computers that can mimic the cognitive functions of human intelligence, including reading, understanding and properly answering, and creating a communication connection. A machine is then said to be intelligent if it can perceive its environment and use learning to take actions that maximise the probability of achieving defined goals in a variety of novel situations. The first research contribution globally recognised in this field is the one by McCulloch and Pitts [59]. They proposed a model of binary artificial neurons that can be only in an "on" or "off" state, triggered to "on" only in response to a certain number of stimuli from the neighbourhood neurons. ML is strictly connected to AI, it is a field of study that focuses on developing algorithms and models aimed at learning from data to enable programs to imitate the way that humans learn. The first contribution to ML research is from Arthur Lee Samuel [60] in 1959, he published a paper showing a self-learning algorithm for playing checkers. In particular, Samuel states that the program learns to play a better game than can be played by the person who wrote the program and can do this in a short period of time giving only the rules and a redundant and incomplete set list of parameters. In the paper appear for the first time the term *Machine Learning*. Over the years, researchers applied the concept of ML to different and the mathematical research keeps going and an article defining the perceptron has been published [61]. Later, in the 1970s, the research on ML slowed down, and a period called ML winter began. In this period, very little advancement in the research was done. The AI winter overs in 1980, when Fukushima [62] proposed a neural network (NN) for a mechanism of visual pattern recognition. Since then, the 1980s saw an explosion in AI research. In 1982 Hopefield [63] popularised the recurrent neural network. Later, in 1986 Rumelhart *et al.* [64] published the backpropagation algorithm, which is the main algorithm used nowadays to train neural networks. In 1988 Hornik *et al.* [65] published the universal approximation theorem showing that given

any function f, exists a sequence of neural networks that can approximate the function itself. In 1989, Watkins [66] developed a thesis that introduced Q-learning that improved the applicability of reinforcement learning. Following, in 1992, a ML model learned to play backgammon and was able to rival the abilities of top human backgammon players. The 1990s, saw the publications of RF, Support Vector Machine (SVM) and LSTM [67, 68, 69]. Anyway, the main limitations to the explosion of the research and the applicability of the technology, were imposed by hardware limitations. These limitations have been overcome in the last 10 years, and thanks to hardware advancements, research in ML made huge steps. This allows the development of models which are able to beat a human player in the board game Go [70], models able to predict the proteins' structure [71] and models able to develop highly sophisticated chat-bots [72, 73] such as ChatGPT.

The exponential rise of data availability fascinated also the researchers working on batteries. They start to apply ML to battery research giving birth to a new field of study referred to as battery informatics, which is defined as the research field where ML is used as the main tool for battery data analysis. In battery research, many are successful examples of the application of ML. For example, in [74, 75, 76] the authors successfully use machine learning in the field of material discovery, allowing a reduction in the number of calculations to speed up the exploration of novel materials. Going to a smaller scale, [77, 78, 79] used machine learning and data-driven methods to analyse the microstructure of cathodes and anodes. These models are trained using data collected from physical tests such as X-ray neutron diffraction or nuclear magnetic resonance. Many other applications of ML to batteries can be found in the literature such as optimising performance [80] or optimising charging protocol [81].

Given the complexity of the problem and the computational effort required to implement accurate battery models, researchers developed data-driven ROM of battery to accurately predict its voltage in given working conditions. Due to the strong dependence on prior information, most data-driven models make use of RNNs [82]. Capizzi *et al.* [83] implemented a hybrid model through a RNN having a double objective: it is used as SoC observer and as an estimator for the parameters of a non-linear mathematical model of a lead-acid battery. They concluded that the RNN model is more accurate than an ECM since having one order of magnitude lower mean square error. Zhao *et al.* [84] developed two models using RNN which can be used in place of an ECM. They used a Gated Recurrent Unit (GRU) model trained with sequential drive cycle data at three different temperatures. Hong *et al.* [85] proposed an LSTM model to estimate battery fault based on the abnormity voltage prediction. The main limitation of their work is represented by the model taking as input the brake pedal signal, which for some applications might be not available. Their model is trained using data coming from real-world operational electric vehicles. Zhu *et al.* [86] used a data-driven model based on LSTM model with a many-to-many architecture to predict the battery voltage using as input past voltage, current and SoC information. The model is trained using data from experiments that are carried out at three different temperatures. Their best model reaches a RMSE of 0.0997 when trained with non-normalised data at a temperature of $T = 0°C$. The main limitation of the approaches presented is the limited amount of temperature values at which the tests are carried out and then given as input in training and the fact that the model takes, as a feature, the vector

of past voltage values. Anyway, many authors [82, 84, 86] see LSTM as a promising approach to accurately predict the voltage of the battery and implement a ROM since the battery has a long time constant.

One of the research branches where ML is applied the most is the estimation of battery SOH and SoC [87, 88]. In the last few years the number of publications exponentially rose, matching the growth interest in ML. In the literature, it is possible to find out a wide range of applications, using desperate algorithms, from simple LR to the more complex RNN. For example, Severson *et al.* [89] implemented a LR to predict and classify cells by cycle lives using discharge voltage curves from early cycles. The dataset used to train the model consists of 124 Lithium Iron Phosphate/graphite cells cycled under fast-charging conditions. The best model reaches 9.1% in qualitatively predicting the cell life using its first 100 cycles. Won You *et al.* [90] developed a data-driven framework to estimate SOH of a battery on the fly by using data coming from BMS such as current, voltage and temperature while leveraging their historical distribution. They used a plain Feedforward Neural Network (FFNN) to estimate battery degradation implementing a high-accuracy model that shows an average error of 2.18%. In [91], Wei *et al.* proposed a SOH estimation methodology based on extreme learning machine (ELM). The model is trained on the publicly available Battery Data Set (NASA PCoE Research Center). With the advancement of chips and the increasing availability of data, researchers have begun to apply DL to the SOH estimation. Compared with shallow NN, DL models can map more complex functions. In [92] Zhang *et al.*, implemented a four-layer NN to estimate the SOH of batteries under a CC discharge. They extracted 5 different features from partial IC curves and processed them with a Gaussian filter starting from the publicly available Battery Data Set from NASA. In [93, 94], the authors use a nonlinear autoregressive exogenous (NARX) model to estimate the SOH of batteries to account for the possibility of tracking degradation given by RNN. One of the main problems in training RNN is the exploding and vanishing gradient (see Section 3.2.3). Vanishing or exploding gradient can translate into inefficient or impossible training. A way to overcome this limitation is by using a gated RNN. Li *et al.* [95] implemented a deep LSTM to estimate the SOH of cells under real-world working conditions using as input the voltage and the time length of a partial charging curve extracted during a CC-CV charging operation. In the best case, their model reaches a mean absolute percentage error (MAPE) of 0.76%. Kim *et al.* [96] implemented a deep variational LSTM with transfer learning to forecast LIBs degradation pattern. They used three public datasets: CALCE is used to train the model, Cavendish Laboratory and NASA datasets are used to implement transfer learning to use the model with different batteries. Despite the high accuracy of LSTM, it might be too computationally expensive to be used on BMS or embedded systems. In [97, 98] the authors, substitute the LSTM model with a GRU model. They show that the GRU is less accurate but is more suitable to be used in embedded systems because of the reduced calculation amount, given by a lower number of parameters.

A research direction explored is that of using voltage relaxation and machine learning to model the battery SOH. In this direction, Zhu *et al.* [99] develop ML models to estimate cell capacity using as input statistical features derived from VR for different battery compositions: a battery made by NCA cells, one made of NCM cells and one made by a combination of the two. The VR is extracted after a full charge and 30 min

at rest for the NCA and NCM batteries and 60 min for the combined battery chemistry. Data are converted into 6 statistical indicators (Variance, Skewness, Maximum, Minimum, Mean and Kurtosis) describing the curve. The experiments to extract data are carried out on 130 commercial Li-Ion cells cycled in a temperature-controlled chamber under various conditions. Their best model reaches a root mean squared error (RMSE) of 1.1%. Anyway, it might be that the length of the test required to extract the data becomes a constraint, and a shorter data recording might be necessary. Moreover, the voltage relaxation is extracted after a full CCCV charging, thus it is possible only when the battery is connected to a charging line. Since the relaxation time used in [99] might be a limitation for some applications, Fan & Zhang [100] implemented a convolutional neural network (CNN) to estimate the capacity of the cells with different degradation paths. The model is implemented using VR data collected for 10 s, which is shown to be correlated with the capacity. The data used in the work are publicly available and formed by 28 commercial NCA/graphite cells tested in a constant-temperature chamber. Their model reaches an average percentage error (APE) of 1.8%. Anyway, as in the work by Zhu [99], the VR is extracted after a CC charging procedure when the cell SoC is 100%.

Despite the accuracy shown by the model presented, most of them use, as features, data collected during a well-defined discharging cycle or during laboratory tests. These conditions are not always reproducible for real-life batteries. This is particularly true for batteries used on F1 cars since it is not possible to access the full or partial charging/discharging curves.

Models created with ML are often referred to as black-box models since it is complex to inspect their internal behaviour. Their advantage is given by that they do not require deep knowledge of the system's physical laws, and the function mapping inputs to outputs is derived directly from the data. Anyway, their quality strongly depends upon the data quality and having high-quality data is, in most cases, highly expensive. This thesis is going to report a detailed analysis of the two works made during the PhD period in Ferrari. The objective of the works presented here is to show how battery intelligence is applied to provide a simple and fast methodology to assess the battery SOH. In the first work reported, the problem is tackled by implementing a data-driven ROM of a brand-new average cell to predict the voltage response while working in given conditions of i, SoC and T. To develop the model, the battery is considered as realised by a series/parallel arrangement of cells all of them behaving the same. The cell is represented through a deep LSTM model predicting the voltage response of the battery at the next time step using the actual working conditions and 2 s of history. The Predicted Voltage is then used to calculate the VE that a new battery should have while experiencing a given current, SoC and temperature. The VE is compared with the VE obtained with an on-car battery in order to estimate its health and experience ageing. To account for the battery health, a new parameter is then defined as the difference between the efficiency of the new battery evaluated through the model and the efficiency of the on-car battery, calculated from measured data. The methodology proposed is tested on three different on-car batteries, for which tests are carried out during their entire lifespan at regular intervals. The results of the study show that the method proposed might be effective in estimating battery health.

In the second work, a methodology for cell capacity is proposed. The methodology

proposed is a simple and fast procedure which can be used when the capacity is not directly measurable, or it is not possible to attach the vehicle to a charging line. The work consists of extracting VR after the application of a charging current spike lasting 5 s, when the current is removed data are collected for the following 10 s. The curve of VR is processed to extract four statistical indicators identified in Maximum (max), Minimum (min), Mean (mean) and Variance (var) representing the curve It is shown that the statistical indicators are correlated with the cell's actual capacity. The innovation reported in this work is given by the fact that the VR is obtained with a current spike. To the best of the author's knowledge, this is the first time that VR has been extracted from current spikes. Anyway, Since the spike in current might be a limitation for battery safety, three spikes of different intensities are analysed and compared to identify the smallest value that can be applied to generate a response containing enough information about the actual capacity. The indicators are used to train three different ML models identified in LR, GBDT and RF to estimate cell capacity and RMSE are used to compare them to choose the best one.

Chapter 3

Fundamentals of AI

In the era of data, AI is becoming a thriving field for many practical applications and research activities. Despite the idea of AI and ML is not new, the recent improvements in hardware development, the increased amount of data and the advances in algorithms efficiency verified in the last ten years allowed a wide spreading and fast-growing of such technology, which quickly make AI becoming an integral part of people's daily life. Nowadays, it is present in many everyday applications, such as smartphones with voice assistance as well as in streaming platforms for personalised recommendations or, for example, in other sectors such as healthcare, finance and automotive, demonstrating a strong influence in modern society and a broad range of possible applications. This Chapter is thought to give information about the fundamentals of AI and its developments. Here, it is reported an introduction to AI and its fundamentals with the aim of giving an idea and an intuition about this groundbreaking technology. In particular, here it is given a general introduction to the main concepts driving the development of AI. Then are underlined the differences between AI, ML and Deep Learning (DL). Eventually, it is reported a detailed description of the algorithms used to realise the works reported in this thesis, explaining the mathematics which guides these models, and their architecture together with an explanation to give an intuition.

3.1 Artificial intelligence, Machine learning and Deep learning

Since the era of computers, researchers have tried to program intelligent entities - that is - machines that can perceive their environment and use learning to take actions that maximise their possibility of achieving defined goals in a variety of novel situations, mimicking the way human learns. Several definitions of AI exist. Some defined intelligence in terms of fidelity to human performance, while others prefer a formal definition called rationality, which can be summarised as the "right thing". At the very beginning, efforts were made in the direction of handcrafting a sufficiently large number of explicit rules for manipulating knowledge. This field of research is known as *symbolic AI*, which is proved to be suitable for solving well-defined logical problems, but it is not for more complex problems such as image classification or speech recognition [101], thus a new field of study was developed referred to as Machine Learning (ML).

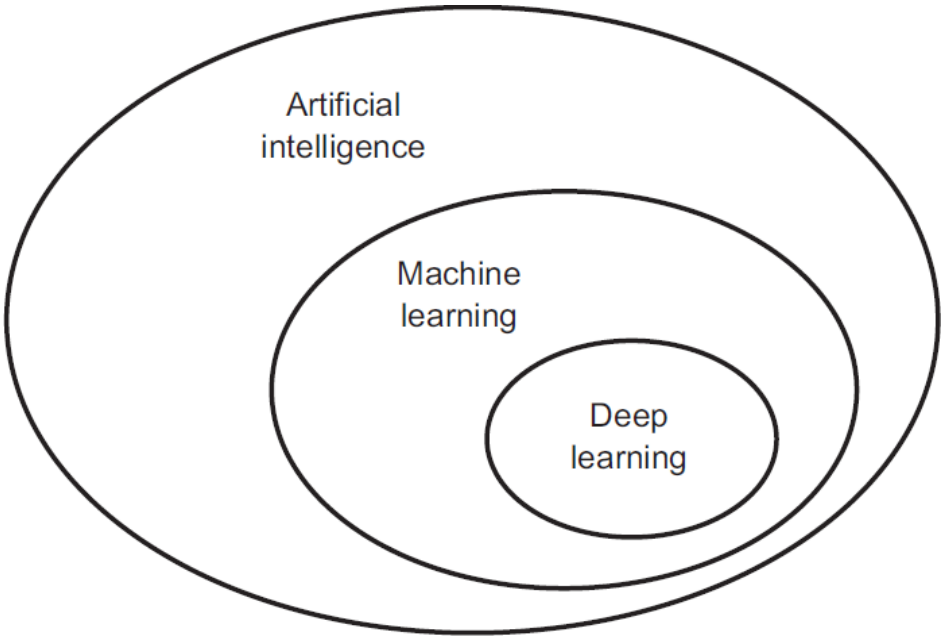


Figure 3.1: Relationship between artificial intelligence, machine learning and deep learning

ML is a branch of AI that enables algorithms to learn from data. In particular, the idea behind ML is to implement an algorithm which can learn the governing rules only by looking at the data, rather than being crafted by programmers. The main difference between classical programming and ML can be synthesised as given by the fact that in classical programming people define the rules and provide data to extract answers from that set of rules, in ML the idea is to provide data and answers to extract the rules. A ML system is then *trained* over the data rather than explicitly programmed. A rigorous definition about what is machine learning is provided by Mitchell [102]: "A program is said to learn from Experience E with respect to some task T and performance measure P, if its performance P at task T improves with experience E". Tasks T are usually described in terms of how a machine learning system should process an example, that is a collection of features measured from the event the ML should process. Typically, an example is represented as an array $\boldsymbol{x} \in \mathbb{R}^n$, where each entry x_i of the vector is a different feature, the arrays can be concatenated together to form a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, where each entry of the matrix is a vector $\boldsymbol{x}$ representing a particular quantity. Many kinds of tasks can be solved, such as regression or classification. Regression consists of a set of statistical processes for estimating a numerical value given some inputs. Once the model is properly trained, it is able to represent a function from the space of the features to the space of the labels, $f : \mathbb{R}^n \to \mathbb{R}^m$. In classification, a model is trained in a way that is then able to specify which of the k clusters an input belongs to. In this case, once the algorithm is trained, it can represent a function that maps the input space to the clusters' space: $f : \mathbb{R} \to 1, \ldots, k$. When $y = f(\boldsymbol{x})$, the algorithm assigns an input described by $\boldsymbol{x}$ to a category identified by y, but it is also possible to train f to output a probability distribution over the classes. The performance P, is a quantitative measure to evaluate the ability of an ML algorithm to predict the correct output for a given task T. As can be deduced, the choice of the performance

function is strictly connected with the task to be solved and sometimes can result in a difficult process. The experience E refers to how a learning algorithm masters the data from which it learns. The easiest angle from which the experience can be described is by dividing the problems in two: supervised and unsupervised. To implement a ML model, it is essential to have three things:

- Input quantities, generally collected in a multidimensional array, referred to as dataset. The quantities contained in the dataset are also referred to as *features*;
- For some kind of experience E, it might be necessary to have examples of the expected output - also referred to as *label* or *target*;
- A proper way to measure whether the algorithm is doing a good job - useful to understand the distance between the model's outcome and the expected output. It is used as feedback to adjust the way the algorithm works.

A ML model transforms input data into meaningful outputs through a set of rules inferred from data during a fitting phase that is defined as *training* phase. During the training phase, the model is exposed to a set of inputs and, if needed a set of labels, which are used for the learning process that consists in looking for rules that can represent a phenomenon described by the data. Fitting a model can be thought of as a procedure for selecting the most proper function to represent the data. The function is chosen from a set of possible functions. The space composed of all the functions is called *hypotheses space*.

As evidenced in Chapter 2, ML can be used in a wide range of environments and applications. Depending on the purpose of the model and on the particular field studied, the learning process can vary. In particular, it is possible to identify three main learning paradigms:

- Supervised Learning;
- Unsupervised Learning;
- Reinforcement Learning.

A schematic representation of the three paradigms can be seen in Fig. 3.2.
Supervised learning consists of training a model in a way that it can map input data to known values. The training procedure consists of using as input a set of examples together with the corresponding desired output such that the algorithm can recognise patterns and learn, similarly to how humans learn from experience and examples. In supervised learning, the model observes input-output pairs and learns a function that maps from input to output. The most appropriate function is extracted from the hypotheses space defined by the data used during the training phase. The goal is to have a model general enough to be able to correctly determine the output of data never seen before. In unsupervised learning, the model learns patterns in the input data without any explicit feedback (the output value is not available). The model tries to learn patterns exclusively from unlabelled data looking to find similarities, differences and structures in input data. It can be applied to different kinds of analysis, such as clustering, dimensionality reduction or anomaly detection. Reinforcement learning

is an interdisciplinary area of ML and optimal control concerned with how a model learns to make decisions in a dynamic environment to maximise cumulative reward. Different from supervised learning, reinforcement learning does not need labelled data or suboptimal actions to be explicitly programmed. It only focuses on the exploration of a particular context and taking actions which maximise the reward. It is inspired by behavioural psychology, in particular the way humans and animals learn through trial and error and reward and punishment. For the works reported in this thesis, it is used

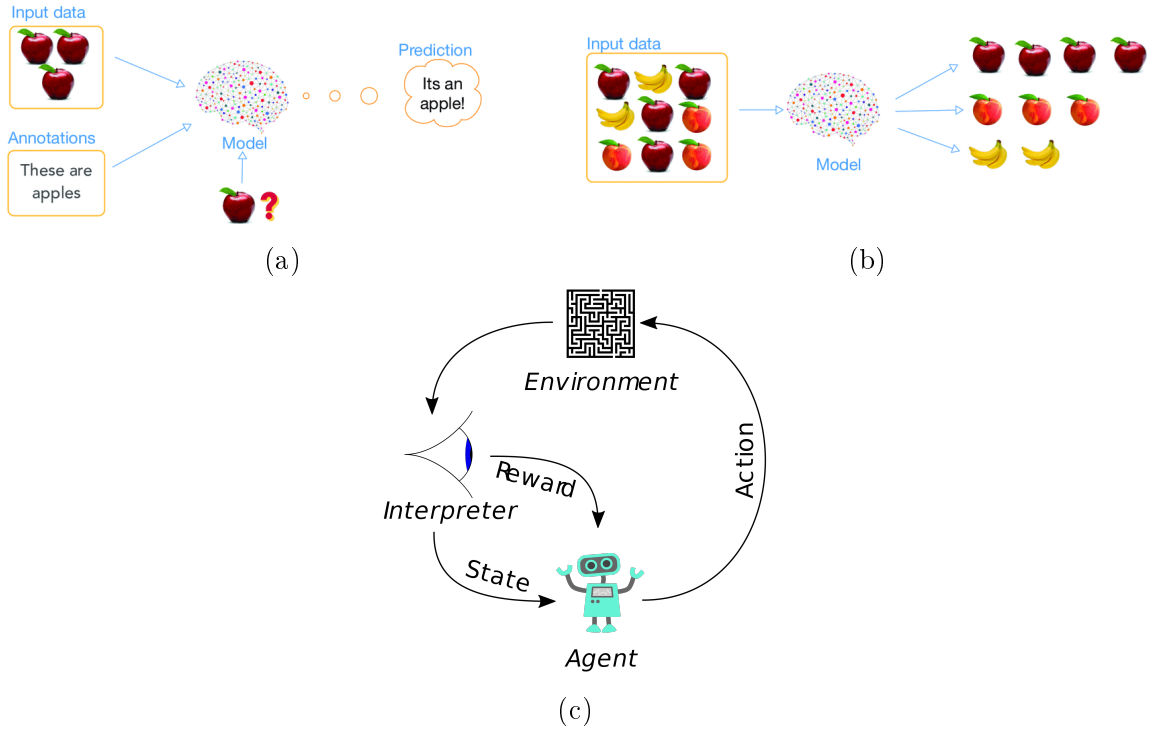


Figure 3.2: Schematic representation of the three main paradigms of machine learning: a) Supervised Learning, b) Unsupervised Learning and c) Reinforcement Learning

only the Supervised Learning paradigm, and thus, here this topic is explored in deep. Any problem faced in the context of this thesis can be seen as a regression problem.

3.1.1 Supervised Learning

A model is learning if it improves its performance after making observations about the environment. When a computer observes some data, builds a model based on that data, it is called supervised learning. More specifically, given a set of n examples input-output pairs

$$(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n),$$

generated by an unknown function $\boldsymbol{y} = f(\boldsymbol{x})$, a supervised learning task consists in discovering a function $\hat{f}$ that approximate the true function. The function $\hat{f}$ is called *hypotheses*. Such function is extracted from a hypothesis space $\mathcal{H}$ containing all the possible functions defined by the data available. In the dataset, any entry $\boldsymbol{x}_i$ of the array $\boldsymbol{x}$ is a different feature and any entry $\boldsymbol{y}_i$ of the label array $\boldsymbol{y}$ is referred to as *ground truth*, that is the true answer required to the model. To effectively extract the function

from the hypothesis space, it is necessary to properly define $\mathcal{H}$, this can be done by having prior knowledge about dependences upon the variables in the dataset (such as in the case of a physical system with well-known governing equations) or through an exploratory data analysis (EDA) which consists in statistically analyse the variables composing the dataset. In cases where there is no knowledge about the system that generated the data and the EDA is not effective, a way is to try multiple hypotheses spaces and choose which one works best. The function is the one for which each $\hat{f}(\boldsymbol{x}_i)$ is as close as possible to $\boldsymbol{y}_i$, this is particularly true for continuous-valued output, for which it is not possible to expect an exact match between ground truth and predictions.

As briefly introduced, it is essential for any ML model to be general. Generalisation refers to the ability of the model itself to be able to make correct predictions on data never seen during training. To ensure generalisation, the prediction ability of the hypotheses function can't be evaluated over the same dataset used to extract the function itself from the hypotheses space. It is thus necessary to define a second dataset (most of the time extracted as a sub-dataset from the original one before training) composed of data not seen during the training procedure. This is important because the generalisation performance is used to guide the choice of the learning method, of the model and to have a measure of the quality of the final model. Typically, ML models have parameters to be tuned, referred to as *hyperparameters* which varies the complexity of the model and affects its accuracy and generalisation ability. Because of that, it is important to properly tune such parameters in order to minimise the estimation error. Generally, the best approach is to divide the original dataset into three sub-datasets with different number of observations:

- Training dataset;
- Validation dataset;
- Test dataset.

The training dataset is used to fit the model, that is the process with which the model learns from the given data, the validation dataset is used to assess the generalisation accuracy of the model while the training goes on and to set the hyperparameters, while the test dataset is used to assess the generalisation performance of the model once training is completed. A schematic representation of a split dataset is reported in Fig. 3.3. There, the starting dataset is subdivided in three different parts, each composed of a different quantity of examples. In general, there is no rule on how to choose the number of observations for each dataset, as this strongly depends on the quality and the quantity of data in the original dataset. The definition of three different datasets, allows the programmer to understand if the ML model is learning correctly while fitting goes on, and helps in detecting if the model is general enough. This arises from the necessity of finding an equilibrium between the generalisation of the model and its optimisation. In this context, optimisation refers to the process of adjusting the model hyperparameters to reach the best performance on training data, while generalisation refers to the ability of the model to properly perform on never-seen data. This is a necessary topic to account in ML training because any approximation function $\hat{f}$ extracted from the hypotheses space, is affected by error, and thus it is necessary to find out the best compromise.

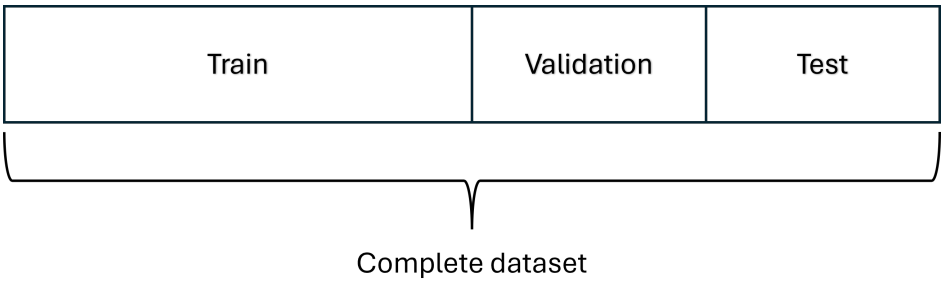


Figure 3.3: Example of how a dataset can be subdivided into training, validation and test subsets to properly train an ML model

3.1.2 Errors in function estimation

As introduced above, one should not expect an exact match between the outcome of the model and the ground truth value. Moreover, it is important for the model to have the ability to generalise. Generalisation is defined as the ability of a model to make accurate predictions, with respect to the expected outcome, over data that are not used during the training process. The generalisation accuracy of this is extremely important because may drive the choice of the learning method and give a metric to evaluate the quality of the model.

Despite the quality of the data and the complexity of the model, in function estimation, there is an error which cannot be avoided. Consider the assumption that the data are representing a function $\boldsymbol{y} = f(\boldsymbol{X})$ describing the relationship between $\boldsymbol{X}$ and $\boldsymbol{y}$. The goal of a ML model is to create an approximator $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{X})$ that can represent f. Anyway, all the ML models are affected by an error which cannot be predicted, and thus is always affecting the outcome, and thus the approximation of any ML model is of the form:

$$\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{X}) + \epsilon, \tag{3.1}$$

where ϵ represents the part which is not predictable. Such error can be thought to be given by the sum of two contributions: bias and variance [103].

By assuming a relation as the one in Eq. 3.1, the expected error of a predictor $\hat{f}$ in a point $x = x_0$ with $\mathbb{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$, where $\mathbb{E}$ represents the expected value and Var is the variance, can be derived. Using the min squared error (MSE) as a loss function (see Eq. 3.38), the expected prediction error can be represented as:

$$Err(x_0) = \mathbb{E}\left[\left(y - \hat{f}(\boldsymbol{x}_0)\right)^2 \mid \boldsymbol{X} = x_0\right]. \tag{3.2}$$

Given the definition of Var being:

$$\mathrm{Var}(X) \coloneqq \mathbb{E}\left[(X - \mathbb{E}(X))^2\right] = \mathbb{E}(X^2) - \mathbb{E}(X)^2, \tag{3.3}$$

with X being a random variable. After some calculation and using the definition 3.3 in Eq. 3.2, this becomes:

$$Err(\boldsymbol{X}) = \sigma_\epsilon^2 + \left[\mathbb{E}(\hat{f}(\boldsymbol{x}_0)) - f(\boldsymbol{x}_0)\right]^2 + \mathbb{E}\left[(\hat{f}(\boldsymbol{x}_0)) - \mathbb{E}(f(\boldsymbol{x}_0))\right]. \tag{3.4}$$

By remembering the definition of Bias

$$\text{Bias}(\hat{f(x)}, f(x)) = \mathbb{E}(\hat{f(x)}) - f(x), \tag{3.5}$$

and using it in Eq. 3.4 it is possible to obtain:

$$Err(x) = \sigma_\epsilon^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \tag{3.6}$$

$$= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}. \tag{3.7}$$

In Eq. 3.7, the first term represents the error which cannot be avoided in estimating a function, no matter how well it is represented, unless $\sigma_\epsilon^2 = 0$. The second term is the squared bias, that is the squared value of the amount by which the average of the estimations differ from the true mean. It is generally given by the erroneous assumption made in the learning algorithm or by the restriction of the hypotheses space. When the bias of an estimator is high, the hypothesis is said to underfit the data. The last term represents the variance, a measure of dispersion. The variance of an estimator represents the amount of change in the prediction due to fluctuations in the training data. When the variance of an estimator is high, it is paying too much attention to the training dataset, it is referred to as an overfitted estimator. Generally speaking, if the generalisation error is measured via MSE, increasing the complexity of the model the Bias decreases and the Variance increases, as can be seen in Fig. 3.4. The best trade-off is reached with a model complexity that allows to reach the minimum total error.

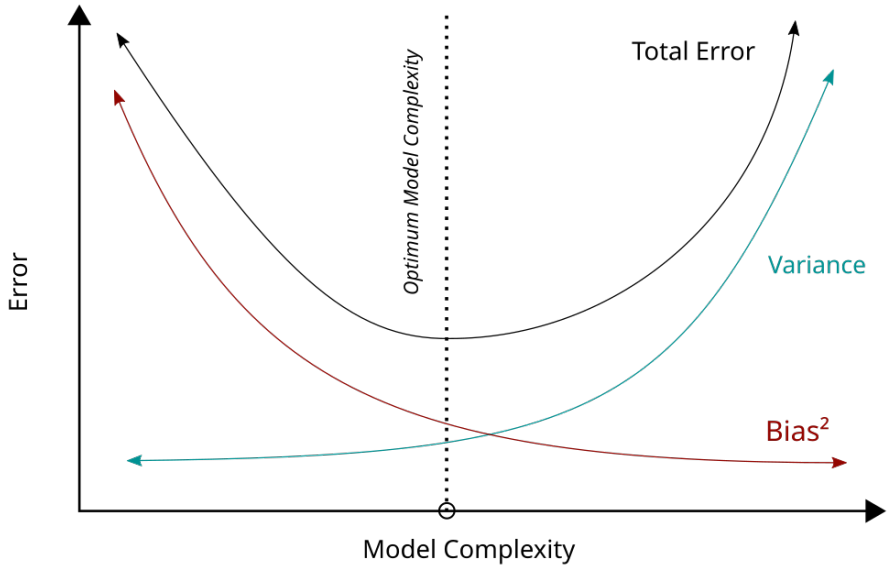


Figure 3.4: Representation of the Bias, Variance and total error as a function of the model complexity

In the remaining parts of this Section, it is reported the mathematical formulation of the algorithm used to develop the work about the cell capacity prediction. These are LR, RF and GBDT.

3.1.3 Linear regression

Linear regression is a ML method for modelling a relationship between a set of inputs and a set of outputs under the hypotheses of linear dependence between them (or, at least, a reasonable approximation). Given a set of independent variables $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ where each $\boldsymbol{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ is a column vector representing a feature, and a set of dependent variables (or labels) $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}^\mathsf{T}$, the linear model can be expressed as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\theta}_0 \tag{3.8}$$

where $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ represent the matrix composed of m examples and n features, $\mathbf{y} \in \mathbb{R}^m$ is the column array of the targets, $\boldsymbol{\theta} \in \mathbb{R}^n$ is the column vector of the parameters and $\boldsymbol{\theta}_0 \in \mathbb{R}^n$ is the bias vector [104]. It is important to point out that for the current treats, it is considered the case of multidimensional input and monodimensional output. In the case of multidimensional output, the treat would be the same, but the output array is a matrix, which can be expressed as $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k\}$, $\mathbf{y} \in \mathbb{R}^m$; $\mathbf{Y} \in \mathbb{R}^{m \times k}$ The objective is to estimate the parameters $\theta_i \in \boldsymbol{\theta}$ that satisfy the Equation 3.8. This can be done by minimising the residual sum of squares (RSS), defined as:

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{3.9}$$

By considering Eq. 3.8 and by including the bias array in the parameter array, it can be written as:

$$\mathrm{RSS}(\theta) = \sum_{i=1}^{n} (y_i - \boldsymbol{X}_i\theta)^2, \tag{3.10}$$

where y is the ground truth and $\hat{y}$ is the predicted value. By re-writing in matrix notation (since it is easier to characterise), it becomes:

$$\mathrm{RSS}(\boldsymbol{\theta}) = (\mathbf{y} - \boldsymbol{X}\boldsymbol{\theta})^\mathsf{T} (\mathbf{y} - \boldsymbol{X}\boldsymbol{\theta}) . \tag{3.11}$$

By differentiating with respect to $\boldsymbol{\theta}$ becomes:

$$\begin{aligned} \frac{\partial RSS}{\partial \boldsymbol{\theta}} &= -2\boldsymbol{X}^\mathsf{T}(\mathbf{y} - \boldsymbol{X}\boldsymbol{\theta}), \\ \frac{\partial^2 RSS}{\partial \boldsymbol{\theta}\, \partial \boldsymbol{\theta}^\mathsf{T}} &= 2\boldsymbol{X}^\mathsf{T}\boldsymbol{X}. \end{aligned} \tag{3.12}$$

Since $RSS(\boldsymbol{\theta})$ is a quadratic function of the parameters $\boldsymbol{\theta}$, it is possible to state that always exists a minimum, but such minimum might not be unique. To ensure the existence and uniqueness of the minimum point, it is necessary to assume that $\boldsymbol{X}$ is a full rank matrix, which means that $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ is non-singular and positive definite. In this condition it is ensured that the minimum point is unique and can be determined by forcing the first derivative to zero:

$$\boldsymbol{X}^\mathsf{T}(\mathbf{y} - \boldsymbol{X}\boldsymbol{\theta}) = 0, \tag{3.13}$$

and by inverting the system

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^{-1}\boldsymbol{X}^\mathsf{T}\mathbf{y}. \tag{3.14}$$

The predicted values for an input $\boldsymbol{X}$ can thus be determined as:

$$\hat{\mathbf{y}} = \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathsf{T}}\mathbf{y}. \tag{3.15}$$

The advantage of using a linear model is given by is its simplicity and clarity. Moreover, the linear model allows an easy comprehension of the relationships existing between the input and the output. It is also highly explainable, giving the possibility to easily understand the reasons for a certain output given the input. On the other hand, the main limitation is represented by the fact that LR considers the relationship as perfectly linear. A representation of the outcome obtained from the linear model is reported in Figure 3.5. There, it is possible to observe that the simple linear regression with only one independent variable results in a straight line. By generalising, it is possible to say that when the output dimension is enlarged up to the n-th, the outcome of the model becomes a hyperplane with dimension n.

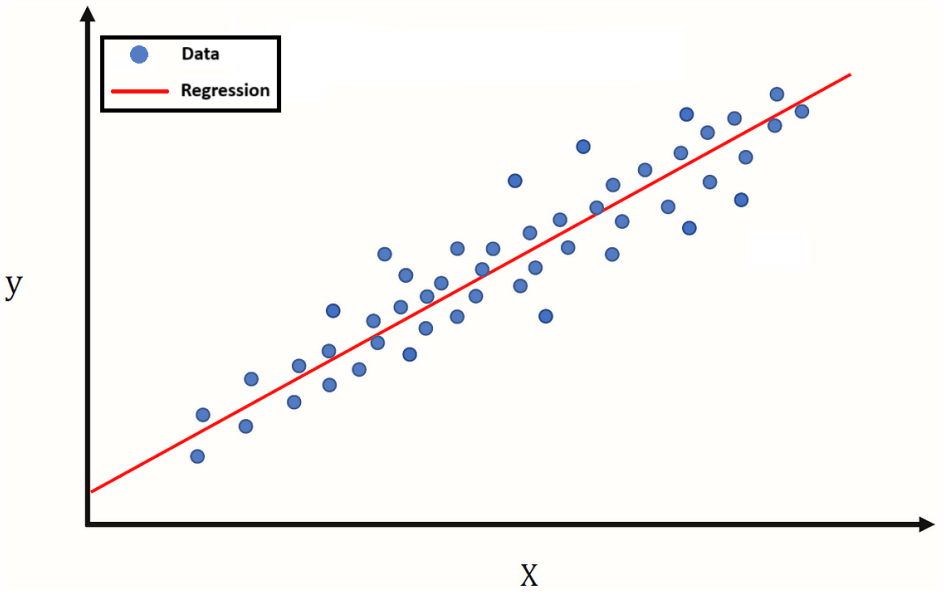


Figure 3.5: Representation of a simple univariate linear regression

3.1.4 Ensemble methods

The other two ML models used to realise the work about cell capacity estimation described in this thesis, are Random Forest Regression (RF) and Gradient Boosting Decision Tree (GBDT). These two models belong to a ML paradigm called *Ensemble Learning*. Ensemble methods are a class of models that combine the prediction coming from several base estimators (often referred to as *weak*) to improve the robustness of the final outcome compared to the one coming from a single estimator. The resulting model built this way, is generally more accurate than any of the individual weak estimators composing the model itself [105]. The basic estimator can be chosen to be any kind of model such as decision tree (DT) or NN. In the context of the work about capacity estimation, it is applied to DTs, and it is used to create RF and GBDT.

In the next sections, the main working principle of DTs is illustrated. Then the concept of RFand GBDT and the methodologies to build up these models are reported.

To realise the work reported in this thesis, it is used the Python library *scikit-learn*, a free and open-source machine learning library for Python, supporting supervised and unsupervised learning [106]. The mathematical formulation reported here is about the training algorithms used by the library.

Tree-based methods

Since in the context of this work, the ensemble learning methodology is applied to DTs to implement RF and GBDT, before treating these it is essential to introduce and explain what a DT is and how it works. DT is a class of non-parametric, supervised learning algorithms that creates a model to predict the value of one or more target variables by learning decision rules inferred from the data. This is realised by building up a hierarchical structure of which a general representation is reported in Fig. 3.6. There, it is possible to observe that a DT is made by four main components:

- Root node, which does not have any way to import data, works as an input node;
- Branches, paths followed by data feeding child nodes;
- Decision nodes, deriving from other nodes and conducting evaluations on sub-datasets coming from parent nodes;
- Leaf node, that is the final node which makes the estimation.

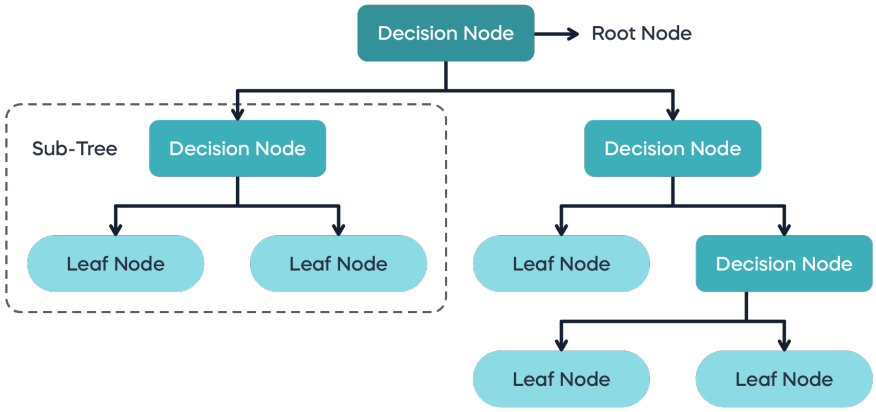


Figure 3.6: General representation of a Decision Tree structure

Over the years, different algorithms have been developed to implement and grow a DT:

- Iterative Dichotomiser 3 (ID3) [107]. The algorithm creates a multiway tree for classification by finding for each node the feature that yields the largest information gain. Trees are grown to their maximum size and then are pruned to improve their ability to generalise on unseen data. The algorithm begins with the whole

dataset in the root node and, at any iteration, it calculates the entropy H [1]or the information gain (IG)[2], looking for its minimum or maximum respectively. Based on the values, the set is split to produce subsets of the original data. Recursion continues until a stopping condition is met. The main limitation of this algorithm is given by the fact that the features must be categorical, ID3 can't work with numerical features. Moreover, ID3 does not guarantee an optimal solution since it can converge on local minima and can easily overfit;

- C4.5 is the successor of ID3, the major difference is given by the fact that the constraints on the features is removed giving the possibility to use also numerical variables. The algorithm converts the trained tree into sets of if-then rules;
- C5.0 is the last version of the above and is distributed under a proprietary licence. It is more memory efficient and accurate than the others;
- Classification and Regression Tree (CART) [108], which differs from the two above since supports numerical variables as targets (regression). It creates binary trees using the features and threshold that yield the largest IG.

The library used in this work to implement the ensemble algorithm is *scikit-learn*, which makes use of an optimised version of the CART algorithm. Because of that this thesis it is only reported the mathematical formulation of the CART [106].

To give an intuition about the working principle of decision trees, consider a binary regression problem, with inputs x_1 and x_2 and output a continuous variable y. The DT learning, is made by dividing the feature space into smaller sections. For each section, the outcome of the DT can be modelled as the average of the observations belonging to that particular region. The result of the division of the feature space is represented in Fig. 3.8. There the feature space is divided in 5 different regions $R_1, R_2, \ldots, R_5$ each determined by choosing 4 threshold values $t_1, \ldots, t_4$. The regression model built up this way, predicts $\hat{y} = \hat{f}(\boldsymbol{x})$ with a constant value c_m in the region R_m as:

$$f(x) = \sum_{m=1}^{5} c_m I\big((x_1, x_2) \in R_m\big), \tag{3.17}$$

where I is the indicator function, c_m is the average of the observation belonging to the region R_m. An indicator function of a subset S of a set is a function that maps the elements belonging to the considered subset to one and all other elements to zero, it can be defined as:

$$I(x) \coloneqq \begin{cases} 1 \text{ if } x \in S \\ 0 \text{ if } x \notin S \end{cases} . \tag{3.18}$$

The model built up can be represented as in Fig. 3.8 As can be seen, the whole

[1]In information theory (IT), the entropy of a random variable is the average amount of uncertainty inherent to the variable's possible outcomes. Given a random variable $X \in \chi$, distributed as $p : \chi \to [0, 1]$, it is defined as:

$$H(X) \coloneqq -\sum_{x \in X} p(x) \log(p(x)) \tag{3.16}$$

[2]In ML, IG is used as a synonym of mutual information. It is a measure of the dependence between two random variables. It quantifies the amount of information obtained for one variable by observing the other.

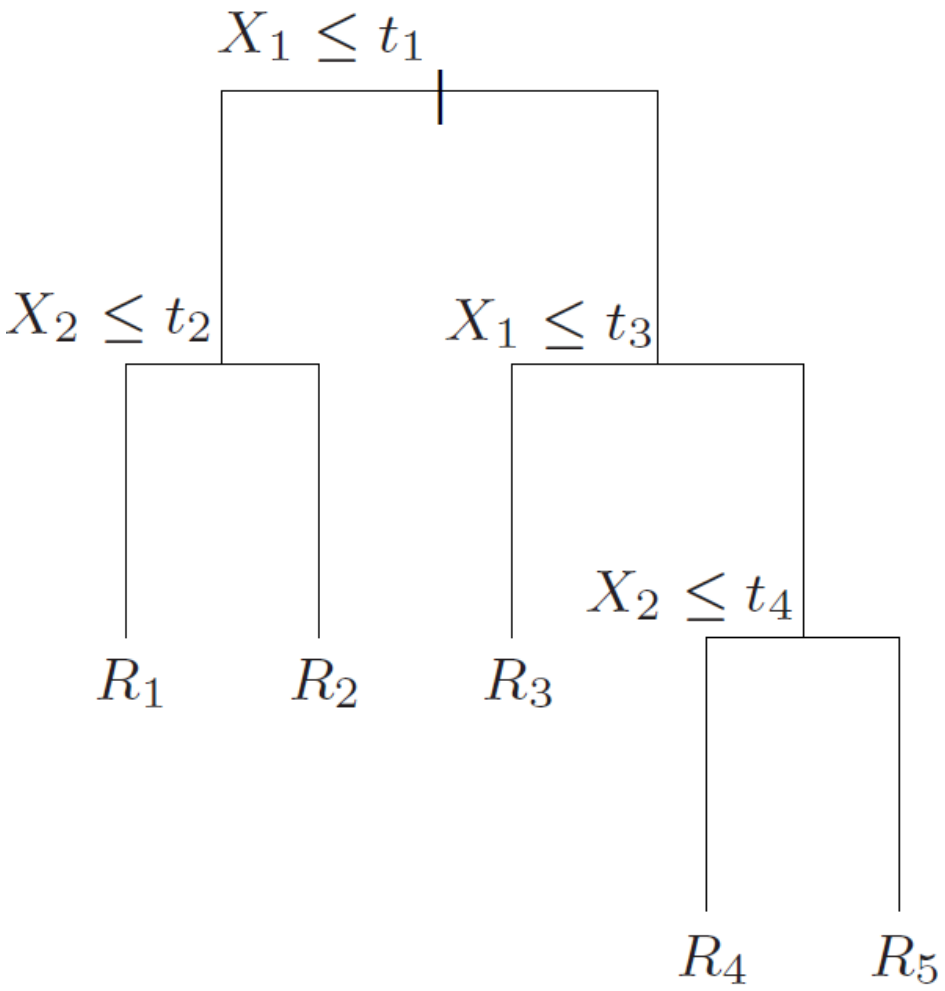


Figure 3.7: Regression tree corresponding to the partition represented in Fig. 3.8

dataset, at the beginning, is at the top of the tree (root node). The observations are then split based on some conditions up to the leaf nodes which correspond to the regions the feature space is divided into. The process of splitting is iterative and continues until a stopping rule is verified.

As introduced, Scikit-learn uses CART algorithm to build up a regression tree. To explain the algorithm, suppose to have an initial dataset $\boldsymbol{X}$ composed of n features and label dataset containing a response y for each of the m observations, such that each observation is a couple $(\boldsymbol{x}_i, y_i)$ with $i = 1, \ldots, n$ and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{in})$. Given this condition, the feature dataset is $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ and the label one is $\boldsymbol{y} \in \mathbb{R}^m$. The algorithm has to automatically find splitting variables splitting points and the topology the tree should have in order to build up the tree itself. Suppose, also, to have a feature space divided into k regions $R_1, \ldots, R_k$, thus the model response is modelled by a constant c_i in each region as:

$$f(x) = \sum_{i=1}^{m} \hat{c}_i I(\boldsymbol{x} \in R_i). \tag{3.19}$$

By adopting the minimisation of the sum of squares as a criterion (see Eq. 3.9), the predicted best constant $\hat{c}_i$ is the average of the observations y_i belonging to the region

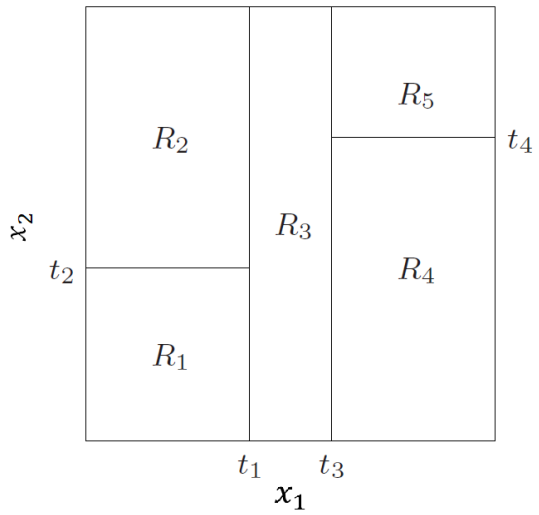


Figure 3.8: Representation of the partition made by DT in the bi-dimensional feature space.

R_i, is given by:

$$\hat{c}_i = \frac{1}{n_i} \sum_{y \in R_i} y_i, \tag{3.20}$$

with n_i number of observations in the region R_i. The procedure of finding the best $\hat{c}_m$ in terms of the minimum sum of squares on the whole dataset would be computationally intensive since would be necessary to consider every possible partition of the feature space. To reduce the complexity, it is possible to proceed with a greedy algorithm to make a locally optimal choice. Starting with all the data (at the root node), it is chosen one splitting variable x_j and a splitting threshold for that variable s, used to divide the starting feature space into two half-hyperplanes, such that:

$$R_1(\boldsymbol{x}_j, s) = \{\boldsymbol{x} | \boldsymbol{x}_j \leq s\} \text{ and } R_2(\boldsymbol{x}_j, s) = \{\boldsymbol{x} | \boldsymbol{x}_j > s\}. \tag{3.21}$$

Then seek for the splitting variable x_j and the splitting threshold s such that each region deriving from the split is as purer than possible which means having the most values respecting the condition, this can be done as:

$$\min_{\boldsymbol{x}_j, s} \left[\min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2)^2 \right], \tag{3.22}$$

That for any choice of j and s the inner minimisation is solved by

$$\hat{c}_1 = \frac{1}{n_1} \sum_{y \in R_1(j,s)} y \text{ and } \hat{c}_2 = \frac{1}{n_2} \sum_{y \in R_2(j,s)} y. \tag{3.23}$$

For each splitting variable $\boldsymbol{x}_j$, the determination of the splitting threshold s can be done quickly, and thus it is possible to scan all the combinations to find out the best

pair $(\boldsymbol{x}_j, s)$ This splitting procedure is recursively repeated on all the resulting regions until a stop condition is met. Tree size, is a tuning hyperparameter which governs the model complexity and the optimal size should be adaptively determined from the data. A large tree may overfit, while a too-small tree may not capture the most important information available in the data. To mitigate these effects, the strategy adopted is that of creating a very large tree and stopping the process only when some stopping criteria are met. Then the large tree is pruned using the *cost-complexity pruning*, which reduces the dimension of a tree and avoids its overfitting. To prune a tree, it has to be defined the expected complexity measure $C(T)$ of a tree $T \subset T_0$ has to be defined as a subtree of the original tree T_0 by removing any number of internal nodes which are not terminal. The complexity of the resulting tree T is measured by the number of its terminal nodes and can be indicated as $|T|$. For a regression problem, the cost-complexity measure is defined as:

$$C_\alpha(T) = C(T) + \alpha|T|, \tag{3.24}$$

where $R(T)$ is the sum of squared errors at the region R_m:

$$C(T) = \sum_{i \in R_m} (y_i - \hat{c}_m)^2. \tag{3.25}$$

The term α is the complexity parameter that controls the trade-off between the tree's complexity and fit. Large values of α result in smaller trees, and vice versa, small values of α results in large trees. For each α, there is a unique subtree T_α that minimises the C_α. To find the tree, it is recursively collapsed the internal node that produces the smallest pre-node increases in cost $C(T)$, and continues until it is produced by the root node. This gives a finite number of subtrees, and it is possible to show that the sequence must contain T_α. Once the optimal α is determined, it is used to prune the initial tree T to obtain $T(\alpha)$.

DT has several advantages that make it very useful for regression problems. The main advantages include the automatic feature selection, making DT highly insensitive to useless features, computational efficiency also for large datasets, automatic handling of unknown values in the features, possibility to handle both numerical and categorical variables, insensitivity to the scale of the feature and, last but not least, the explainability which allow to easily extract the relationship between the inputs and the output. On the other hand, DT for regression has low accuracy in several domains given by the piecewise constant approximation provided to build the tree and they are unstable with respect to small changes on the data [109].

Gradient Boosting Decision Tree

The goal of function approximation is to find an estimator $\hat{f}(\boldsymbol{X})$ that maps $\boldsymbol{X}$ to $\boldsymbol{y}$ such that the expected value $\mathbb{E}$ of a specified loss function is minimised, over the joint distribution of all the $(\boldsymbol{x}_i, y_i)$ values. Given a training dataset $(\boldsymbol{X}, \boldsymbol{y})$ composed of random input and labels, the problem goal is to find an approximator function $\hat{f}$, such that over the joint distribution of all $(y, \boldsymbol{x})$ values, the expected value of a specified loss function is minimised

$$f(\boldsymbol{X}) = \arg\min_{f(\boldsymbol{X})} \mathbb{E}_{\boldsymbol{y},\boldsymbol{X}}\, L(\boldsymbol{y}, \hat{f}(\boldsymbol{X})). \tag{3.26}$$

Boosting is one of the most powerful learning methodologies. It consists of constructing additive regression models by sequentially fitting simple base learners to actual residuals by least squares at each iteration. The residuals on which the subsequent learners are trained are the gradients of a certain loss function with respect to the model values at each training data point. The generic additive expansion for the approximation of a function is of the form

$$\hat{f}(\boldsymbol{X}; \beta_j, \boldsymbol{a}_j) = \sum_{j=1}^{m} \beta_j h(\boldsymbol{X}; \boldsymbol{a}_j). \tag{3.27}$$

There, the function $h(\boldsymbol{X}; a_j)$ is generally a parametrised function of the input variables $\boldsymbol{X}$ and the parameters $\boldsymbol{a}$. Here it is reported the case when the function is a tree produced through the CART algorithm, and thus, by using the same terminology of Par. 3.1.4, the base learner can be written as function of $\boldsymbol{X}$ and $\boldsymbol{s}$, where $\boldsymbol{X}$ is the feature matrix and $\boldsymbol{s}$ containing splitting variables. It assumes the form $h(\boldsymbol{X}; \boldsymbol{s}_j)$. Boosting approximate the function f by an expansion of the form:

$$\hat{f}(\boldsymbol{X}) = \sum_{j=1}^{m} \beta_j h(\boldsymbol{X}, \boldsymbol{s}_j), \tag{3.28}$$

where β represents the expansion coefficient and $h(\boldsymbol{X}, \boldsymbol{s}_j)$ represents the base learner. In boosting, the expansion coefficient and the parameters $\boldsymbol{s}$ are fit to the training data in a forward stage-wise manner. To do this it is necessary to begin with a guess on $\hat{f}(\boldsymbol{X})$. Let the initial guess be $\hat{f}_0(\boldsymbol{X})$, and then for any $j = 1, 2, \ldots, m$:

$$(\beta_j, \boldsymbol{s}_j) = \arg\min_{\beta, \boldsymbol{s}} \sum_{j=1}^{n} L(y_j, \hat{f}_{m-1}(\boldsymbol{x}_j) + \beta h(\boldsymbol{x}_j, \boldsymbol{s})), \tag{3.29}$$

and

$$\hat{f}_m(\boldsymbol{X}) = \hat{f}_{m-1}(\boldsymbol{X}) + \beta_m h(\boldsymbol{x}, \boldsymbol{s}_m) \tag{3.30}$$

Gradient boosting solves the Eq. 3.29 for arbitrary differentiable loss function $L(y, \hat{f}(\boldsymbol{X}))$ with a two-step procedure. The first step consists in fitting the weak learner $h(\boldsymbol{x}_j, \boldsymbol{s}_j)$ by least-squares:

$$\boldsymbol{s}_j = \arg\min_{\boldsymbol{s}, \rho} \sum_{j=1}^{n} \left(\tilde{y}_{jm} - \rho h(\boldsymbol{X}, \boldsymbol{s}_j)\right)^2, \tag{3.31}$$

to the current residuals

$$\tilde{y}_{jm} = - \left[\frac{\partial L(y_j, \hat{f}(\boldsymbol{X}_j))}{\partial \hat{f}(\boldsymbol{X}_j)} \right]_{\hat{f}(\boldsymbol{X}) = \hat{f}_{m-1}(\boldsymbol{X})}, \tag{3.32}$$

then, given the estimator $h(\boldsymbol{X}, \boldsymbol{s}_j)$, the optimal value of the coefficient β_j is calculated as:

$$\beta_j = \arg\min_{\beta} \sum_{j=1}^{n} L(y_j, f_{m-1}(\boldsymbol{X}_j) + \beta h(\boldsymbol{X}_j, \boldsymbol{s}_m)). \tag{3.33}$$

The approach described above replaces a function optimisation problem as the one in Eq. 3.29 with a problem based on least-squares (see Eq.3.31) followed by a parameter

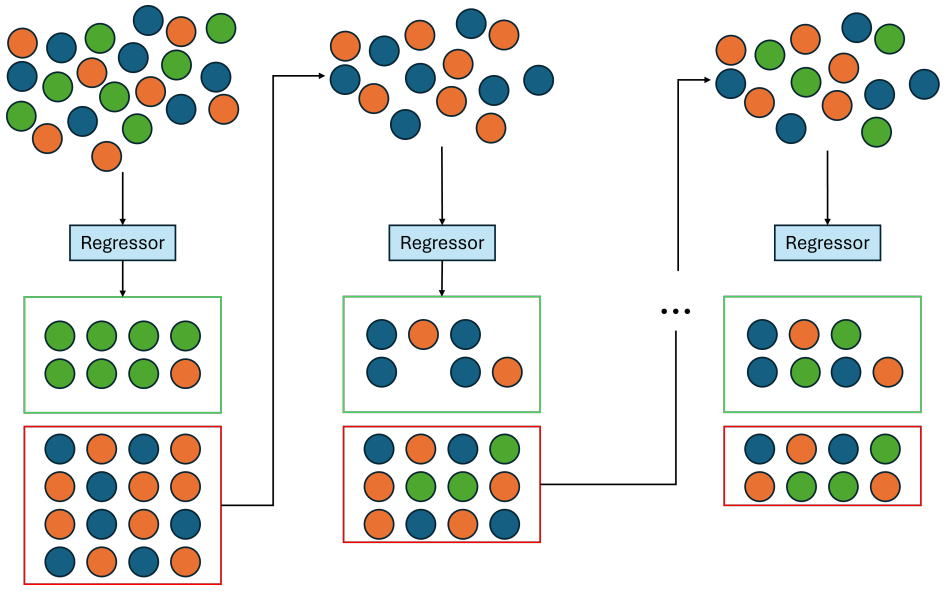


Figure 3.9: Example of a boosted gradient for GBDT implementation.

optimisation as reported in Eq. 3.33 based on a general loss function L. A schematic representation of the working principle is represented in Fig. 3.9.

At each step, the solution tree is the one that minimises Eq. 3.31 given the current model $\hat{f}_{m-1}(\boldsymbol{x}_j)$

The final approximation is given by:

$$\hat{f}_m(\boldsymbol{X}) = f_{m-1}(\boldsymbol{X}) + \nu c_m I(\boldsymbol{X} \in R_m), \tag{3.34}$$

where ν represents the shrinkage parameter. It is defined such that $0 < \nu < 1$.

Random Forest Regression

The other ensemble model used in this work is Random Forest Regression. As for the GBDT, also Random Forest Regression combines multiple weak learners to create a robust and accurate model [110] reducing its variance. Random Forest is composed of a collection of DTs, where each tree makes an individual prediction. The final prediction is then determined by taking the average, or the weighted average, of the predictions from all the trees. The idea in random forest is to reduce the variance by a reduction in the correlation between the trees. This is achieved with a tree-growing process through random selection of the input variables. Before continuing with the description of random forest it is necessary to introduce the concepts of *bootstrap* and *bootstrap aggregation (bagging)*. Bootstrap provides a way to assess uncertainty by sampling from the training data. Suppose here to have a training dataset $\boldsymbol{Z} = z_1, \ldots, z_n$, with $z_i = (x_i, y_i)$. Bootstrap consists of extracting b datasets of dimension n with replacement from the original training dataset and fitting a chosen model for each of the b random datasets. Bagging consists of training a regressor for each of the b bootstrapped datasets, as represented in Fig.3.10, and making the average of the predictions of any regressor. For each boost sample k, $k = 1, \ldots, b$, a model is fitted

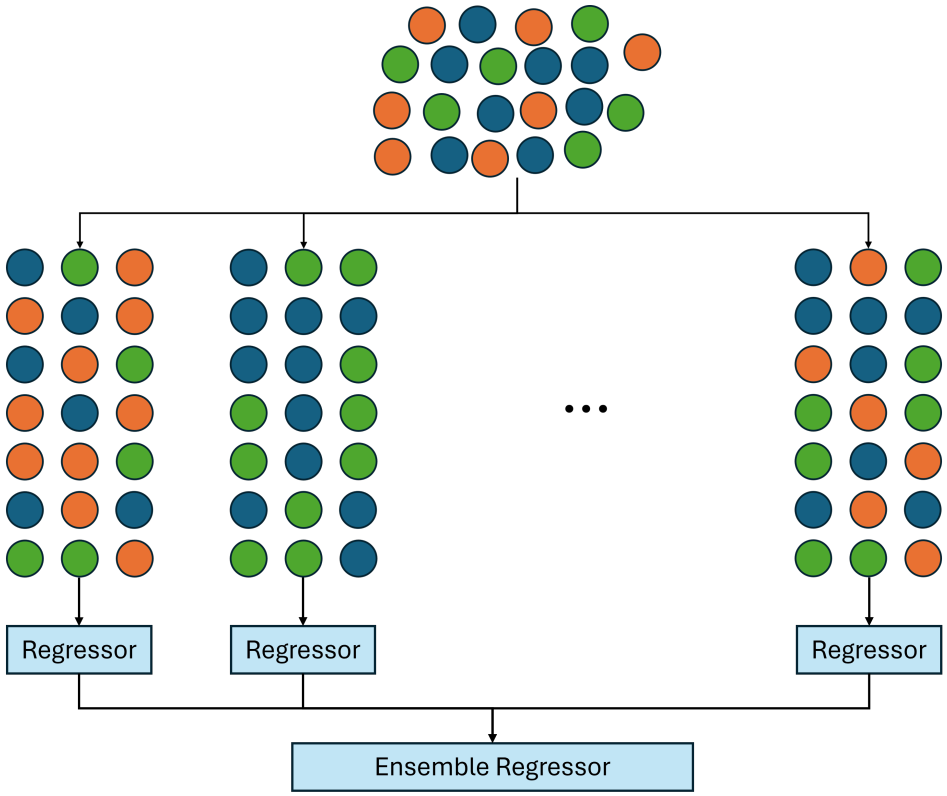


Figure 3.10: Example of bagging for random forest implementation.

whose prediction is $\hat{f}^*(\boldsymbol{X})$. The bagging estimation is thus given by:

$$\hat{f}_{bag}(\boldsymbol{X}) = \frac{1}{b}\sum_{k=1}^{b}\hat{f}_k^*(\boldsymbol{X}). \tag{3.35}$$

Regression trees are the ideal candidate for bagging because if grown sufficiently deep, have relatively low bias. Moreover, since each tree generated in boosting is identically distributed, the expectation of an average of b trees is the same as the expectation of any one of them. That means that the bias of the bagged tree is the same of any single tree and thus the only improvement can be made on the variance

By considering b identically distributed random variables, each with variance σ^2, has a variance of $\frac{1}{b}\sigma^2$, if they have a positive pairwise correlation ρ, the variance of the average is given by:

$$\rho\sigma^2 + \frac{1-\rho}{b}\sigma^2. \tag{3.36}$$

As b increases, the second term decreases. Thus, the size of the correlation between the pairs of bagged trees limits the benefits of the averaging. Random forest is used to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance, and this can be achieved in the growing process through random selection of the input variables. After that b of such trees are grown, the random forest regression prediction will be of the form:

$$\hat{f}_{rf}(\boldsymbol{X}) = \frac{1}{b}\sum_{i=1}^{b}T(\boldsymbol{X}, \boldsymbol{s}_i), \tag{3.37}$$

where $\mathbf{s}_i$ is the parameters array of the $i-th$ tree in the random forest, containing splitting pints, splitting nodes and terminal-nodes values.

3.2 Deep learning

In the previous Chapter, it is given an overview of the main concepts of ML together with the mathematical formulation of the algorithms used to realise the work about capacity prediction presented in this thesis. In the following paragraphs, it is given an introduction to deep learning, what it is and which is the mathematics behind this concept. Eventually, a deep overview of the LSTM algorithm is reported.

Deep learning is a specific subgroup of machine learning that puts emphasis on learning successive layers of increasingly meaningful representation. These layers are learned through models called neural networks and are stacked on top of each other. The adjective "deep" refers to the fact that multiple layers are stacked to build up the model. This particular architecture is thought to have long computation paths that allow variables to interact in complex ways being sufficiently expressive to capture real-world complexities for many important kinds of learning problems.

3.2.1 Technical notes of Neural Networks

A deep feedforward network is the base of deep learning modelling. As for the application of ML, the goal is to approximate a function $\boldsymbol{y} = f(\boldsymbol{X})$ by defining a mapping function $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{x}, \boldsymbol{\theta})$ and determine the parameters $\boldsymbol{\theta}$ that results in the best function approximation. These models are called feedforward neural networks. The name arises from the fact that it is not present in any connection that feeds back the output into the network itself. When this feedback exists, they are called recurrent neural networks (RNNs). The world network is given because they are typically represented by composing many different functions. For example, having three different functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$, they are connected in a chain to form $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$. This is the most common structure of a neural network where the $f^{(1)}$ is called the first layer, $f^{(2)}$ is the second layer and $f^{(3)}$ is the third layer and so on. A graphical representation of a general NN is reported in Fig. 3.11 The overall number of layers of the chain is called

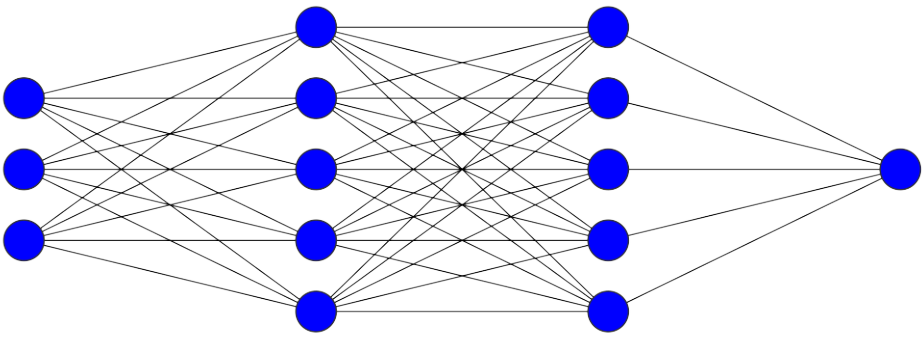


Figure 3.11: General scheme of a feed-forward neural network. This particular architecture uses one input layer $\in \mathbb{R}^3$, two hidden layers $\in \mathbb{R}^5$ and one output layer$\in \mathbb{R}^1$

depth of the model and the number of nodes of each intermediate layer determines the *width* of the network. The behaviour of the other layers is not directly specified, the training algorithm has to decide how to use these layers in order to match the output. Since the intermediate layers don't show the desired output, these layers are referred

to as *hidden layers*. By supposing that the intermediate layers of the NN provide a set of hidden features defined by $\boldsymbol{h} = f(\boldsymbol{x}, \boldsymbol{\theta})$, the role of the output layer is to transform these hidden features to complete the task the NN is built for. With reference to Fig. 3.11, the network represented is composed of an input layer, two hidden layers defining the NN depth and one output layer. As can be seen, each hidden layer is composed of 5 nodes, this defines the width. As briefly introduced, the objective of the training is to derive a function $\hat{f}(\boldsymbol{x})$ to approximate $f(\boldsymbol{x})$. Each example $\boldsymbol{x}$ is paired with a label $y = f(\boldsymbol{x})$ which drives the learning path of the neural network. Neural networks can be thought to be efficient non-linear function approximators which use gradient descent to minimize an error function, they are designed to achieve statistical generalisation. Neural networks are thought to extend the limited capability of linear models, despite that linear models are easy, computationally cheap and can be fitted efficiently, they are limited to linear function only, and thus cannot be used in these contexts where the relation is not linear.

One way to model the non-linear function of an input $\boldsymbol{X}$ is to apply a linear function to a transformed input $\phi(\boldsymbol{X})$, where ϕ is the non-linear transformation which gives a new representation of $\boldsymbol{X}$. The strategy applied by using DL, is to learn the function ϕ. The approach consists in having an estimator $\hat{\boldsymbol{y}} = \hat{f}(\boldsymbol{X}; \boldsymbol{\theta}, \boldsymbol{w})$ and making a parametrisation as $\phi(\boldsymbol{X}, \boldsymbol{\theta})\boldsymbol{w}^{\mathsf{T}}$. The parameters $\boldsymbol{\theta}$ are used to learn the particular ϕ from a wide class of function and the parameters $\boldsymbol{w}$ are used to map $\phi(\boldsymbol{x})$ to the output. This is an example of a hidden network with ϕ defining a hidden layer. Deep learning, have introduced the concept of hidden layer. To properly design each hidden layer, it is required to choose an *activation function* that is used to compute the output of each node. Training a NN, is not much different from training any other algorithm of ML with gradient descent. The main difference is given by the fact that the non-linearities occurring in neural networks, cause some loss functions to be non-convex. This translates in using an iterative gradient-based optimisation algorithm to drive the cost function to very low values, instead of applying convex optimisation algorithms with global convergence guaranteed. Gradient descent (or stochastic gradient descent) applied to non-convex functions, has no such convergence guaranteed and is sensitive to the values of the initial parameters. Most of the algorithms used to train NN are a particular refinement of the gradient descent algorithm.

An important aspect in designing a NN, is the choice of the cost function. Generally, the cost function is made up by combining a primary function with a regularisation term. Despite the wide range of applicability for NNs, in the context of this work the model is used for regression problems, and thus, to learn conditional statistics of $\boldsymbol{y}$ given $\boldsymbol{X}$. Suppose to have a predictor $\hat{f}(\boldsymbol{X}, \boldsymbol{\theta})$ and use it to predict $\boldsymbol{y}$. If the NN is sufficiently powerful, it is possible to think that it is able to represent any kind of function $\boldsymbol{y} = f(\boldsymbol{x})$ described by the features collected in a matrix $\boldsymbol{X}$. In this case, it is possible to build up a cost function that has its minimum value when the approximator corresponds to the function that maps $\boldsymbol{X}$ to the expected value of $\boldsymbol{y}$. The introduction of a cost function is important to define a methodology to assess the performance of the model, which is generally evaluated as a measure of the deviation of the model outcome from the ground truth. The goal of the training is to minimise such function to have outcomes as similar to the ground truth as possible. The choice of the loss function is crucial for the proper working of the model because:

- It drives the model during the training phase, and it is the basis for the optimisation process. It is used by gradient descent algorithms to optimise the model's parameters;
- It provides a benchmark to evaluate model performance;
- The type of loss function affects the learning, including how fast the model parameters are updated and what kind of errors are penalised more heavily.

The most used cost functions for regression problems are MSE, RMSE and mean absolute error (MAE).

MSE, is defined as the average of the squared difference between the predicted values and the actual values:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{3.38}$$

The MSE function is easy to understand and is differentiable, and thus it is suitable for gradient-based optimisation problems. Due to the presence of the square, using MSE as a loss function means that larger errors are more penalised. This can ensure that the trained model does not have large error predictions. RMSE is defined as the square root of the MSE, it offers the same intuition behind the MSE but is measured with the same units of the predicted variable. Its mathematical definition is:

$$\text{RMSE} = \sqrt{(\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2)}. \tag{3.39}$$

MAE, instead, is defined as the absolute value of the difference between the predicted value and the ground truth as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|. \tag{3.40}$$

The advantage of using MAE as a loss function is given by the fact that all the errors are weighted the same, and thus there is no effect of magnifying the larger errors. The choice of the cost function is tightly related to the choice of the output layer.

Another important choice to be made in defining NN, is that of choosing the activation function giving the output of the nodes for each of the hidden layers. The design of hidden units is still a research area and there is not a theoretical guide for this choice [103]. Most of the hidden units composing a NN, can be described as accepting an input array $\boldsymbol{X}$, computing an affine transformation as,

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{X} + \boldsymbol{b}, \tag{3.41}$$

and then apply an element-wise activation function $g(\boldsymbol{z})$. Depending on the task the NN is built for, different activation functions can be used for the output layer, such as linear or sigmoid. Actually, there is not a general approach to select the correct activation function for the particular task, it can only be done through a trial and

error approach and by using the intuition given by the knowledge of the functions. The most typical choice for feedforward neural networks is the rectified linear unit (ReLU), defined as:

$$g(x) = \max(x, 0). \tag{3.42}$$

The plot of ReLU is represented in Fig. 3.12. The advantages of using ReLU activation

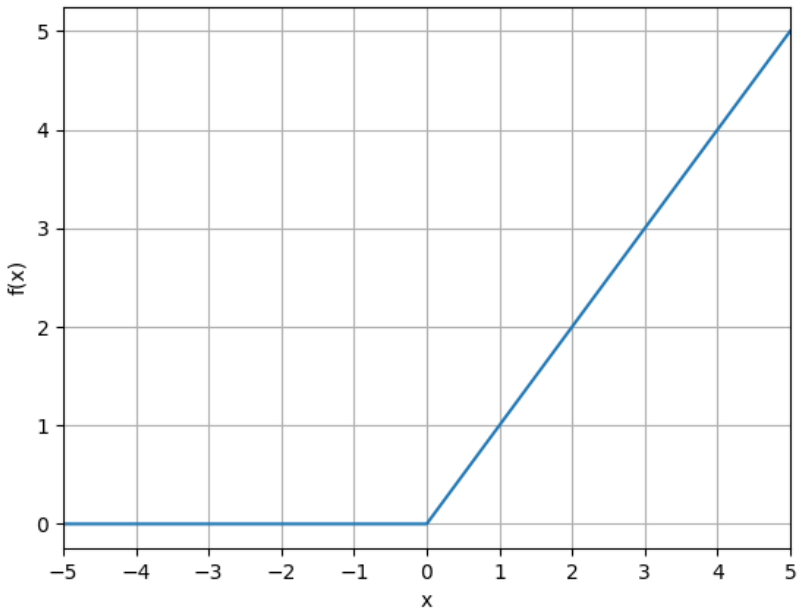


Figure 3.12: Plot of the ReLU

function consist in having an efficient computation since it is very similar to the linear unit. The only difference is that ReLU outputs zero for half of the domain. By analysing differentiation, the ReLU definition allows to have a derivative which is large whenever the unit is active and a null second derivative anywhere. This means that the gradient direction is more useful for learning than it would be for different activations with different peculiarities. It is to be noted, moreover, that the ReLU, as occurring for other activation functions, is not differentiable at any point. This means that the units cannot learn with gradient-based methods on examples where their activation is zero. Anyway, since it is not expected to train to reach a point in which the gradient is exactly zero, this is acceptable for the minima of the cost function, to correspond to points with an undefined gradient. To overcome such limitation, different generalisations of the ReLU can be used to guarantee the gradient exists in any point, as in leaky ReLU, absolute value rectification or parametric ReLU.

Before the introduction of ReLU, activation functions for neural networks were sigmoid and hyperbolic tangent. The sigmoid function is a special form of the logistic function: a set of S-shaped curves. It is defined as

$$g(x) = \sigma(x) = \frac{1}{1 + e^{-x}}. \tag{3.43}$$

As Fig. 3.13 shows, this particular function saturates to 1 for large values of x and to 0 for small values of x, and it's extremely sensitive only when the input is close to zero. The advantage of using a non-linear function as activation, such as the sigmoid

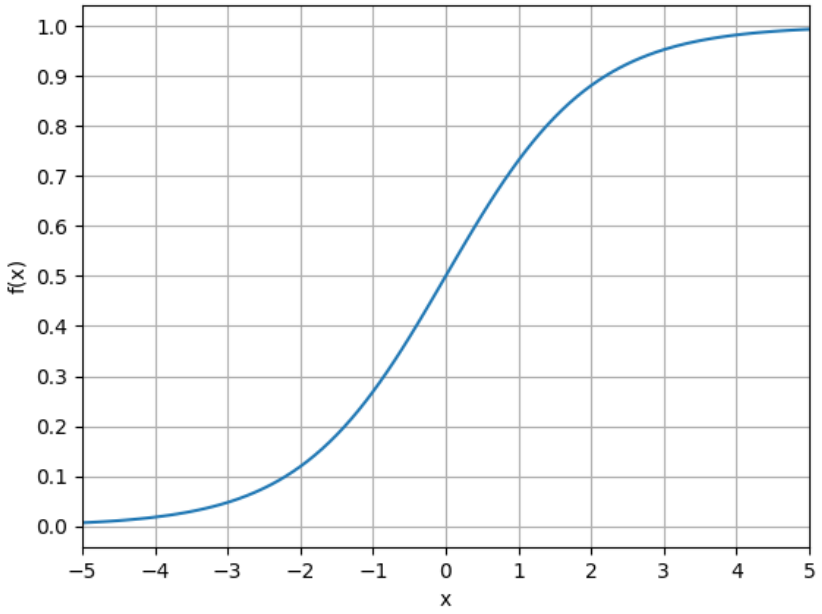


Figure 3.13: Plot of the sigmoid function

function, is the possibility of including the non-linearities in the function representation, and thus having the possibility to represent non-linear domains. Anyway, the saturation can make gradient-based learning very difficult, for that reason its use as an activation function for hidden units is not recommended. When it is necessary to use a non-linear function, the hyperbolic tangent usually performs better than the sigmoid. The definition of the hyperbolic tangent is given by

$$g(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{3.44}$$

and its graph is represented in Fig. 3.14. The hyperbolic tangent function allows to map the inputs to a domain in the range $(-1, 1)$. The advantages, compared with the sigmoid, are given by the fact that tanh is symmetric around the origin, which can be helpful also in case the input data has both positive and negative values, and it is zero-centered. The latter characteristic makes tanh similar to the identity function while the input is close to zero ($\tanh(0) = 0$) and thus, in that region it can resemble training a linear function, making the training easier.

Another key consideration in designing a neural network is the choice of its architecture. It is the overall structure of the neural network, such as defining how many layers compose the NN and how many units for each layer should be used. Most NN arrange layers stacked one upon the other creating a chain structure. In this organisation, each layer is a function of the preceding layer, as can be seen in Fig. 3.11. By

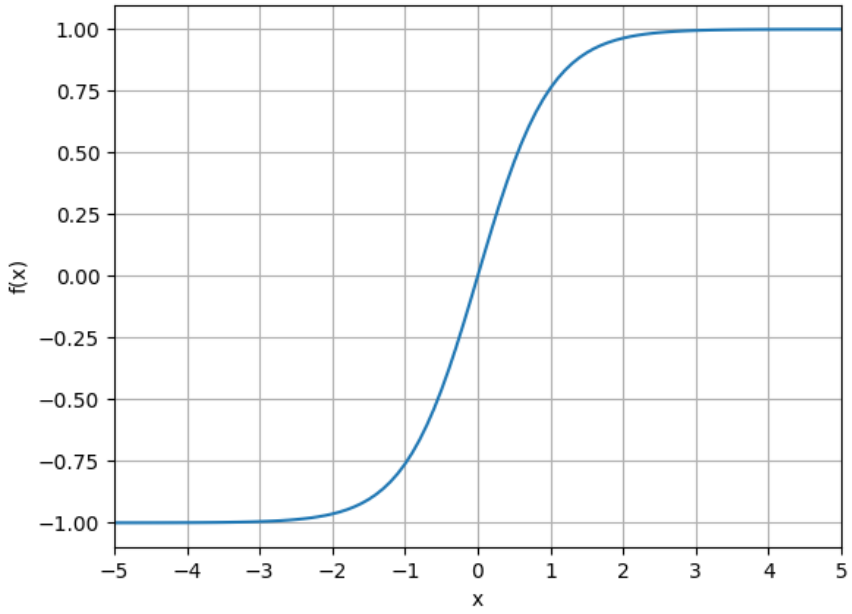


Figure 3.14: Plot of the hyperbolic tangent function

referring to the configuration displayed in Fig. 3.11, the first hidden layer is given by

$$\boldsymbol{h}^{(1)} = g^1(\boldsymbol{W}^{(1)T}\boldsymbol{x} + \boldsymbol{b}^{(1)}), \tag{3.45}$$

the second layer is given by

$$\boldsymbol{h}^{(2)} = g^2(\boldsymbol{W}^{(2)T}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}). \tag{3.46}$$

Eventually, the output of the network can be represented, by supposing a linear activation function for the output layer, as

$$\hat{y} = \boldsymbol{W}^{(3)T}\boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)}. \tag{3.47}$$

If the number of layers is greater, the general representation of the $i-th$ hidden layer of a neural network is

$$\boldsymbol{h}^{(i)} = g^i(\boldsymbol{W}^{(i)T}\boldsymbol{h}^{(i-1)} + \boldsymbol{b}^{(i)}), \tag{3.48}$$

where the superscript (i) is the number of the hidden layer. Choosing the architecture of the neural network is a time-consuming task since there is not a general rule which gives some guidance in this direction.

The objective of a NN is to learn a non-linear function to map input to output. At the first hypothesis, it is possible to think that the task can be solved only with a specialised model family for the non-linearity to be represented. Anyway, it is possible to demonstrate that a feedforward network with hidden layers provides a universal approximator. This is guaranteed by the universal approximation theorem which ensures that, regardless of the complexity of the function to be represented, any measurable function closed and bounded in a subset of $\mathbb{R}^n$ (also said Borel measurable), can be

approximated with a neural network with any desired non-zero error provided that the network is large enough. Moreover, the network has to have a linear output and, at least, one hidden layer with non-linear activation (such as sigmoid or tanh functions). It is to be pointed out that the theorem only ensures the existence of such an approximator, without ensuring that the NN is able to learn that function. Learning can fail due to inability of the learning algorithm to find the proper value of the parameters or because of the overfitting on the training dataset. It only ensures that there exists a network large enough to represent any function with the desired accuracy but doesn't give any indication about its dimension.

While using a feedforward neural network taking an input $\boldsymbol{x}$ and giving a prediction $\hat{\boldsymbol{y}}$, information flows in one direction only, forward from the input layer toward the output layer. This process is called forward propagation. During the training phase, data flows from the input layer toward the output layer following the entire NN structure and a cost function $\boldsymbol{L}(\theta)$ is produced. At this point, to properly vary the parameters, it is necessary to compute the gradient of the loss function $\boldsymbol{L}(\theta)$ with respect to the weights, and propagate it from the output layer toward the input layer. This last operation is called back-propagation and allows using a gradient descent algorithm (or its variants such as stochastic gradient descent) to perform training using such gradients. For simplicity in the representation, here it is considered the case where a NN only have a single output target. Anyway, the backpropagation can be easily extended to the multidimensional case of many output variables. Back-propagation algorithm moves the gradient from the output layer backwards toward the input layer by using the chain rule of calculus. Let $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$ be both arrays, $f : \mathbb{R}^n \to \mathbb{R}$ mapping from $\mathbb{R}^n$ to real numbers and $g : \mathbb{R}^m \to \mathbb{R}^n$. Suppose $\boldsymbol{y} = g(\boldsymbol{x})$ and $\boldsymbol{z} = f(g(\boldsymbol{x})) = f(\boldsymbol{y})$. The chain rule states that

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}. \tag{3.49}$$

In vectorial notation can be written as

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \nabla_{\boldsymbol{y}} z, \tag{3.50}$$

where $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ is the Jacobian of g. The back-propagation algorithm consists of performing the Jacobian-gradient product reported in Eq. 3.50 for each operation in the neural network.

As briefly introduced earlier, back-propagation allows the use of the gradient descent algorithm to train neural networks. Gradient descent is an optimisation algorithm whose task is to maximise or minimise an objective function $f(\boldsymbol{x})$ by varying $\boldsymbol{x}$. For the case of a neural network, as the one described here, the objective function is represented by the cost function, and thus, gradient descent is used to minimise such function. To minimise the function f, it is necessary to find the direction in which the function itself decreases the fastest, this can be done by finding the directional derivative. The directional derivative in direction $\boldsymbol{u}$, is the slope of the function f in direction u. Its definition states that

$$\min_{\boldsymbol{u}, \boldsymbol{u}^{\mathsf{T}}\boldsymbol{u}=\boldsymbol{1}} \boldsymbol{u}^{\mathsf{T}} \nabla_x f(\boldsymbol{x}), \tag{3.51}$$

which can be written as

$$\min_{\boldsymbol{u},\boldsymbol{u}^\mathsf{T}\boldsymbol{u}=\boldsymbol{1}} |\boldsymbol{u}||\nabla_x f| \cos(\alpha) \tag{3.52}$$

where α is the angle between the gradient and the direction $\boldsymbol{u}$. By considering $|\boldsymbol{u}| = 1$ ($\boldsymbol{u}$ is considered as the unit vector) the Eq. 3.52 simplify to

$$\min_u \cos(\alpha), \tag{3.53}$$

which is minimised when the $\boldsymbol{u}$ points in the opposite direction of the gradient. That means that f can be decreased by moving in the direction of the negative gradient. This methodology is known as gradient descent. After one step, the gradient descent proposes a new point as

$$\boldsymbol{x}' = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}), \tag{3.54}$$

where ϵ is the *learning rate*, a positive scalar determining the size of the step. The algorithm converges when every element of the gradient is very close to zero. Most of the learning algorithms are powered by the stochastic gradient descent (SGD) algorithm, which is an extension of the gradient descent algorithm. It is the most used optimisation algorithm for training neural networks, in particular deep learning models. Such an extension is useful in particular when dealing with large datasets since is based on the fact that it is possible to obtain an unbiased estimation of the gradient by taking the average gradient on a minibatch $\mathbb{B}$ of m samples drawn from the dataset. Generally, the cost function of the learning algorithm can be decomposed as a sum over training examples. For example, the MSE can be written as

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{y}}^{(i)})^2 = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{y}^{(i)} - \boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}^{(i)})^2, \tag{3.55}$$

for these additive cost functions, gradient descent requires computing the gradient

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} 2(\boldsymbol{y}^{(i)} - \boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}^{(i)})\boldsymbol{x}^{(i)}, \tag{3.56}$$

by using the examples taken from the subset $\mathbb{B}$ of the complete dataset $\mathbb{D}$. The parameters of the neural network are then updated as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \tag{3.57}$$

As with many machine learning algorithms, also neural network training involves optimisation, a highly complex and time-consuming task. Moreover, when dealing with neural networks, there exists also the difficulty given by a non-convex objective function which can translate into several challenges. The most prominent is ill-conditioning of the Hessian matrix $H(f(\boldsymbol{x}))$. The matrix is defined as the collection of all the second derivatives of the function. It is defined such that:

$$H(f(\boldsymbol{x}))_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}). \tag{3.58}$$

Hessian ill-conditioning manifests itself by causing SGD to stuck, in the sense that even very small steps increase the cost function.

One of the most interesting characteristics of a convex function and a convex optimisation problem is that it can be reduced to the problem of finding a local minimum, given that any local minimum for a convex function is guaranteed to be a global minimum. When dealing with non-convex functions as in NN, it is possible to have many local minima that may represent a problem if they have a high cost compared with the global minimum. This is still an open problem in literature, but most of the experts believe that for sufficiently large neural nets, most of the local minimums have acceptable low-cost functions, and thus it is not important to find a global minimum, as can be seen in Fig. 3.15.

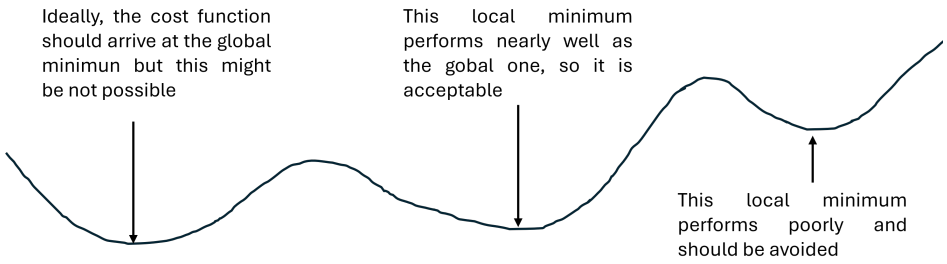


Figure 3.15: Example of the possibility of accepting local minimum points instead of global minima

Another problem with high-dimensional non-convex functions is the presence of saddle points. A saddle point is characterised by a null gradient and can be seen as a minimum point for the function along one cross-section and a maximum for another cross-section of the same function, as represented in see Fig. 3.16. The effect of a saddle point in optimisation is still unclear. For first-order optimisation algorithms that use only gradient information, it seems that gradient descent can rapidly escape the flat region or a saddle point [111]. The same can be seen when approaching maxima, which from an optimisation perspective are the same as saddle points and thus gradient-based algorithms are not attracted to them. Major problems may arise in large flat locations since both gradient and Hessian are zero.

All the problems presented until now, correspond to properties of the loss function at a single point. Anyway, it can be difficult to overcome all of these problems and still perform purely if the direction that results in the most improvement locally, does not point toward regions of much lower cost.

3.2.2 Optimisers

Despite the SGD remains a very popular optimisation algorithm, learning with it can be very slow. A way to accelerate learning is the method of momentum. The algorithm introduces a new variable v that plays the role of velocity: it is the direction and the speed with which the parameters move in the parameter space. A hyperparameter $\alpha[0, 1)$ determines the contribution of the velocity on the previous gradients. The

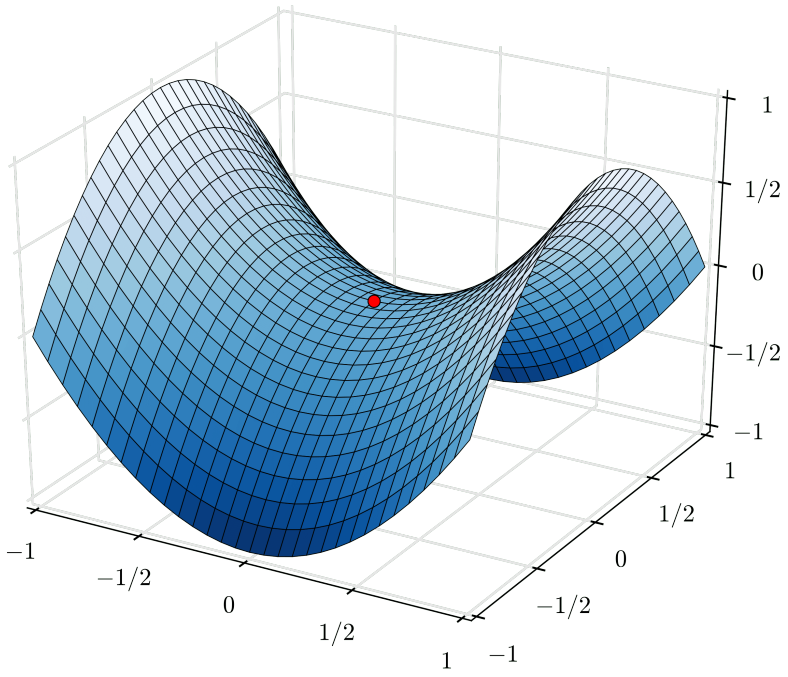


Figure 3.16: Representation of a function with a highlighted saddle point

parameters update rule becomes:

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \nabla_\theta \left(\frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \right), \tag{3.59}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}, \tag{3.60}$$

as can be seen from Eq. 3.59, velocity accumulates the gradients and the larger α with respect to ϵ, the more previous gradients affect the current direction. The advantage of this algorithm is given by that the step size depends on how large and aligned a sequence of gradients are. The step size is largest when many successive gradients point in exactly the same direction. The effect of momentum can be seen in Fig. 3.17, where the blue arrows represent the SGD and the red arrows represent the momentum algorithm. Researchers have realised that the learning rate is one of the most difficult

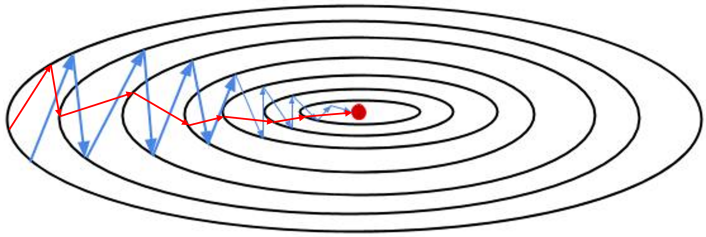


Figure 3.17: Schematic comparison between the SGD and the momentum algorithm

hyperparameters to set because it has a significant impact on model performance and thus, they have introduced methods to adapt the learning rate of model parameters.

Another popular optimiser is AdaGrad (adaptive gradient), which is a modified version of the stochastic gradient descent with a per-parameter learning rate to help improve the convergence performance over standard stochastic gradient descent. The general learning rate ϵ is modified at each time step t for every parameter θ_i based on the past gradients as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta_t} - \frac{\epsilon}{\sqrt{\boldsymbol{G}_t + \epsilon}} \boldsymbol{g}_t, \tag{3.61}$$

with $\boldsymbol{G}$

$$\boldsymbol{G} = \sum_{\tau=1}^{t} \boldsymbol{g}\boldsymbol{g}^{\mathsf{T}}, \tag{3.62}$$

and $\boldsymbol{g}$ is the gradient of the function with respect to the weights at the iteration τ

$$g = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)}) \tag{3.63}$$

The formula to update the paraments becomes:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - - \frac{\epsilon}{\sqrt{\boldsymbol{G}_t + \epsilon}} \odot \boldsymbol{g}. \tag{3.64}$$

The effect is an increased learning rate for sparse parameters and a decrease for ones that are less sparse. In the context of convex optimisation, AdaGrad allows greater progress in the more gently sloped directions of parameter space. However, the accumulation of squared gradients from the beginning of the training can result in a premature excessive decrease in the learning rate.

A modification to the AdaGrad [112] which performs better in the context of non-convex functions is the RMSProp (Root Mean Square Propagation) algorithm. Here, the learning rate is adapted for each of the parameters as in AdaGrad but using an exponential decaying average to discard the most old history. Here, the learning rate is adapted through a moving average of the squared gradients. The running average is calculated as:

$$E(\boldsymbol{g}^2)_{t+1} = \gamma E(\boldsymbol{g}^2)_t + (1 - \gamma) \left(\frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x}, t) \right)^2, \tag{3.65}$$

where γ is called the decay factor. The parameters are updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\epsilon}{\sqrt{(\boldsymbol{g}^2)_t}} \left(\frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x}, t) \right). \tag{3.66}$$

It is an effective optimisation algorithm for deep neural networks.

The most popular adaptive learning optimisation algorithm is the Adam (Adaptive Moments). It can be seen as a combination of the RMSProp and momentum algorithm with important distinctions. It incorporates bias correction to the estimation of both the first-order moment and the second-order moment. The weight update is performed as

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \epsilon \frac{\hat{m}}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.67}$$

with

$$\hat{m} = \frac{m_t}{1-\beta_1^t} \tag{3.68}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3.69}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\boldsymbol{g} \tag{3.70}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\boldsymbol{g}^2 \tag{3.71}$$

with β_1 and β_2 forgetting parameters. Adam is generally seen as being robust to the choice of hyperparameters, though the learning rate sometimes needs to be changed from the suggested default value.

3.2.3 Sequence modelling

A difficulty that neural networks have to overcome, consists in dealing with data that presents a sequential ordering instead of a tabular ordering. Initially, this limitation arose in trying to process sentences or speeches with plain neural networks. IT is caused by the absence of recurrences in the model which feed information of the output back to the input of the network. In this context, a family of networks which can help to solve the problem is the Recurrent Neural Network (RNN). That is a family of NN specialised for processing sequential data, for which the appearance order is important. The main idea behind RNNs, is the possibility to share the same parameters across different parts of the model, which is not possible with feedforward networks. Feedforward neural networks, can only map form input to output, while with RNNs is, at least in principle, possible to map the entire history of previous input to each output [113]. A general representation of a RNN is displayed in Fig. 3.18. There, the arched arrow represents the recurrent connection, which is the key point that allows RNN to have a sort of "memory" of previous input to persist in the internal state and thereby influence the network output.

Before explaining the working principle of RNNs, it is important to introduce the concept of computational graph. It is a way to formalise the structure of a set of computations. Despite many ways of formalising computational graphs are possible, in the context of this thesis each node of the graph represents a variable and each connection between nodes represents an operation, which is a function of one or more variables. The concept of graph results is particularly useful when dealing with RNN since gives the possibility to represent graphically the sequential property of RNNs through the unfolding.

To simplify the explanation of the working principle, here it is considered a single self-connected hidden layer, with hyperbolic tangent as the activation function for the hidden layer neuron and linear for the output layer, as the one represented in Fig. 3.19a. RNNs can be designed by considering different architectures, the most important are:

- RNN that produces an output for each time step and has recurrent connections between hidden units, as displayed in Fig. 3.19b;
- RNN that produce an output for each time step and have recurrent connection only form the output of one-time step to the hidden unit of the next time step as displayed in Fig. 3.19c;

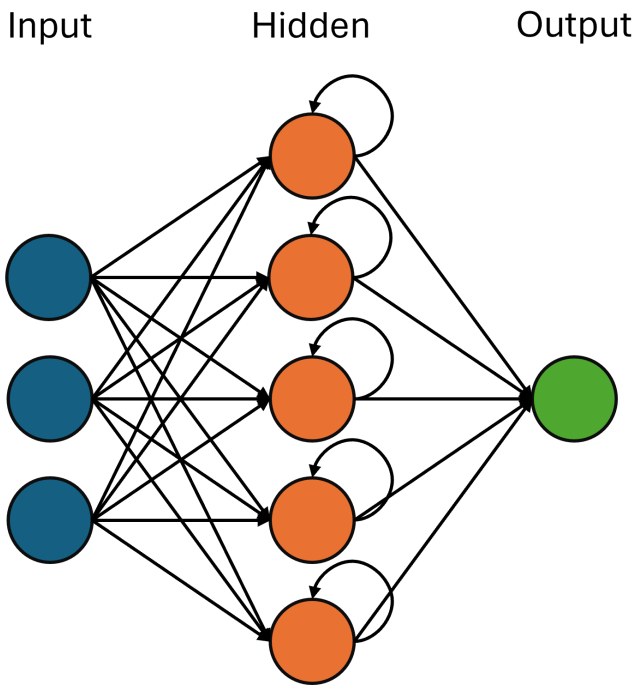


Figure 3.18: General scheme of a recurrent neural network

- RNN with a recurrent connection between hidden units that read an entire sequence to produce a single output, as displayed in Fig. 3.19d.

Suppose to have a model as the one reported in Fig. 3.19b. For that configuration, the forward propagation operation begins with a specification of the initial hidden state $\boldsymbol{h}^{(0)}$, then for each time step the update equations are given by:

$$\boldsymbol{a}^{(t)} = \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{b}_a \tag{3.72}$$

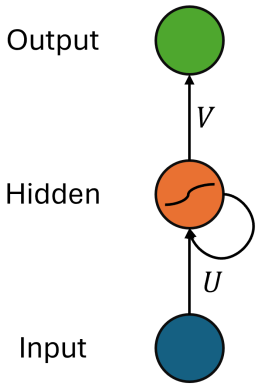
$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}) \tag{3.73}$$

$$\hat{\boldsymbol{y}}^{(t)} = \boldsymbol{V}\boldsymbol{h}^{(t)} + \boldsymbol{b}_o, \tag{3.74}$$

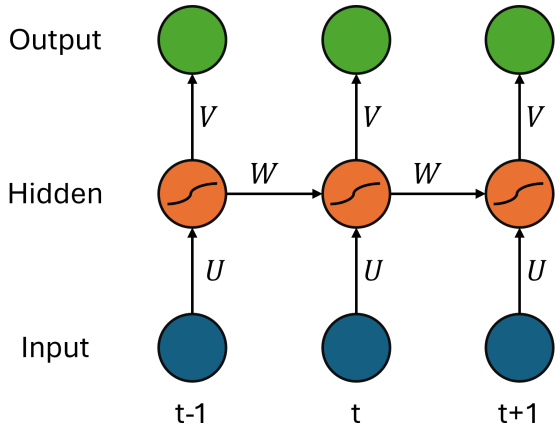
where $\boldsymbol{b}$ represents the bias vector and $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$ are the weight matrix for input-to-hide, hidden-to-hidden, hidden-to-output respectively, the equations (3.72), (3.73) and (3.74), refer to a network that maps an input sequence to an output sequence of the same length. The total loss for a given sequence of $\boldsymbol{x}$ values paired with a ground-truth sequence $\boldsymbol{y}$, is given by the sum of the losses over all the time steps

$$L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\tau}\sum_{t=1}^{\tau}(\boldsymbol{y}^{(t)} - \hat{\boldsymbol{y}}^{(t)})^2. \tag{3.75}$$

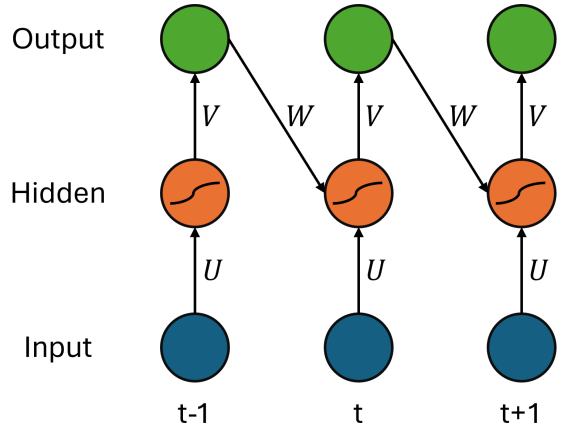
Computing the gradient in a RNN is straightforward. It can simply be applied the back-propagation algorithm to the unfolded graph of Fig. 3.19b, which is generally referred to as back-propagation through time (BPTT) algorithm, and then use any general-purpose gradient-based technique to train the RNN. Given a loss function L, it has to be computed the gradient of the loss function recursively for each node N composing



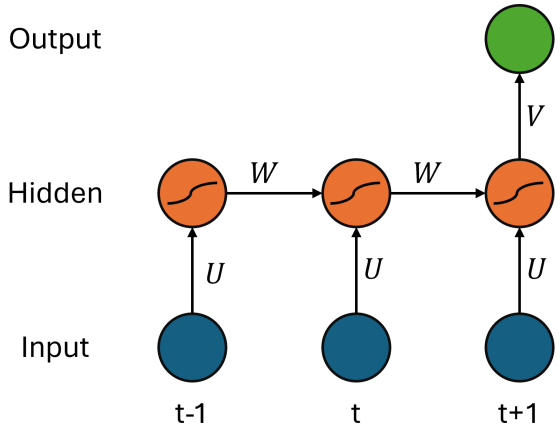
(a) Representation of a circuit diagram for an RNN.



(b) Schematic representation of an unfolded RNN that maps an input sequence to a corresponding output sequence.



(c) Schematic representation of an unfolded RNN whose only recurrence is the connection between the output and the hidden layer.



(d) Schematic representation of an unfolded RNN with a single output at the end of the sequence.

the computational graph. The recursion begins with the nodes immediately preceding the final loss

$$\frac{\partial L}{\partial L^{(t)}} = 1. \tag{3.76}$$

Then the gradient of the loss function with respect to the output is

$$\nabla_{\hat{\boldsymbol{y}}^{(t)}} L = \frac{\partial L}{\partial \hat{\boldsymbol{y}}^{(t)}} = \frac{2}{\tau} \sum_{t=1}^{\tau} (\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}), \tag{3.77}$$

then, going backwards for all the parameters composing the network, it results in:

$$\frac{\partial L}{\partial \boldsymbol{b}_o} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}^{(t)}} \frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{b}_o} = \nabla_{\hat{\boldsymbol{y}}^{(t)}} L \tag{3.78}$$

$$\frac{\partial L}{\partial \boldsymbol{V}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}^{(t)}} \frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{V}} = \nabla_{\hat{\boldsymbol{y}}^{(t)}} L \, \boldsymbol{h}^{(t)} \tag{3.79}$$

$$\frac{\partial L}{\partial \boldsymbol{h}^{(\tau)}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}^{(\tau)}} \frac{\partial \hat{\boldsymbol{y}}^{(\tau)}}{\partial \boldsymbol{h}^{(\tau)}} = \boldsymbol{V}^{\mathsf{T}} \nabla_{\hat{\boldsymbol{y}}^{(t)}} L \tag{3.80}$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{W}} = \boldsymbol{h}^{(t-1)} \nabla_{\hat{\boldsymbol{y}}^{(t)}} L \tag{3.81}$$

$$\frac{\partial L}{\partial \boldsymbol{U}} = \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{U}} = \boldsymbol{I} \left(1 - (\boldsymbol{h}^{t+1})^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)}, \tag{3.82}$$

it is to be noted that the Eq. 3.80 is valid only for the last time step τ. Iterating backward in time, the equation becomes:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{h}^{(t)}} &= \frac{\partial L}{\partial \hat{\boldsymbol{y}}^{(t)}} \frac{\partial \hat{\boldsymbol{y}}^{(t)}}{\partial \boldsymbol{h}^{(t)}} + \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial L} \\ &= \boldsymbol{V}^{\mathsf{T}} \nabla_{\hat{\boldsymbol{y}}^{(t)}} L \; + \boldsymbol{W}^{\mathsf{T}} \boldsymbol{I} \left(1 - (\boldsymbol{h}^{t+1})^2\right) \nabla_{\boldsymbol{h}^{(t+1)}} L, \end{aligned} \tag{3.83}$$

where $\left(1 - (\boldsymbol{h}^{t+1})\right)^2$is the Jacobian of the hyperbolic tangent associated with the hidden unit at time $t+1$ and $\boldsymbol{I}$ is the Identity matrix.

An important characteristic of the RNN is their ability to use past information to map an input sequence to an output. Anyway, standard RNN shows a difficulty with long-term dependences. These phenomena arise from the exponentially smaller weights given to long-term interactions compared with the short-term ones. This may cause the gradients to explode or vanish. Exploding gradient may lead to oscillating weights and vanishing gradients may increase learning time up to a prohibitive amount or does not work at all. From a mathematical perspective, the exploding or vanishing gradient limitation is given by the composition of the function employed by general RNN which involves the composition of the same function, once per time step. The function composition can be thought of as a matrix multiplication as

$$\boldsymbol{h}^t = \boldsymbol{W}^{\mathsf{T}} \boldsymbol{h}^{t-1}, \tag{3.84}$$

by recurrently applying the matrix multiplication, after τ time steps it can be described by the power function:

$$\boldsymbol{h}^{\tau} = (\boldsymbol{W}^{\tau})^{\mathsf{T}} \boldsymbol{h}^0. \tag{3.85}$$

If $\boldsymbol{W}$ admits eigendecomposition of the form $\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\mathsf{T}}$, the recurrence can be simplified as:

$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}\boldsymbol{\Lambda}^{(t)}\boldsymbol{Q}\boldsymbol{h}^0, \tag{3.86}$$

and thus, since the eigenvalues Λ are raised to the power of the time τ, they cause the exploding of those with a magnitude greater than one and the vanishing for those with a value lower than one. To overcome these limitations, over the years researchers made different attempts such as non-gradient-based training algorithms or different architectures. One of the most effective in overcoming this limitation is the LSTM, which is used here to implement the cell model.

3.2.4 Long Short-Term Memory

Over the years, different models have been developed to model long-term dependencies such as Echo State Network, the application of Leaky hidden units to standard RNNs, or again adding connections between the most distant time variables and the present values. These solutions are able to properly represent long-term dependencies. Anyway, once that information is used, it might be useful to forget the old state, and thus, an algorithm able to decide when to do it without the manual selection is needed. This is what gated RNNs do. The most effective model for sequence modelling belonging to the class of gated recurrent units is LSTM. The main idea at the base of the LSTM, is to create a path through time that has derivatives that neither vanish nor explode. This is created by using connection weights that may change at any time step.

The core idea of the LSTM, is the creation of a self-loop to create a path where the gradient can flow for a long time. A crucial evolution has been the addition of the possibility of making the weights of the self-loop conditioned by the context, rather than fixed. This allows a change of the scale of integration based on the input sequence and also for a model with constant parameters and can be done because the time constants are output by the model itself. The LSTM diagram is represented in Fig. 3.20. Instead of applying an element-wise nonlinearity to the affine transformation of inputs and recurrent units, LSTM cell presents an internal recurrence (the self-loop) in addition to the external one. Each cell has the same input of an ordinary RNN but presents a system of gating units that controls the flow of the information. The most important

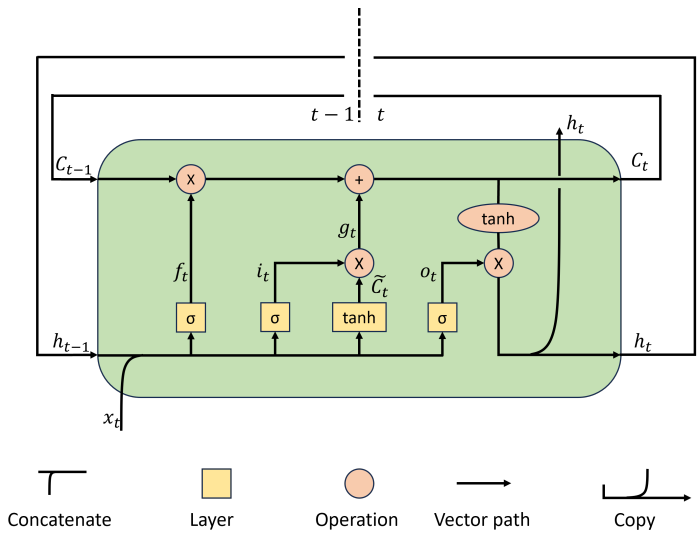


Figure 3.20: Representation of an LSTM cell

component is the cell state $C^{(t)}$ that has the weights controlled at any time step. Each LSTM cell contains three multiplicative units which are referred to as gates: input, forget and output gates. Forget gate is used to control which of the previous information is still important in the learning procedure, and thus they are added to the cell state. With reference to Fig. 3.20, for the time step t, the forget gate is updated

as f:

$$f^{(t)} = \sigma\left(\boldsymbol{b}^f + \sum_j \boldsymbol{U}^f \boldsymbol{x}_j^{(t)} + \sum_j \boldsymbol{W}_j^f \boldsymbol{h}_j^{t-1}\right), \tag{3.87}$$

where $\boldsymbol{U}$, $\boldsymbol{W}$, $\boldsymbol{b}$ represent the input weights, recurrent weights and the biases, $\boldsymbol{x}^{(t)}$ is the input array and $\boldsymbol{h}^t$ is the hidden layer vector at time instant t. The input gate's purpose is to control which part of the new information is important to add to the cell state, at the time step t it is updated as:

$$i^{(t)} = \sigma\left(\boldsymbol{b}^g + \sum_j \boldsymbol{U}^g \boldsymbol{x}_j^{(t)} + \sum_j \boldsymbol{W}_j^g \boldsymbol{h}_j^{t-1}\right) \tag{3.88}$$

$$\tilde{C}^{(t)} = \tanh\left(\tilde{\boldsymbol{b}} + \sum_j \tilde{\boldsymbol{U}} \boldsymbol{x}_j^{(t)} + \sum_j \tilde{\boldsymbol{W}}_j \boldsymbol{h}_j^{t-1}\right) \tag{3.89}$$

The output gate determines whether and how the memory cell should influence the output at the current time step, it is updated as:

$$o^{(t)} = \sigma\left(\boldsymbol{b}^o + \sum_j \boldsymbol{U}^o \boldsymbol{x}_j^{(t)} + \sum_j \boldsymbol{W}_j^o \boldsymbol{h}_j^{t-1}\right). \tag{3.90}$$

The internal cell state cell C is then updated as:

$$C^{(t)} = f^{(t)} \odot C^{(t-1)} + i^{(t)} \odot \tilde{C}^{(t)} \tag{3.91}$$

the output of the LSTM cell is then given by:

$$\boldsymbol{h}^{(t)} = \tanh(C^{(t)}) \odot o^{(t)}. \tag{3.92}$$

The advantages of using LSTM to model a sequence are given by the possibility of managing long sequences and removing the typical limitation affecting RNN of gradient vanishing or exploding. Anyway, the LSTM is computationally expensive compared to the conventional RNNs and requires more data to be properly trained avoiding the overfit on the training data. Moreover, the tuning of the parameters is complex and their explainability is almost null.

Chapter 4

Voltaic Efficiency for battery health monitoring

In the previous chapters, an introduction to the main concepts of ML and DL is given. Moreover, the general mathematical formulation for the algorithms employed to realise the works presented in this thesis has been presented. The purpose of this Chapter is to illustrate the procedure developed to face the problem of SOH estimation of a battery. In particular, here the problem is tackled by employing an innovative methodology which consists of using Voltaic Efficiency (VE) to compare the performance of an aged battery working on a car, with the performance of a new battery. The VE of the new battery is obtained with a ROM. To make the comparison possible both the model and the real battery experience the same cycle, over which the VE is calculated.

4.1 Introduction

Battery is a highly non-linear dynamical system where several physical phenomena occur concurrently. Moreover, the behaviour at the actual time step is affected by the working conditions experienced in the previous time instants, showing an inter-dependence between past and actual response, making the order with which data are organised in the dataset important. In particular, that order has to follow the chronological development of the events. This particular ordering methodology is referred to as sequence and the use of ML for its modelling is called sequence modelling. When data are ordered chronologically, as for the case under scrutiny, they are referred to as time series, that is, an ordered collection of data samples denoting events occurring at a specific time. The non-linear nature of the battery makes it a complex system to study and extract information from. This is particularly true for what concerns information about actual capacity and resistance, which are directly connected to the health of the battery itself. Given its characteristic of high power and high energy density, LIBs represents the dominant energy storage system used in the F1 environment. For this particular application field, the battery is extremely important since is a crucial part of the performance of the car, influencing the lap time and thus the possibility of winning races. Because of such importance, it is necessary to reduce as much as possible the uncertainties and predict as more accurately as possible which is the battery performance expected for the next race to optimise the energy deployment

with the aim of minimising the lap time. Despite the different methodologies developed to estimate the SOH available in the literature and presented in Chapter 2, these are not suitable to be applied in the F1 environment for different reasons, the principal of which is given by the fact that a lot of these methodologies are based on laboratory experiments or on a CC charging or discharging curve. Conditions that are not always available in the particular context.

This work arises from the necessity to have a simple and effective methodology to estimate the SOH of the battery race by race to properly manage the battery itself and to approach the races with a reduced probability of encountering uncertainties. The problem is faced here by the development of a data-driven ROM of a brand-new cell. The ROM goal is to predict the voltage of the cell given specific working conditions in terms of current (I), SoC and temperature (T). Here, the ROM is implemented by using a deep LSTM model. The Predicted Voltage (V_p) is used to calculate the VE of a battery (which can be seen as a series/parallel composition of cells) as if it were realised with n identical cells, all of them behaving as the one modelled through the ROM. The VE is then compared with the same indicator calculated using data directly measured from a battery on the car. The two VE are both calculated for the same working conditions and thus a direct comparison between them is feasible. The comparison between these indicators can be seen as a health indicator for the battery in the car, given that the comparison is made with a model representing a brand-new battery. The results of the study presented in the following paragraphs prove that the approach of comparing the VE of a running battery with that of a new similar one can be proficiently used as a health estimation. Given the lightness of the model, it is also suitable to be used on BMS. The rest of this Chapter is organised as follows: Section 2 provides the mathematical formulation of the different efficiencies defined for a battery. Section 3 reports the data collection process and its pre-processing implemented to obtain the final datasets employed to feed the model. Then, Section 4 reports information about the architecture of the LSTM used to implement the ROM model together with its validation on a test dataset. Eventually, it is shown the results obtained with the methodology described when applied to three different batteries actually mounted on three different cars.

4.2 Battery efficiency

As for all the systems, also in the battery energy losses occur. The energy retrieved after a full charge is less than the energy that has been stored by the battery itself. This phenomenon occurs because parasitic reactions are present within the electrochemistry of the cell. For batteries, three different efficiencies can be defined:

- Coulombic efficiency or Ah-efficiency (η_{Ah})
- Voltaic efficiency (η_U)
- Energy efficiency (η_{Wh})

Their mathematical formulation is respectively:

$$\eta_{Ah} = \frac{discharged\ Ah}{charged\ Ah} = \frac{1}{CF} = \frac{\int_{\Delta t} I\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} < 0 \\ I &= 0 \text{ if } I_{battery} \geq 0 \end{cases}}{\int_{\Delta t} I\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} \geq 0 \\ I &= 0 \text{ if } I_{battery} < 0 \end{cases}}, \tag{4.1}$$

$$\eta_{U} = \frac{\int_{\Delta t} IU_{battery}\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} < 0 \\ I &= 0 \text{ if } I_{battery} \geq 0 \end{cases}}{\int_{\Delta t} IU_{battery}\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} \geq 0 \\ I &= 0 \text{ if } I_{battery} < 0 \end{cases}} * CF, \tag{4.2}$$

$$\eta_{Wh} = \frac{\int_{\Delta t} IU_{battery}\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} < 0 \\ I &= 0 \text{ if } I_{battery} \geq 0 \end{cases}}{\int_{\Delta t} IU_{battery}\,\mathrm{d}t \begin{cases} I &= |I_{battery}| \text{ if } I_{battery} \geq 0 \\ I &= 0 \text{ if } I_{battery} < 0 \end{cases}}, \tag{4.3}$$

where CF is the Charge Factor: the ratio between the charges flowed into the battery in Ah and the charges extracted for the particular cycle. The Coulombic efficiency is the inverse of the expression of how much excess charge is applied to the battery. Voltaic efficiency describes the difference between the average voltage during discharge and charge and depends on the current and the temperature. Energy efficiency compares the energy extracted during discharging to the energy charged into the battery [36]. As can be seen from Eq. 4.1, 4.2 and 4.3, the battery efficiency can be determined from measured current and voltage.

4.3 Dataset

The first step in developing the ROM is that of collecting and properly processing the data to be used to train and test the data-driven model. The data used to develop the work described in this Section are kindly furnished by Ferrari S.p.A, and thus they are protected by a Non-Disclosure Agreement (NDA). To respect the agreement, the value of any quantity used in the context of this work is scaled with respect to an internal reference value. Consequently, the current, voltage, temperature and SoC become:

$$I_s = \frac{I_{te}}{I_r}, \qquad V_s = \frac{V_{te}}{V_r}, \qquad T_s = \frac{T_{te}}{T_r}, \qquad SoC_s = \frac{SoC_{te}}{SoC_r}, \tag{4.4}$$

where the subscript s indicates the scaled quantity and te is used to indicate the value measured and r is the internal reference value.

The data used to develop the ROM of the cell are measured from different vehicles, all of them equipped with the same measurement instrument. Given the particular application, it is not possible to define a complete discharge-charge cycle for the battery, mainly because it is never fully discharged or recharged. Therefore, here the life of the battery is measured in travelled distance with measurement unit to be in km. Data

collected from sensors are to be considered sequential, which means that the order in which they appear in the dataset is important since the value at the actual time instant depends upon the history of the system. In particular, for the case under scrutiny, data are ordered chronologically, and thus it is possible to refer to the dataset as a time series.

The purpose of the model is to represent the behaviour of an average cell composing the batteries which are used on the cars. Such a signal is not available in the data collected from the car, and thus it is necessary to preprocess them. To obtain the needed quantities, data coming from sensors are properly engineered to extract the quantities pertaining to a single cell. This is realised starting from data measured for the whole battery and then by considering the battery as a series/parallel combination of identical cells. Moreover, given the fact that the model has to represent a brand-new cell, it is mandatory to define a mileage providing enough data to train the model but, at the same time, it should be short enough to not compromise the battery health. The threshold has been determined through internal experiments from which it is possible to be sure that the conditions are respected. For the case under scrutiny, it is experimentally found that the threshold showing a good trade-off between the requirements is 200km.

For some ML and DL algorithms, working with data having differences in the order of magnitudes, might limit the learning process. This limitation is mainly given by the optimisation algorithm used in training. If the input features have a broad range of values, inconsistent gradients may arise and thus large-scale features dominate the gradient calculations, with the result that these high-value quantities have a larger influence in the training and weight definition compared to the smaller quantities. This translates into inefficient learning leading to a compromised model accuracy in prediction. To overcome this limitation, ease and speed up the convergence of the training process, the scaling technique is applied. This ensures that the order of magnitude of the quantities involved is comparable. Feature scaling is a method used to normalise the range of independent variables and labels. Other than the advantages of speeding up the convergence of the gradient descent, feature scaling is important also where regularisation is used. Different scaling methodologies exist, and the choice of the one to be adopted strongly depends on the particular application. The main normalisation techniques are:

- min-max scaling;
- mean normalisation;
- Z-score normalisation.

Min-max scaling consists of rescaling raw data in a desired range of values, the formula is given:

$$\begin{aligned} \boldsymbol{X}_s &= \frac{\boldsymbol{X} - \min(\boldsymbol{X})}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})} \\ \boldsymbol{y}_s &= \frac{\boldsymbol{y} - \min(\boldsymbol{y})}{\max(\boldsymbol{y}) - \min(\boldsymbol{y})}, \end{aligned} \tag{4.5}$$

where $\boldsymbol{X}_s$ is the scaled features array, $\boldsymbol{X}$ is the vector of the non-scaled features, $\max(\boldsymbol{X})$, $\min(\boldsymbol{X})$ are the arrays containing the maximum and the minimum value of

each feature, $\boldsymbol{y}_s$ is the scaled array of the targets and $\boldsymbol{y}$ is the vector of the non-scaled labels.
Mean normalisation is a form of normalisation obtained by subtracting from the array its average and then dividing by the distance between the maximum and the minimum value. The general formula is:

$$\begin{aligned} \boldsymbol{X}_s &= \frac{\boldsymbol{X} - \bar{\boldsymbol{X}}}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})} \\ \boldsymbol{y}_s &= \frac{\boldsymbol{y} - \bar{\boldsymbol{y}}}{\max(\boldsymbol{y}) - \min(\boldsymbol{y})}, \end{aligned} \tag{4.6}$$

where $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{y}}$ are the average of the features and labels array respectively.
Z-score normalisation crates a dataset where each feature have zero mean and unit variance, its general formula is given by:

$$\begin{aligned} \boldsymbol{X}_s &= \frac{\boldsymbol{X} - \bar{\boldsymbol{X}}}{\sigma} \\ \boldsymbol{y}_s &= \frac{\boldsymbol{y} - \bar{\boldsymbol{y}}}{\sigma}, \end{aligned} \tag{4.7}$$

where σ is used here to indicate the standard deviation.

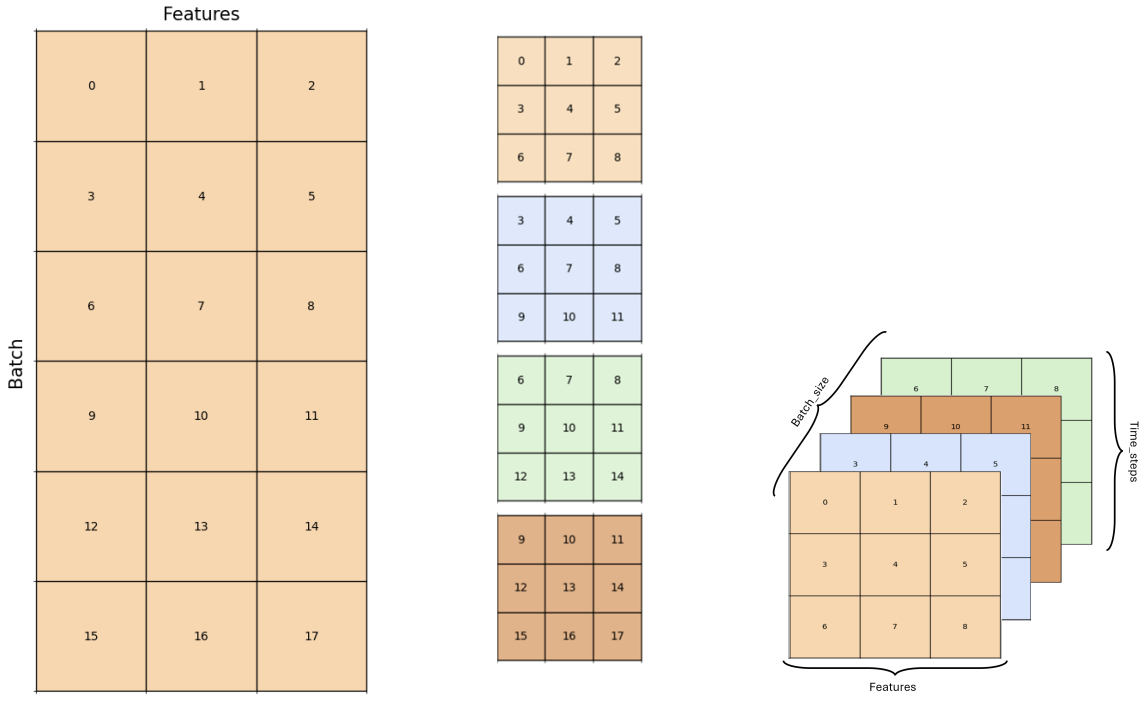
In the context of the work presented here, it is used the min-max scaling. In particular, for each of the features composing the dataset an absolute maximum and an absolute minimum value are defined and then are used in the Eq. 4.5 to implement the scaling. The adoption of absolute values allows to removal of the dependence of the scaled dataset from the original one, making the scaling general and applicable to datasets with different maximum and minimum values. Min-max scaling is the simplest rescaling method, here all the features are scaled in the interval $[0, 1]$.

A common procedure to train ML models is to define three different sub-datasets extracted from the original one, they are referred to as training, validation and test datasets and any has a specific purpose. The training dataset is used to fit the model parameters, it is passed in batches during the training phase and, based on that, the loss function is calculated and consequently the weights are updated. A validation dataset is a set of data that is used at the end of a weight update process to tune the model's hyperparameters. The test dataset is used to evaluate the results of the final version of the model. Given the time series nature of the data, the sub-datasets have to be extracted by respecting the sequential characteristic of the data. There is not a general rule to realise the split in the three different datasets, it is strongly dependent upon the dimension of the original dataset. A good choice is to select the largest part of the dataset for the training since these are the data over which the model is fitted, and splitting the remaining part in the other two. Here the splitting of the original dataset is made by extracting the first 90% for training, then the following 5% for validation and the remaining 5% for the test dataset.

The work presented here is realised by using the ML framework TensorFlow (TF). In particular, to implement the ROM of the cell, the available LSTM layer is employed. The implementation available in TF, is based on the work published by Hochreiter [69]. The LSTM, is a model developed for sequence modelling, it is part of the gated RNNs

and, as stated by Goodfellow [103], it is the most effective networks group for sequence modelling.

Given the nature of the problem, the dataset has to be arranged in a way that, when fed in the LSTM, the algorithm can recognise the presence of a sequence. A proper data organisation to feed the LSTM can be done by using a moving window sliding along the dataset and stacking each window along the third dimension to create the tensor in input to the model. The new dataset can be seen as a stacked collection of windows, each representing a sequence used to make predictions. Different ML frameworks organise the tensors in different ways. To realise the model in TF, the final shape of the tensor has to be 3D with dimensions ($batch\_size$, $time\_steps$, $features$). The first dimension indicates the number of samples to be shown to the network before a weight update is performed, the second dimension indicates the width of the window used to extract sequences from the original data and the third dimension indicates the number of features. The steps to properly create the tensor are represented in Fig.4.1. Starting from a structured dataset as the one represented in Fig. 4.1a, where the rows are populated with time instants of the signal and columns with features, it is possible to define a window of a fixed size sliding over the rows and including all the columns. For example, by defining a window wide 3 time steps and moving on the dataset represented in Fig. 4.1a, results in 4 windows including three-time steps, as represented in Fig. 4.1b. To obtain the final shape of the tensor, the resulting windows have to be stacked, as reported in Fig. 4.1c, whose final shape is $(4, 3, 3)$.



(a) (b) (c)

Figure 4.1: Representation of the different phases of dataset manipulation to properly arrange data and create the tensor. a) Representation of the dataset before data preprocessing b) Representation of the windowed dataset, each colour representing a different window c) Representation of the tensor to be fed to the LSTM.

For the particular case reported in this work, each window has been set to a fixed width of 3 s. Depending upon the frequency of the signals, the *time_step* dimension of the tensor changes. Here, it results in a number of samples for each time window of ℓ, the final shape of the feature and label tensors used in the development of the model is

$$\begin{aligned} \boldsymbol{X}_s &\in \mathbb{R}^{m \times \ell \times 3} \\ \boldsymbol{y}_s &\in \mathbb{R}^{m}, \end{aligned} \tag{4.8}$$

where m is the number of samples after the windowing process, ℓ is the number of windows and 3 is the number of features.

The dataset used to realise the work reported here is composed as

$$\begin{aligned} \boldsymbol{X} &= [I,\, SoC,\, T], \\ \boldsymbol{y} &= [V] \end{aligned}$$

where $\boldsymbol{X}$ is the feature dataset and $\boldsymbol{y}$ is the target dataset. Here, $\boldsymbol{y}$ represents the voltage of an average cell. A portion representing 10s of the dataset used for training is shown in Fig. 4.2.

4.4 LSTM architecture

Once the dataset is defined, it is necessary to choose the architecture of the model. The main characteristics to be chosen are the number of layers, the number of units for each layer and the cost function used to perform the training. Then, it is necessary to define the other hyperparameters to optimise the training procedure. As for the choice of the activation function, there is no rule to properly set the hyperparameters, it is an iterative process based on a trial-and-error approach which can lead to a long process before finding out the optimal architecture for the particular problem. As described in Par. 3.2.1, sequences can be modelled with different RNN architectures:

1. RNN to map an input sequence to an output sequence of the same length with connections between the hidden layer, see Fig. 3.19b;

2. RNN to map an input sequence to an output sequence of the same length with a connection between the output and the hidden layer, see Fig. 3.19c ;

3. RNN to map an input sequence to a single value in output, see Fig. 3.19d.

The problem faced here is a multistep forecasting problem and thus the most suitable variant is the number 1. Despite the other two architectures still valid alternatives, the chosen one represents the best compromise in terms of computational effort and prediction accuracy.

The ROM of the cell, described here, is developed through a 3 layers deep LSTM whose goal is to estimate the cell voltage using I, SoC and T as input variables. The

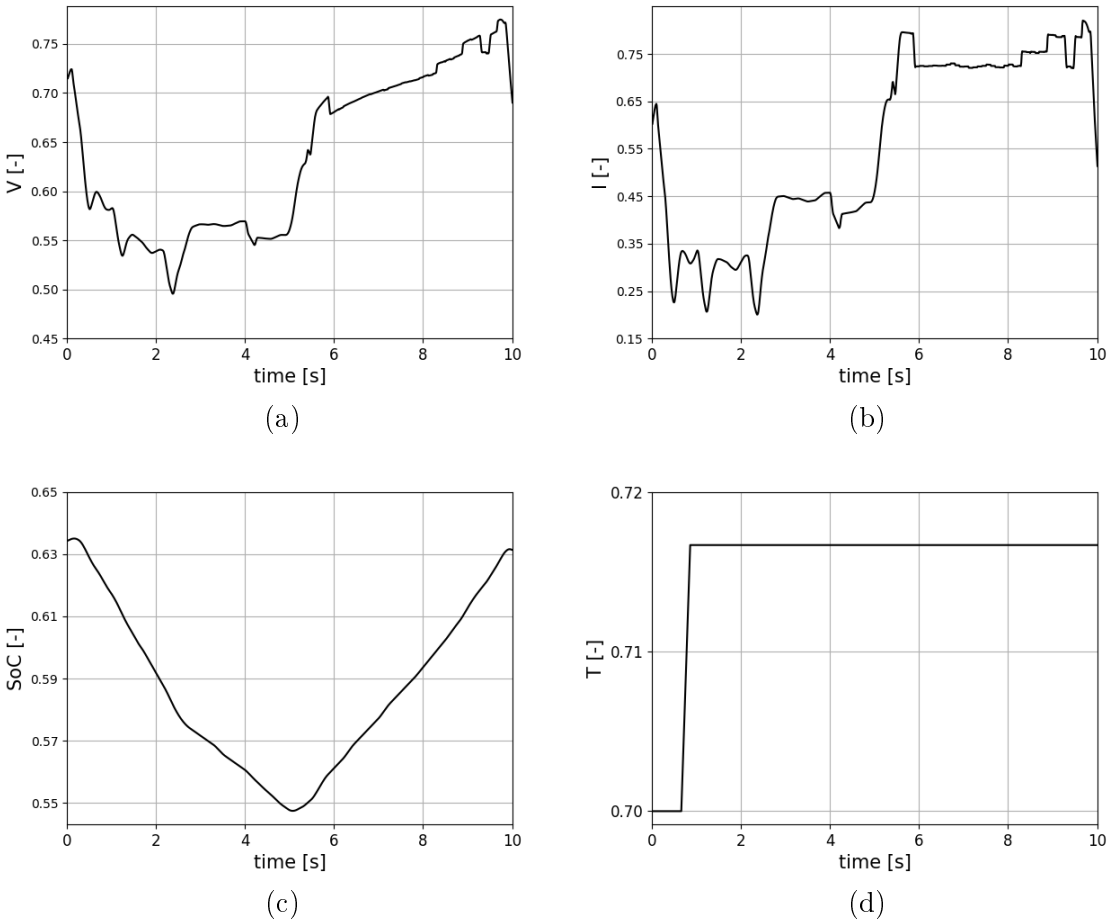


Figure 4.2: Portion of 10 s representing the scaled cycle used to implement the cell model. a) Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time.

architecture of the LSTM is:

Input layer
LSTM(16, activation = 'tanh', return_sequences = True)
LSTM(16, activation = 'tanh', return_sequences = True)
LSTM(16, activation = 'tanh', return_sequences = True)
Dense(1, activation = 'linear')

The model is trained using the Adam optimiser (see Section 3.2.1), whose parameters are reported in Table 4.1, with a *batch_size* of 128 and using MSE as a metric to evaluate model training. Callback functions are used while training proceeds to save checkpoints by monitoring the *val_loss* of the model at the end of each epoch. The training is carried out for 1500 epochs.

Parameter	Value
Learning_rate	0.001
β_1	0.900
β_2	0.999
ϵ	$1e^{-7}$

Table 4.1: Parameters defining the Adam optimiser

4.5 Battery health monitoring

This Section reports the results obtained for the health monitoring of the battery. The health estimation is assessed by comparing the VE obtained from a data-driven ROM and the VE get for three on-car aged batteries. In particular, this section is organised in two parts. The first part reports the performance of the data-driven ROM in predicting the cell voltage over the training dataset and three different test datasets. One of them is a general driving working condition and the other two are CC discharging cycles. The second part reports the results obtained with the proposed method. A battery can be seen as a series/parallel combination of cells. In the context of this work, the battery is considered to be made by n_c identical cells behaving as an average cell. The purpose of the model is to predict the voltage of the cell given as input the variables defining a known working condition. These are identified in current, SoC and temperature.

4.5.1 ROM performance

The problem of implementing a ROM model of a cell, can be seen as a regression of a time series, with data ordered chronologically. Given the nature of the data, RNN is the most suitable class of algorithm to properly represent the function driving the phenomenon. In particular, here it is used a LSTM, since it is part of the gated RNN set, the most suitable for sequence modelling problems, and thus, to represent time series and sequential data. The prediction accuracy of the ROM is assessed through the RMSE (see Sec. 3.2.1) evaluated over the training dataset, a test dataset populated with data extracted from real driving working profile and two different CC discharging cycles, with dataset populated during laboratory tests.

The RMSE resulting from the training, for the test dataset and two different CC discharging cycles, are reported in Table 4.2. To give an intuition about the model's

Dataset	RMSE [mV]
Train	1.18
Test	2.17
CC Test 1	4.91
CC Test 2	5.47

Table 4.2: Model performance evaluation through RMSE calculated over the training, the test datasets and the two CC discharging cycles.

prediction ability, a graphical comparison between the ground truth and the outcome of the model is shown in Fig. 4.3 and Fig. 4.4, where a portion of the training and

test dataset respectively are reported, and in Fig. 4.5 and Fig. 4.6 for the comparison against the two CC discharging tests.

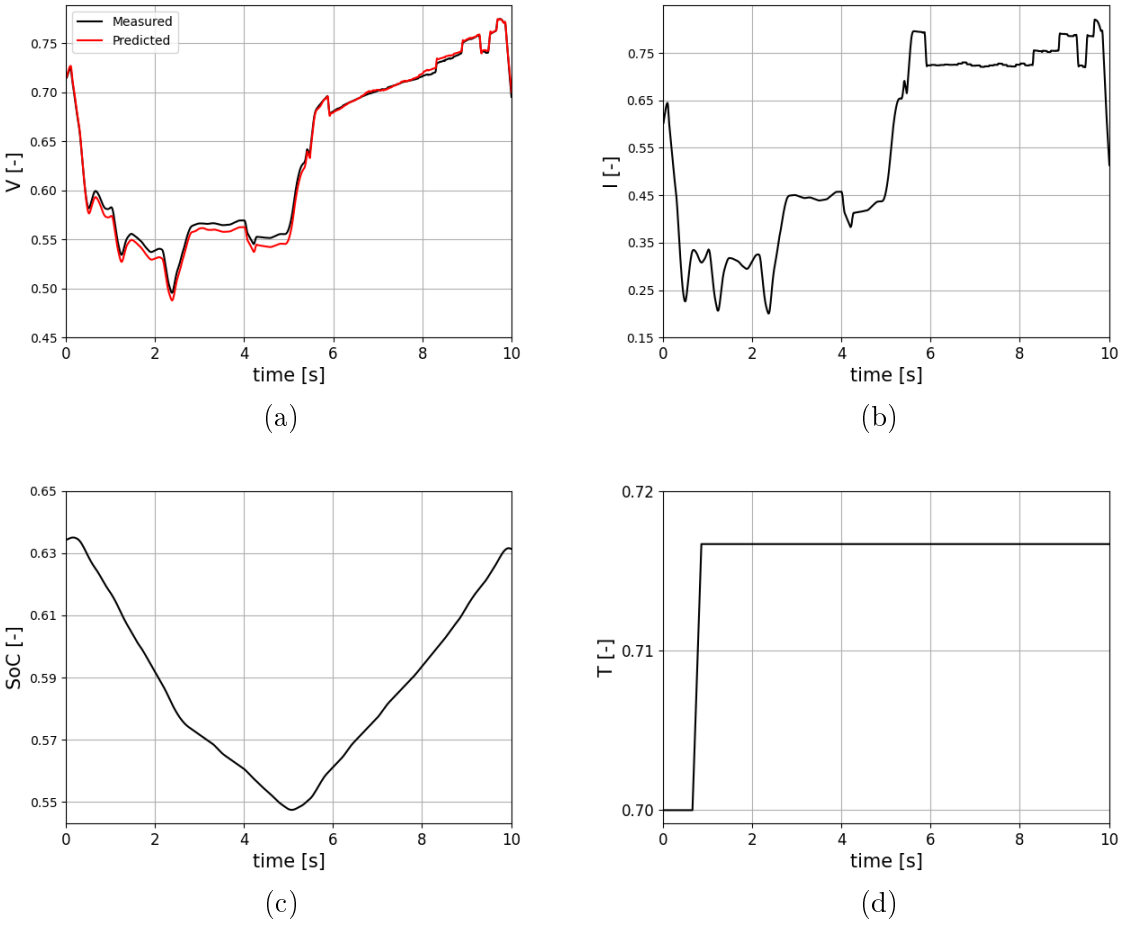


Figure 4.3: Portion of 10 s representing the comparison between the ground truth and the model outcome over the training dataset. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time.

Given the results presented in this Section and reported in Table 4.2, it is possible to observe that the prediction accuracy obtained by the data-driven ROM is high. Despite there is not an objective method to determine if the model is accurate enough, in the context of this work the errors shown can be considered small enough to define the model accurately. This consideration derives from the fact that the shown error is in the same range as the error shown by the measurement instruments used to measure the features. Given this, it is then possible to conclude that an accurate ROM of a cell is implemented. Moreover, it can profitably be used to represent the voltage behaviour of a brand-new battery experiencing given working conditions, defined through current, SoC and temperature.

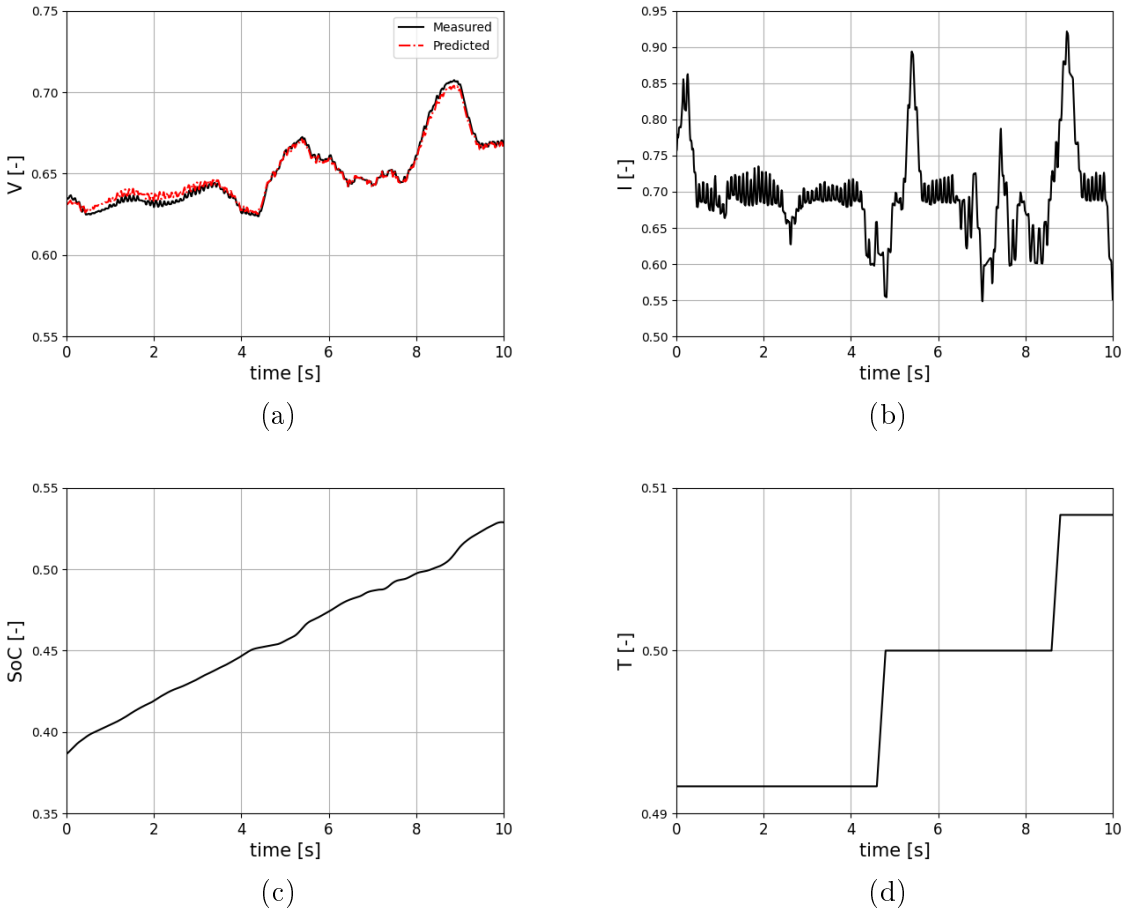


Figure 4.4: Portion of 10 s representing the comparison between the ground truth and the model outcome over the test dataset. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time.

4.5.2 Health monitoring

In this section, the results obtained with the proposed method for battery health monitoring are presented. The health of a battery is assessed via the definition of a new parameter referred to as $\Delta\eta$. It is defined as the difference between the VE (see Eq. 4.2) obtained while predicting voltage with the model and the VE get by the measured voltage of a real battery. Recalling the nomenclature used for the voltage in Eq. 4.2, the mathematical formulation of the health index introduced, is given by:

$$\Delta\eta = \eta_{U_m} - \eta_{U_b}, \tag{4.9}$$

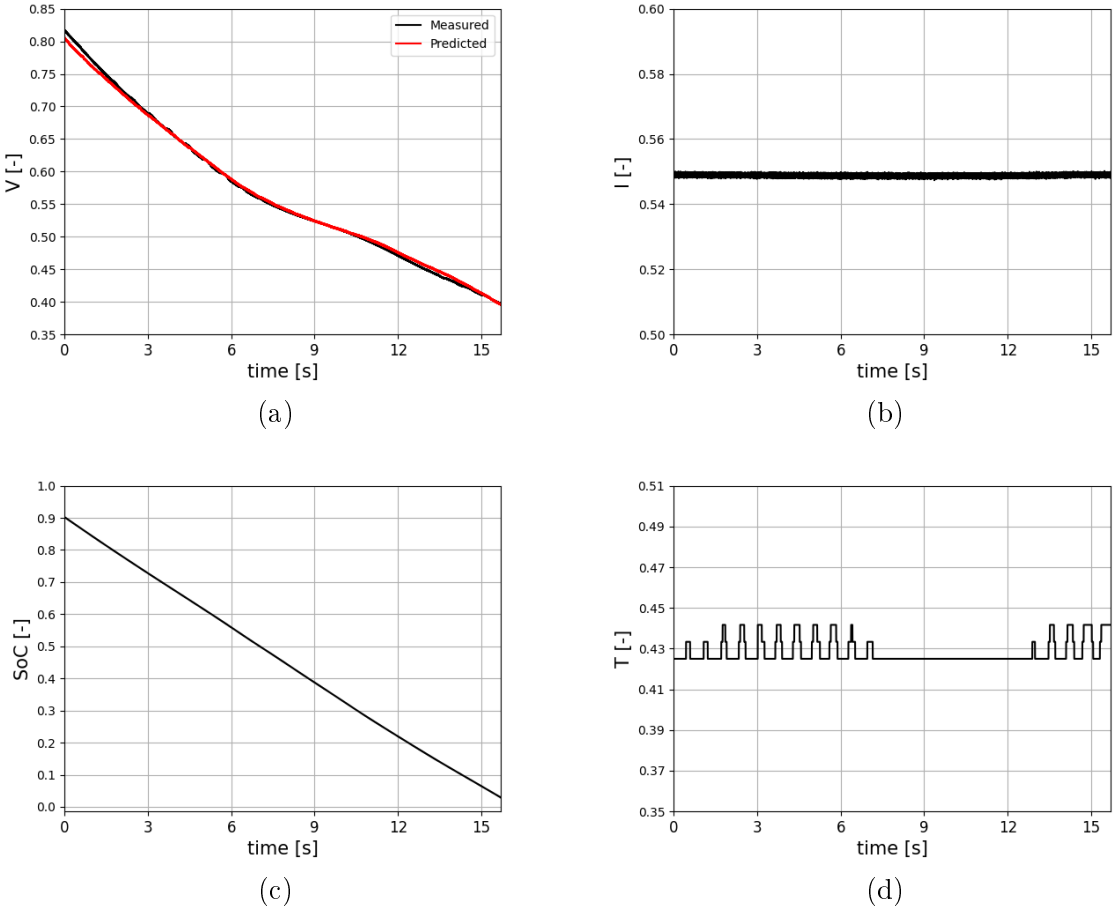


Figure 4.5: Comparison between the ground truth and the model outcome on a CC discharging cycle. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time.

where:

$$\eta_{U_m} = \frac{\int_{\Delta t} IU_m \, \mathrm{d}t \begin{cases} I &= |I| \text{ if } I < 0 \\ I &= 0 \text{ if } I \geq 0 \end{cases}}{\int_{\Delta t} IU_m \, \mathrm{d}t \begin{cases} I &= |I| \text{ if } I \geq 0 \\ I &= 0 \text{ if } I < 0 \end{cases}} * CF, \tag{4.10}$$

$$\eta_{U_a} = \frac{\int_{\Delta t} IU_a \, \mathrm{d}t \begin{cases} I &= |I| \text{ if } I < 0 \\ I &= 0 \text{ if } I \geq 0 \end{cases}}{\int_{\Delta t} IU_a \, \mathrm{d}t \begin{cases} I &= |I| \text{ if } I \geq 0 \\ I &= 0 \text{ if } I < 0 \end{cases}} * CF, \tag{4.11}$$

where m and a subscripts indicate the model and actual battery respectively. To assess the health of the battery, the $\Delta\eta$ indicator is monitored at regular intervals during the whole life of the battery. In particular, after the battery has experienced a given mileage, a proper cycle is carried out and the quantities needed to calculate the $\Delta\eta$ are

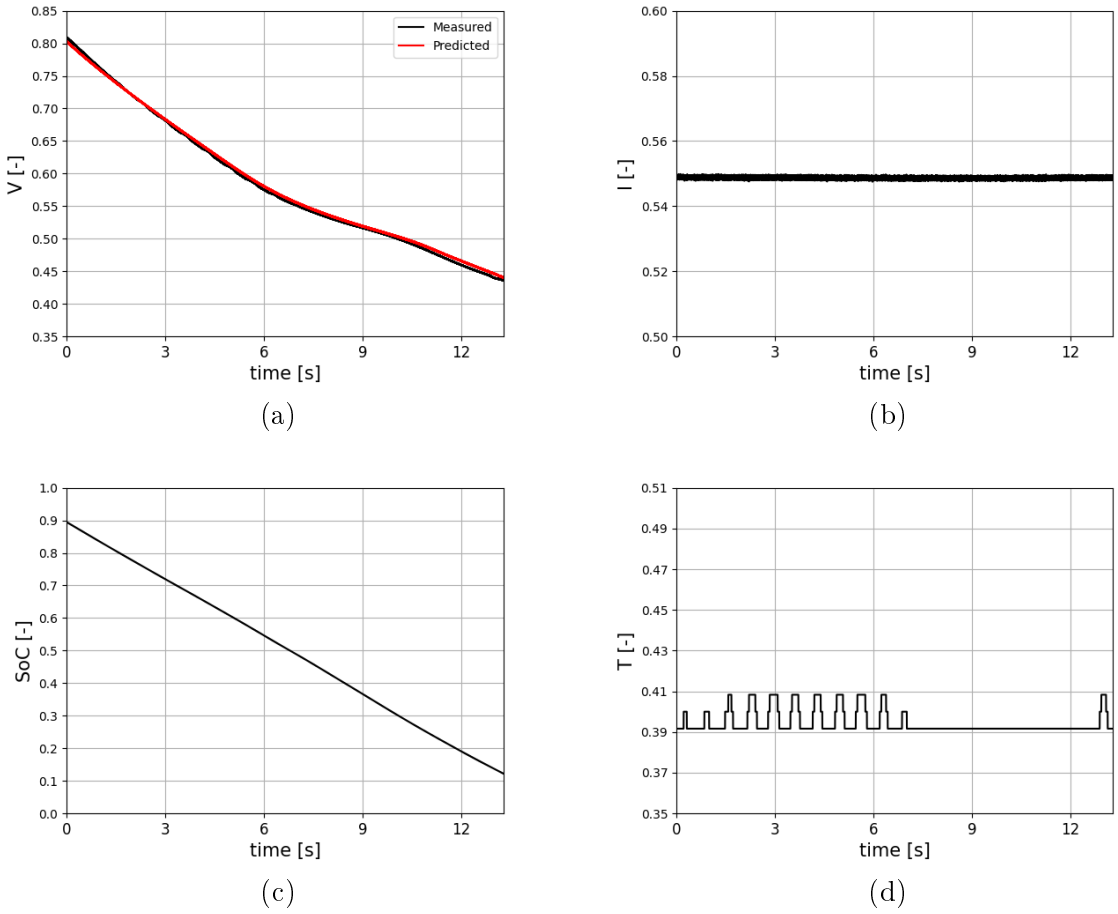


Figure 4.6: Comparison between the ground truth and the model outcome on a CC discharging cycle. a) Comparison of Voltage vs time, b) Current vs time, c) SoC vs time, d) Temperature vs time.

measured. Intuitively, such an approach allows to compare the actual voltaic efficiency of the battery which is experiencing ageing, and the voltaic efficiency of a brand-new battery (represented by the model), both experiencing the same cycle, and thus the two can be directly compared.

The effectiveness of the proposed method is shown by reporting the results obtained while monitoring three batteries. For each battery, the parameter $\Delta\eta$ is evaluated 11 times at regular intervals during its lifespan. The behaviour of the parameter $\Delta\eta$ in the following comparisons is shown in Fig. 4.7. Figures 4.7a, 4.7b and 4.7c, show the results obtained by using the proposed methodology to estimate the health of the battery. There, it is possible to observe three different lines. The blue one represents the VE of the battery which is mounted on the car, and for which the ageing increases for any subsequent evaluation cycle n_{eval}. The orange one represents the VE that a brand-new battery should have when working in the same conditions as the on-car battery, it is evaluated through the ROM. The green line represents the behaviour of the health index $\Delta\eta$ (see Eq. 4.9) used to assess the ageing of the battery. From the figure, it is possible to spot that the indicator $\Delta\eta$, increases while the ageing of the actual battery

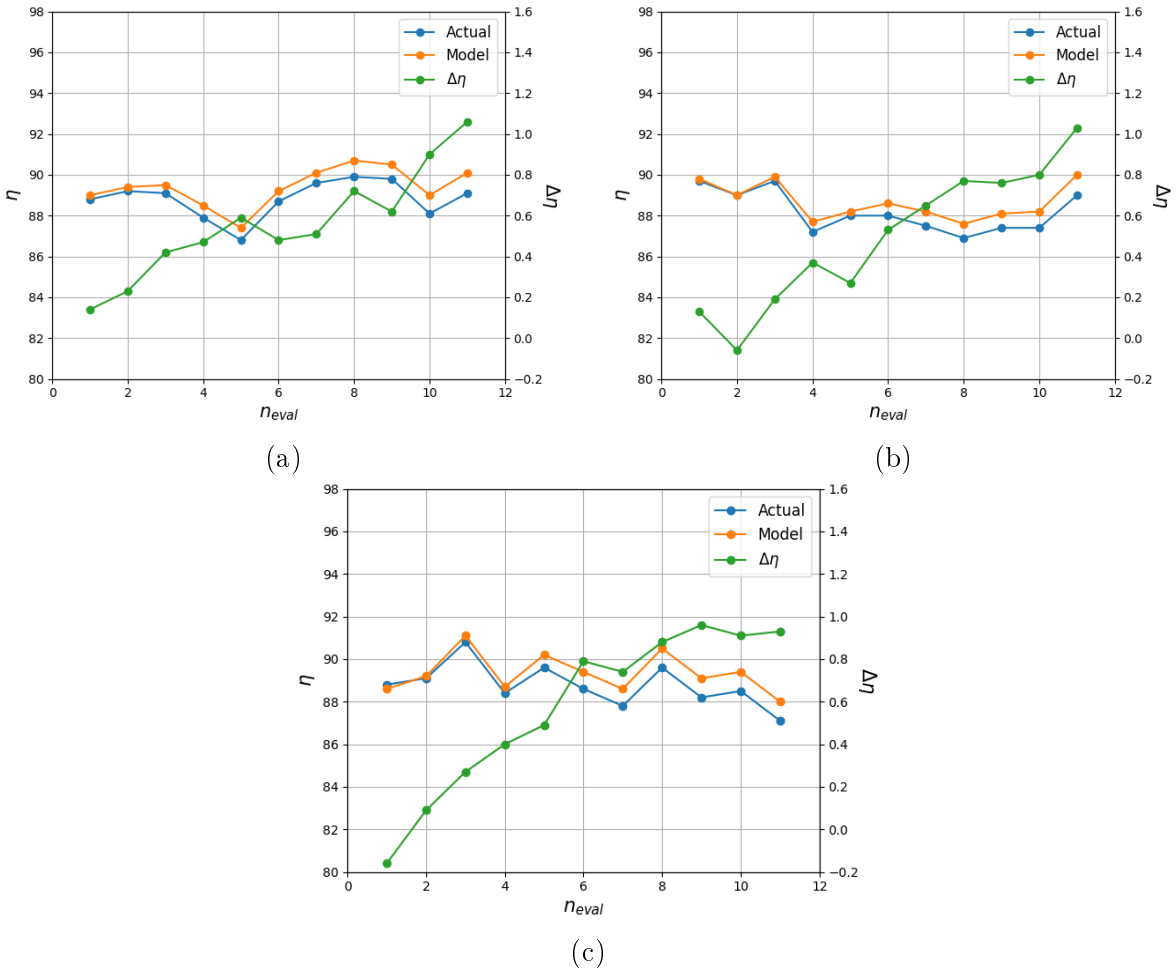


Figure 4.7: VE of the actual battery and VE calculated by predicting the voltage with the model and the difference between them. a) Results for battery number 1, b) Results for battery number 2, c) Results for battery number 3.

increases. This indicates that the VE obtained from the batteries mounted on the car is reducing, with respect to the value that should have the battery if it were new, while the ageing goes on. It is to point out, the fact that $\Delta\eta$ is not showing a monotonic increase for subsequent comparisons. It is observable an alternating behaviour. The reduction in the health index is observed when the comparison is carried out for a battery that undergoes a long rest time after a highly stressful cycle. By comparing the Fig. 4.7a, 4.7b and 4.7c, it is also possible to observe that different batteries age differently, and at the end of life (EOL), different values of the $\Delta\eta$ are shown. That means that batteries are nominally identical, experiencing different working conditions, and different life statuses are observable at the same mileage.

The results presented in this section suggest that the health monitoring through VE proposed, can effectively be used to estimate the health of a battery. Once the ML model is implemented, which is the most challenging and time-consuming part, the health estimation is straightforward. The methodology proposed allows a rapid and effective estimation of the health of a battery even if the particular conditions do not allow to use of a charging line, and thus where there is no possibility to access

CC charging or discharging curves. Moreover, given the simplicity of the methodology presented, it is suitable to be used on BMS.

Chapter 5

Data-driven battery degradation modelling

This chapter focuses on the work aimed at estimating the capacity of a cell starting from the signal of voltage collected after the application of a current spike. The behaviour shown by the voltage signal in this scenario is known as voltage relaxation, that is the voltage shown by a cell immediately after the current is removed. This Chapter, reports all the steps followed to implement the work and examines each of them in deep. In particular, here is reported all the needed information about the data collection and the population of the dataset, about the methodologies employed to properly manipulate the data, then it focuses on the definition of the models in terms of structure and hyperparameters before showing the results obtained with the application of the proposed methodology. The work reported in this Chapter is developed by using Python and the open-source machine learning library Scikit-learn.

5.1 Introduction

As introduced in Chapter 1, the battery is a critical device for the proper working of the PU. In particular, battery is a sensible component concerning the performance, and changing the outcome of a race. Given the tight restrictions on the number of batteries which can be used during a season without incurring penalties imposed by the FIA, it is important to estimate as accurately as possible the health of the battery. The estimation is important because it gives an insight into the performance which can be extracted from the battery and, moreover, it allows to prepare for the coming races at best. The work presented in this chapter arises from the necessity to estimate cell capacity with a simple procedure in a context where it is not directly measurable through the CC discharging test. The objective of the work reported here is to provide a methodology to estimate SOH from VR in contexts where the battery cannot be plugged into a charging line, as happens for the batteries used in the context of F1.

The work consists of extracting VR after the application of a charging current spike lasting 5 s. When the current is removed, data are collected for the following 10 s for each cell composing the battery. The VR signal is then manipulated with the application of statistical functions to extract four indicators: Maximum (max), Minimum (min), Mean (mean) and Variance (var). They are a statistical representation

of the curve, and it is shown to be correlated with the actual capacity of the respective cell. The statistical functions are used to train three different ML models, identified in LR, GBDT and RF. Their purpose is to estimate cell capacity from the statistical functions used as input. Here, are used three different models trained with different algorithms (See Paragraphs3.1.3, 3.1.4 and 3.1.4) to find out the best one in terms of RMSE. Since high current spikes might be dangerous for the battery, speeding up its degradation, three spikes of different intensities are applied and for each spike, the tree ML models are implemented and compared with the aim of identifying the smallest current spike that can be applied to generate a voltage response containing enough information about the actual capacity to allow a proper implementation of a predictive model.

5.2 Dataset

The first step in developing the presented work is that of collecting and properly processing the data to be used to train and test the models. Same as for the work described in Chapter 4, the data used to develop the work presented in the following Paragraphs are kindly furnished by Ferrari S.p.A, and thus they are protected by a NDA. To respect the agreement, the original value on any of the quantities used in the next Paragraphs is scaled by using an internal reference value. In particular, the values of the current, the voltage, the temperature and the capacity become:

$$I_s = \frac{I_{te}}{I_r} \qquad V_s = \frac{V_{te}}{V_r} \qquad T_s = \frac{T_{te}}{T_r} \qquad C_s = \frac{C_{te}}{C_r} \tag{5.1}$$

where I, V, T and C stand for current, voltage, temperature and capacity respectively, the subscript s is used to indicate the scaled quantity, te indicates the value collected from the test and r indicates the reference value of the particular quantity. Given the particular working conditions of the cells during races, it is not possible to define a complete charge/discharge cycle of the cells, and thus in this work its life is measured in travelled distance and the measurement unit is taken to be km.

The data needed to develop the work, are collected through on-purpose tests in a controlled environment both to replicate the on-cars conditions and to ensure the repeatability of the test. The tests are organised in five different phases, which can be summarised as:

1. Capacity measurement through the coulomb counting method during CC discharging test for the new battery;
2. Application of the current spikes at $SoC = 60\%$ to extract the VR;
3. Ageing procedure consisting of a given working condition which allows to reach a travelled distance of 1000 km;
4. Measurement of the capacity through the coulomb counting method through CC discharging test for the aged battery;
5. Application of the current spike to extract VR at $SoC = 60\%$;

6. Loop over the points 3 and 5 until the capacity reaches a threshold value below which it is considered damaged and cannot be used any more.

During the tests, a high-precision watt meter is used to measure the current and voltage and a Negative Temperature Coefficient (NTC) is employed to measure the temperature. The tests are carried out on 10 high-power Li-Ion cells nominally identical. The CC discharging test is carried out always in the same OCV range for each cell. The test starts when the OCV of the $i-th$ cell overcome the upper threshold whose normalised value is $V_s = 1$ and overs when its OCV overcome the lower threshold of $V_s = 0.65$. The same approach controlling the variables is replicated for the tests where current spikes are applied: these tests are kept with the same starting OCV for all the cells and in a temperature-controlled chamber to ensure the repeatability of the test. All the tests are carried at a constant cell temperature of $T_s = 0.4$ Given that the current spike may damage the battery, three suitable current spikes are applied to find out which is the smallest one carrying enough information to properly describe battery degradation. Following the definition given in Eq. 5.1, the intensity of the three currents used during the test is $I_s = 1$ $I_s = \frac{2}{3}$ and $I_s = \frac{1}{3}$.

The processing of the data consists of isolating the voltage relaxation and the following 10 s, this is realised through a script which exactly locates the point where the current becomes null and considers a 10 s window starting from there, isolating the VR curve. The curve is then transformed and summarised by using four statistical indicators. The four indicators used for the purpose are maximum (max), minimum (min), average (mean), and variance (var). Summarising statistics is shown to be effective in representing the shape and position change of the voltage curve [89, 99]. In order to study the correlation between each statistical indicator and the capacity, as a first step it is evaluated the Pearson Correlation Coefficient (PCC), is a measure of the linearity of the correlation. The coefficient assumes values in the range $[-1, 1]$: if $|PCC| = 1$ the correlation is linear, if $PCC = 0$ the correlation is not linear with all the intermediate values representing different degrees of linearity. It is important to point out that in the case of $PCC = 0$ does not mean that the variables are not correlated, but it only means that if the correlation exists it is not linear. The PCC values for the statistical indicators and the capacity are shown in Figs. 5.1a, 5.1b and 5.1c. Pictures show that the correlation, between the capacity and the four statistical indicators is strong and can be considered linear. Moreover, they also show that while the variance and maximum indicators keep a high linearity correlation for all three spike values, this is not true for minimum and mean, whose correlation with the capacity decreases while reducing the spike, being weak for the lowest peak.

The distribution of the capacity against the four statistical indicators for different ageing levels can be seen in Figures 5.2, 5.3 and 5.4 for current spikes $I_s = 1$, $I_s = \frac{2}{3}$ and $I_s = \frac{1}{3}$ respectively. From Figs. 5.2, 5.3 and 5.4, it is also possible to observe that the capacity shows a negative correlation with the four statistical indicators. Moreover, it is also possible to observe that a reduction in the current spike is associated with a reduction in the correlation between the statistical indicators and the capacity. This happens because a reduction in the current spike brings a loss in the information carried by the VR.

As introduced, a common procedure for training ML models is that of defining three different sub-datasets starting from the original one. These are called training

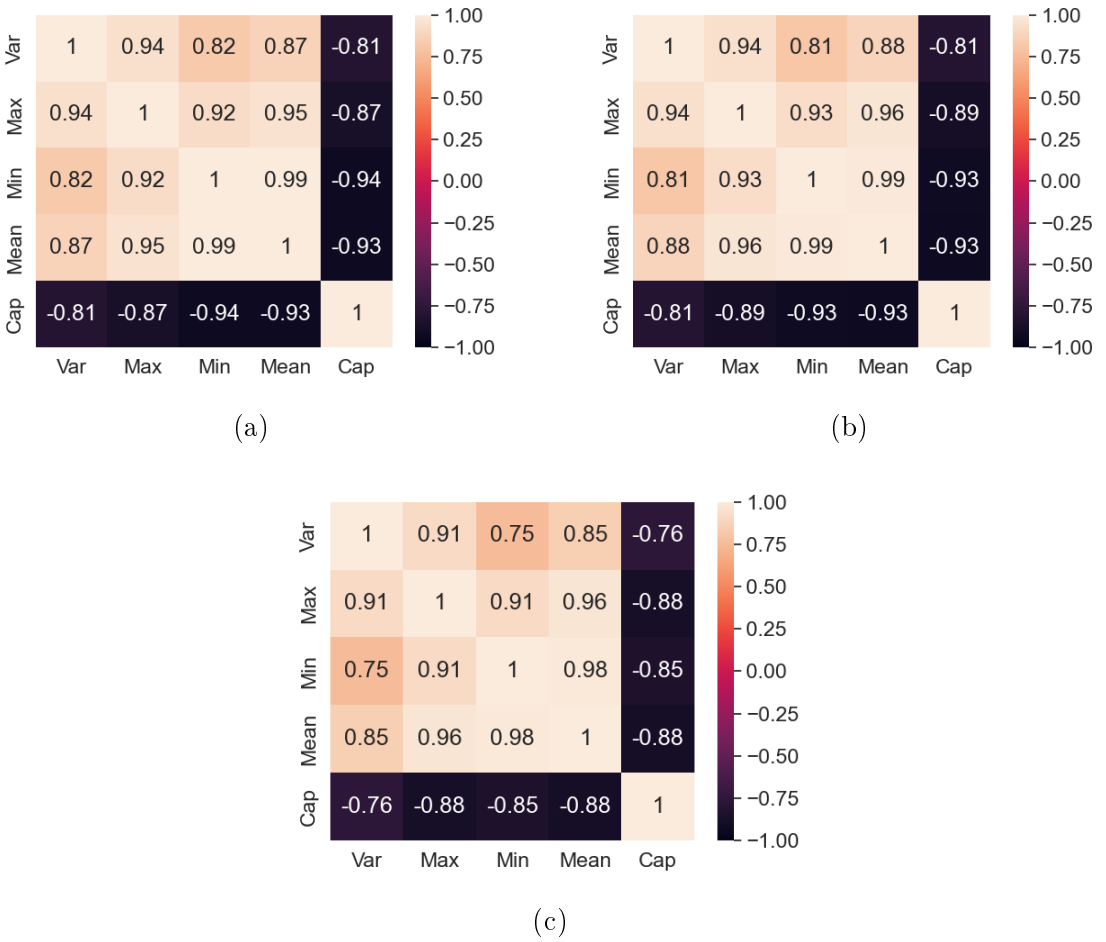


Figure 5.1: Linear correlation matrix of the features extracted from the voltage relaxation curves and the capacity. a) $I_s = 1$, b) $I_s = \frac{2}{3}$ and c) $I_s = \frac{1}{3}$.

dataset, validation dataset and test dataset. The training dataset is used to perform the training of the model, the validation dataset is a never-seen set of data useful to assess an estimation of the model skills while training goes on. The test dataset is a never-seen set of data used to evaluate the model prediction error when the training is complete, it gives an overview of the generalisation of the model and its ability to work with never-seen data. Due to the small quantity of data available, only the training and test datasets are defined. They are extracted by randomly splitting the original dataset into two sub-datasets, where the 90% of the data is used for the training dataset, and the remaining 10% is used for the test dataset. The function used to perform this operation is part of the scikit-learn python library and is called *sklearn.model_selection.train_test_split*. It allows to randomly split the data and, through the activation of a parameter called *random_state*, it is possible to have a reproducible output across multiple calls [106]. This can be obtained by setting the parameter always to the same integer value. For the work under scrutiny, it is activated and set to an integer to get the same output in different function calls.

In some datasets, may happen that the difference between the values of the features and the target is orders of magnitude. In this context, scaling data might be beneficial

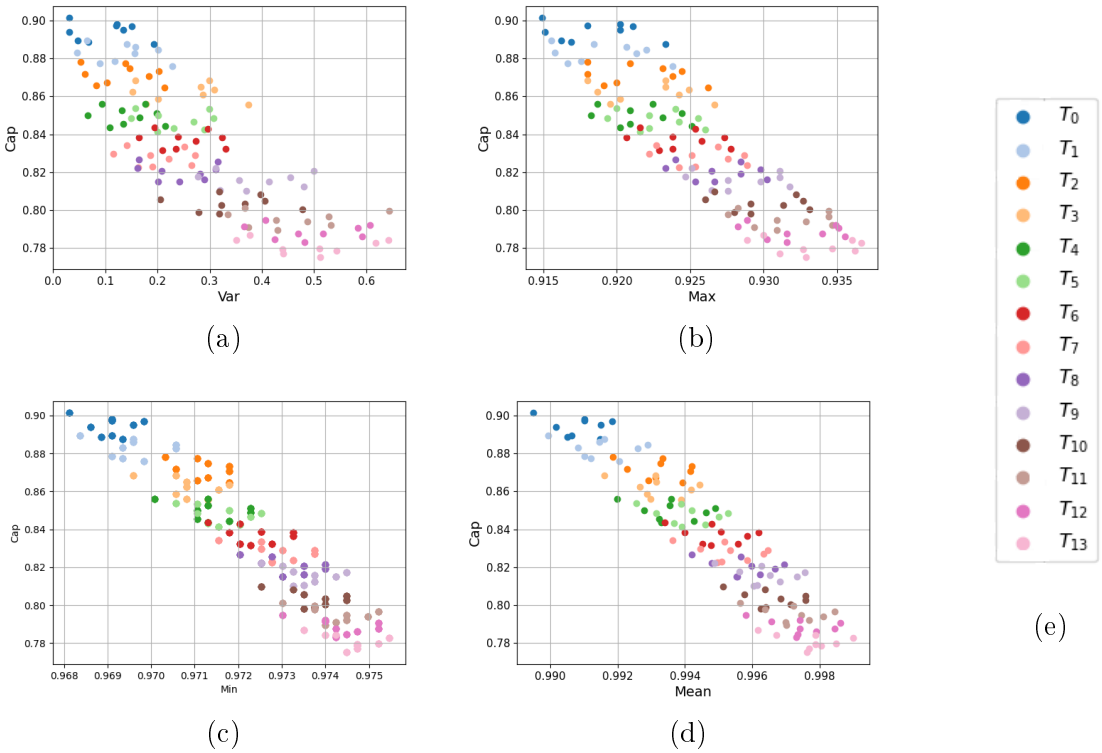
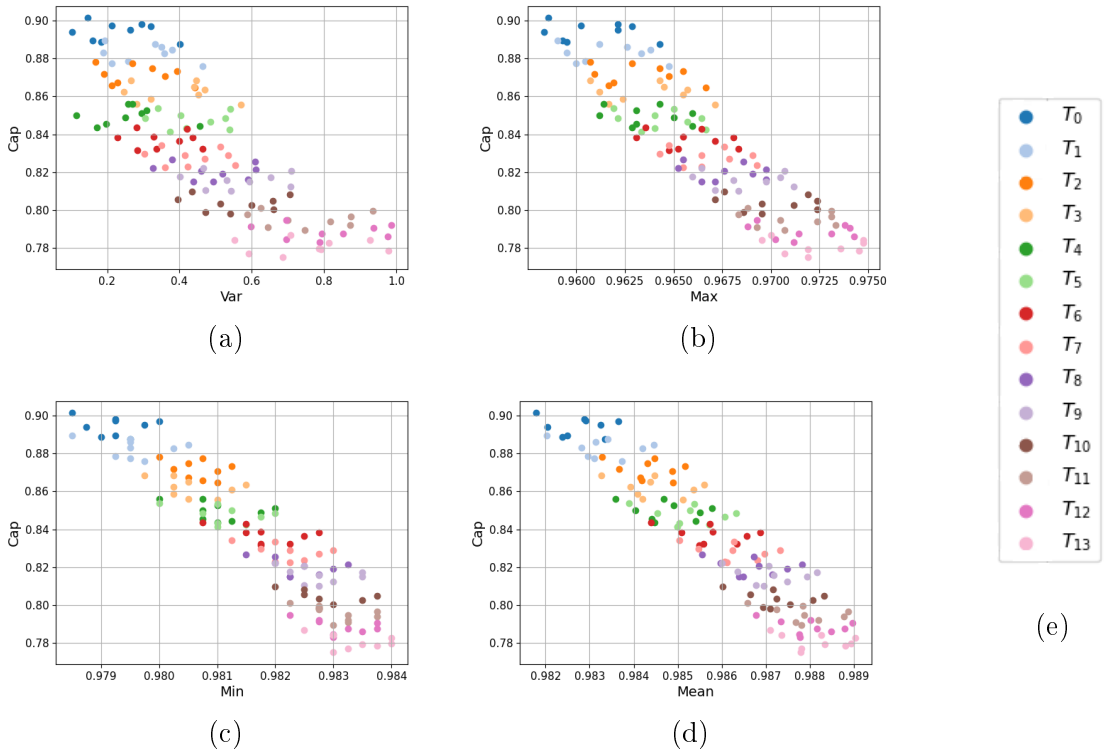


Figure 5.2: Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = 1$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test.

for the training of some ML models (see Section 4.3). For the work presented here, min-max scaling is employed. This results in a new dataset where each variable value is in the range $[0, 1]$. The scaling is implemented by defining an absolute maximum and minimum value for each variable involved and then by applying the formula:

$$\begin{aligned} \boldsymbol{X}_s &= \frac{\boldsymbol{X} - \min(\boldsymbol{X})}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})} \\ \boldsymbol{y}_s &= \frac{\boldsymbol{y} - \min(\boldsymbol{y})}{\max(\boldsymbol{y}) - \min(\boldsymbol{y})}, \end{aligned} \tag{5.2}$$

where $\boldsymbol{X}_s$ is the scaled features array, $\max(\boldsymbol{X})$ and $\min(\boldsymbol{X})$ are the arrays of the maximum and the minimum values of each feature respectively. The Same procedure is applied to the label vector where $\boldsymbol{y}_s$ is the scaled label array, $\min(\boldsymbol{y})$ and $\min(\boldsymbol{y})$ are the arrays of the maximum and the minimum value of each label. For the algorithms used in this contribution, scaling improves the learning ability of LR, and it is not beneficial for GDBT and Random Forest Regression, thus the data are scaled only before the training of LR.



(a) (b) (c) (d) (e)

Figure 5.3: Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = \frac{2}{3}$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test.

5.3 Models definition & Training

The dataset described in the Paragraph above is used to implement three different ML models. What pushes to development of three different models, is the necessity to find out the best one in terms of prediction accuracy, and thus three completely different models with different training algorithms are trained and compared in terms of RMSE. The models implemented here are Linear Regression, Gradient Boosting Decision Tree and Random Forest Regression, for which the mathematical description is reported in Sections 3.1.3, 3.1.4 and 3.1.4 respectively. The objective of each model is to estimate the capacity of each cell composing the battery given in input the dataset (see Par. 5.2) populated with statistical functions describing the VR curve. The VR curve is summarised by using the statistical indicators maximum, minimum, average and variance. All the models used in this work are implemented using Python 3.8.10 as a scripting language and the library Scikit-learn 1.3.0 [106]. In this Paragraph, is reported the architecture of any model implemented in the context of this work. Moreover, detailed information about the hyperparameters and the training procedure are given.

As introduced, the models used to realise this work are implemented through the

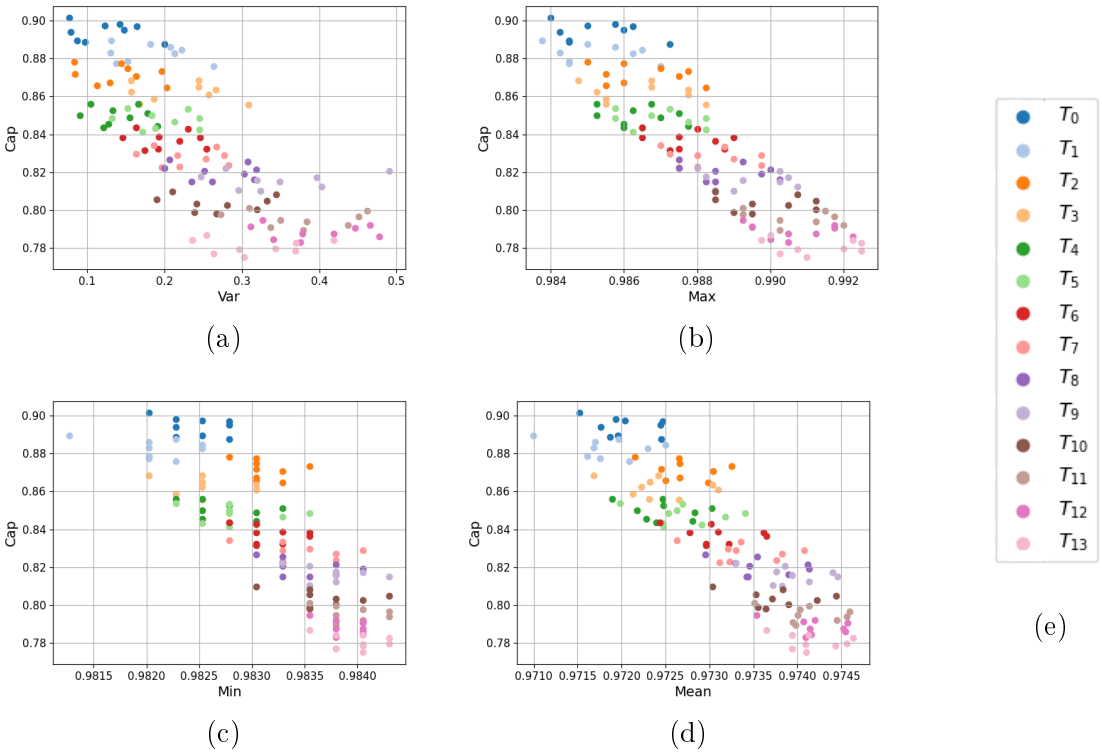


Figure 5.4: Capacity distribution for different ageing conditions as a function of the statistical indicators extracted from the curves obtained with a spike current of $I_s = \frac{1}{3}$. a) Capacity vs variance, b) capacity vs maximum, c) capacity vs minimum, d) shows the relationship between capacity and the average value of voltage and e) shows the legend, T stands for Test and the subscript stands for the progressive number of the test.

Scikit-learn library [106], in particular, they are implemented using the available class:

- sklearn.linear_model.LinearRegression;
- sklearn.ensemble.GradientBoostingRegressor;
- sklearn.ensemble.RandomForestRegressor.

The class LinearRegression fits a linear model to minimise the RSS of squares between the ground truth value and the outcomes predicted with a linear approximation of the function described by the data. Class GradientBoostingRegressor builds an additive model in a forward-wise fashion allowing for the optimisation of many differentiable functions, which for the case under scrutiny is chosen to be MSE. In each iteration, a regression tree is fit on the negative gradient of the loss function. The class RandomForestRegressor fits a number of decision tree regressors on various sub-samples of the dataset and uses averaging to improve the predictive accuracy, control over-fitting and reduce the variance of the model. The parameters used to define the LR model are reported in Table 5.1, the algorithm used for the optimisation of the parameters is Ordinary Least Squares (OLS). The hyperparameters used to define GBDT are re-

Parameter	Value
fit_ intercept	True
n_jobs	None

Table 5.1: Hyperparameters used for the implementation of the LR

ported in Table 5.2, while the hyperparameters used to define the Random Forest are shown in Table 5.3. In order to have the model as accurate as possible, it is necessary to properly set these parameters. Finding a proper setting of the parameters consists in finding out the best combination which reduces the error prediction and allows a high generalisation. This involves a highly time-consuming procedure of fixing the value of a parameter and varying all the others in a search space, training the model and comparing the results to find the best compromise. This operation can be done manually, requiring a huge amount of time given the large dimension of the search space within which it is possible to choose the parameters defining the models. Fortunately, Scikit-learn provides a Class to automatically realise this work. The name of the Class is *GridSearchCV*. Given a defined search space, the function implements a fit-and-score method to extract the best combination of the hyperparameters which minimise a given loss function for a particular model. For the case under scrutiny, the loss function is MSE. This approach is used to train both GBDT and RF. The results of the hyperparameters tuning are used to define the final model, they are reported in Table 5.2 and Table 5.3 for GBDT and RF respectively.

Parameter	Value
n_ estimators	12
max_ depth	2
learning_ rate	0.9
min_ samples_ split	12
min_ samples_ leaf	12
loss	*squared_ error*

Table 5.2: Hyperparameters used for the implementation of the GBDT

Parameter	Value
n_ estimators	24
min_ sample_ split	4
max_ depth	*None*
min_ samples_ leaf	2
max_ features	2
criterion	*squared_ error*

Table 5.3: Hyperparameters used for the implementation of the Random Forest

5.4 Results

The objective of the work presented in this Chapter is to implement a simple and forward methodology to estimate the actual cell capacity for applications where the charging process is not employed, such as for batteries employed in F1. The problem is tackled here by using the VR triggered by a current spike. The VR curve is processed to extract four statistical indicators, identified in minimum, maximum, average and variance, to summarise the curve itself. It is shown that the four statistical indicators are correlated to the actual capacity of the battery.

As introduced, current spikes might damage the battery. To reduce as much as possible that possibility, three different magnitudes for the spike are used. For each spike value, it is analysed the effect on the correlation between the four statistical indicators and the capacity. This analysis is made to highlight if a variation in the current spike affects in the possibility of estimating the cell capacity. Eventually, three different machine learning models are implemented, or each current spike, to estimate the cell capacity, and the best one in terms of RMSE over the training and test dataset is selected. In this Paragraph, the results obtained with all the models implemented are reported. The prediction accuracy of the three models is assessed via the root mean squared error (RMSE), defined in Eq 3.39. The same is used to compare the three models for each value of the current spike.

The results of the predictions carried out for both the training and the test dataset are shown in Table 5.4. From Table 5.4, it is possible to observe that all three spikes

Model	Current spike	Train RMSE*100	Test RMSE*100
Linear Regression	1	2.20	2.27
GBDT	1	1.72	1.84
Random Forest	1	1.87	1.95
Linear Regression	2/3	1.61	2.60
GBDT	2/3	1.78	1.87
Random Forest	2/3	1.84	1.99
Linear Regression	1/3	2.31	3.08
GBDT	1/3	2.42	2.71
Random Forest	1/3	2.25	2.60

Table 5.4: RMSE obtained with each model over the train and test dataset

can be used to estimate the cell capacity with good accuracy. Anyway, the results suggest that the best model is the GBDT trained with the spike at $I_s = 1$. This shows an error of 0.0172 Ah on the training dataset and of 0.0184 Ah on the test dataset. The result that the smallest error is obtained with the highest current spike, is not surprising. This derives from the fact that the highest spike carries the largest amount of information about battery health in comparison to the other two spikes. A reduction in the magnitude of the current spike translates into a VR curve with a narrower shape, which affects the statistical indicators chosen to estimate the cell capacity. The variation of the voltage curve caused by a smaller spike brings information losses which affect the final result.

The comparison between the actual value and the prediction obtained with the best model (the GBDT with the spike at $I_s = 1$) can be seen in Figures 5.5 and 5.6 for the train and test dataset respectively.

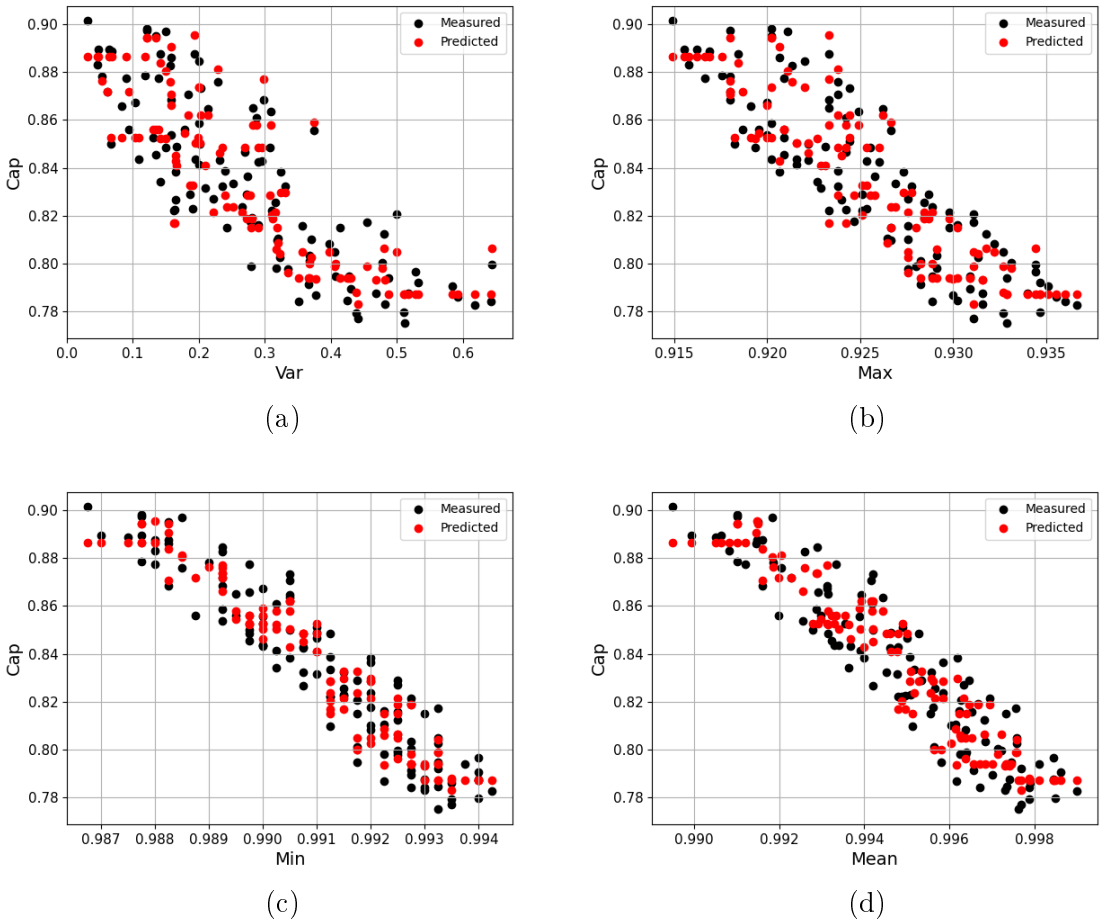


Figure 5.5: Comparison between the ground truth and the prediction obtained with the GBDT model for the spike at $I = 1$ over the training dataset, Figure (a) capacity against the variance, (b) capacity against maximum, (c) capacity against minimum, (d) capacity against the mean.

The results reported in this Chapter, show that the proposed methodology can be profitably used to estimate the actual cell capacity with high accuracy starting from a simple test. Given the simplicity of the steps to get the data, it is possible to implement this methodology in all those contexts where it is not possible to connect the vehicle to a charging line. Moreover, the models implemented to map from the voltage relaxation curve to the capacity are very cheap in terms of memory requirement and thus this methodology is suitable to be implemented on BMS.

Despite the good effectiveness of the model, the main limitation of this study is represented by the fact that the cells are tested in a controlled environment with fixed temperature and with a single value for SOC at 60%. This may affect the result when the procedure is applied with the battery in different environmental conditions, such as different temperatures or different SOC.

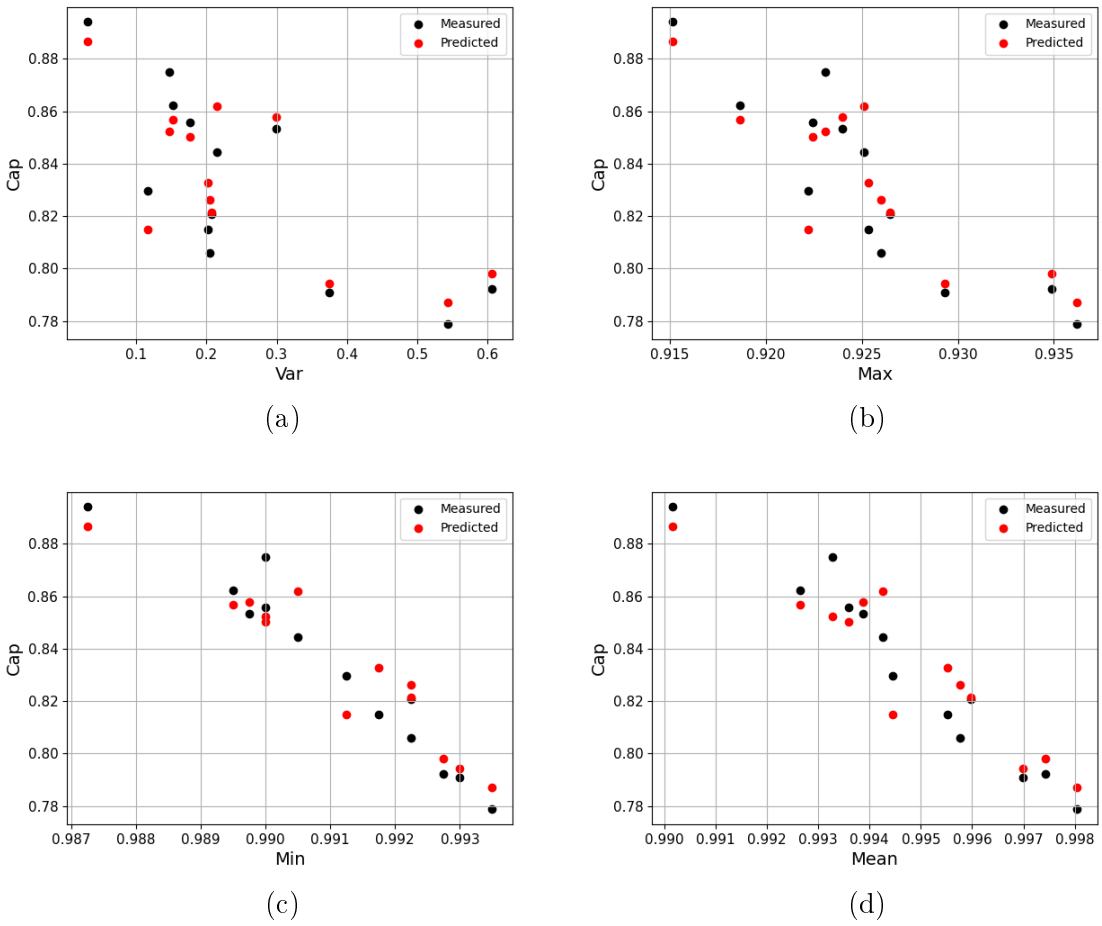


Figure 5.6: Comparison between the ground truth and the prediction obtained with the GBDT model for the spike at $I = 1$ over the test dataset, Figure (a) capacity against the variance, (b) capacity against maximum, (c) capacity against minimum, (d) capacity against the mean.

Chapter 6

Conclusions

Decarbonisation of transportation is seen as a key factor in reducing environmental pollution, and thus in the last years strict regulations have been imposed to limit the CO_2 emissions. Following these regulations, carmakers started to invest massively in the production of hybrid electric vehicles and fully electric vehicles which rely on batteries as a supplementary or principal energy storage system respectively. Following the tradition of the F1 of keeping pace with the evolution trends in automotive engineering, in 2014 also the F1 saw the introduction of an extremely complex hybrid system, named the power unit. The system is composed of an internal combustion engine, two different electric motors, a turbocharger, a battery and the respective control units. All these systems are orchestrated and together cooperate to properly provide power to the car. In all of these contexts, the Lithium battery assumes an important role, being the main energy storage system for the electrical part of the powertrain. This success is mainly driven by its characteristic of high power and high energy density, long lifespan and low self-discharge which make these systems the most suitable for applications where high reliability and low weight characteristics are required. Anyway, besides the desired characteristics reported, the battery is subject to deterioration which is referred to as ageing. On the one hand battery ageing lowers the capacity, which translates into a reduced amount of charges that can be stored; on the other hand increases the value of the resistance reducing the amount of power that can be extracted from the battery and its conversion efficiency. Given the importance and the actuality of the argument, in the last years, a lot of research has focused on developing methodologies to accurately estimate the health status of the battery, referred to as SOH. Many approaches are available in the literature which can be organised into three main categories: experimental, physics-based and data-driven. Anyway, some approaches make use of hybrid methodologies belonging to different groups which can be collocated at the intersection between the main categories. Despite all the efforts made, nowadays there is not yet a methodology universally applicable to all situations, and this makes the topic still an open research problem.

In this thesis, two different methodologies to estimate battery SOH are proposed. The works reported here, arise from the necessity to find a fast and easy methodology to estimate SOH which can be used in contexts, such as F1, where the battery cannot be charged by connecting to a charging line, and thus it is not possible to access full of partial charging/discharging curves.

In one of the two works, the one reported in Chapter 4, the problem is tackled by implementing a data-driven ROM of a cell. The model is developed under the hypothesis that a battery can be seen as a series/parallel arrangement of cells all behaving the same, as the modelled one. The purpose of the cell model is to estimate the battery voltage taking in input current, SoC and temperature. As stated, battery behaviour is the result of the interdependence of multiple physical phenomena, which depends also upon time. To account for these dependencies, it is necessary to use an RNN to develop the model. In the work reported, it is used an LSTM (which is the most promising algorithm to properly model cell behaviour) with a many-to-many architecture taking as input 2s of history of the current, SoC and temperature. Based on these features the model estimates the battery voltage. LSTM is chosen because the battery is a highly non-linear system where different physical phenomena occur concurrently. These phenomena not only are highly interdependent but also depend upon their past, giving a dependence upon time. The voltage estimated through the model over a given working profile characterised by a given current, SoC and temperature, is used to evaluate the voltaic efficiency that a battery would have when working in those conditions if it were new. Such value is then compared with the voltaic efficiency calculated from measured data of an aged real battery working over the same profile. The comparison is carried out to estimate the degradation imposed by the ageing of the real battery. Such a comparison can be done since the profiles used to calculate the voltaic efficiency are the same. Two different results are presented. One is to evaluate the prediction accuracy of the model, the other reports the results obtained by applying the proposed methodology to estimate the degradation of three real batteries. The prediction accuracy of the model is assessed by RMSE and by a visual comparison for a portion of a driving cycle over four different sets: training dataset, test dataset and two datasets representing a CC discharge. The results show that the model can accurately predict all the different working profiles where the battery is working. The worst RMSE is shown for a dataset representing a CC discharging test, its value is 5.47 mV. On the other hand, the results obtained by applying the proposed methodology to estimate the degradation to three real batteries working on three different cars is presented. They show that the proposed method, based on the comparison of the voltaic efficiency of an aged real battery with that of a new battery, is an effective methodology to estimate the degradation to which the battery is subject. Moreover, the results show that different batteries, subject to different ageing cycles, age differently and thus also the difference between the efficiencies changes for two batteries with the same mileage. The methodology proposed in this work can be profitably used to estimate the battery SOH. Moreover, given the approach employed, it can be used for online applications by embedding the model on a BMS since the required computational effort can be sustained by modern BMS.

The other work, described in Chapter 5, proposes a methodology to estimate the cell capacity through the voltage relaxation triggered by a current spike. To the best of the author's knowledge, this is the first time that the voltage relaxation used to estimate the capacity is triggered by current spikes. Most of the studies available in the literature make use of the VR collected after a full charge procedure with CCCV. The problem of estimating cell capacity is faced with a hybrid methodology, which can be categorised as half experimental and half data-driven. The method consists

of applying a current spike of 5 s leading to a voltage step and, once the current is removed, to a voltage relaxation. The data are collected for 10 s after the current removal and are properly processed to extract four statistical indicators, they are used as features to implement three machine learning models aimed at estimating battery capacity. The processing of the data consists of isolating the voltage relaxation and the following 10 s, this is realised through a script which exactly locates the point where the current becomes null and considers a 10 s window starting from there, isolating the VR curve. The curve is then used to extract the four statistical indicators describing the curve itself. These are identified in maximum, minimum, average and variance, which are shown to be correlated to the actual cell capacity. Since applying current spikes might danger the battery, three different spikes are used with the aim of finding the smallest one resulting in a voltage relaxation curve carrying enough information about the battery capacity. Eventually, three ML models are implemented with the goal of estimating cell capacity as a function of the four statistical indicators. These are identified in multivariate linear regression, gradient boosting decision tree and random forest regressor. The objective of the implementation of three different models is to find out which is the best one, in terms of RMSE, to approximate cell capacity and with what current value. The accuracy of each model is assessed by testing them over the training dataset and a test data set. The results show that the best model is the GBDT with the highest current spike. It shows a RMSE of 0.0172 Ah over the training dataset and 0.0184 Ah over the test dataset. On the other hand, the worst prediction accuracy is provided by the multivariate linear regression working with voltage relaxation triggered with the smallest current spike. The RMSE over the training dataset is 0.0231 Ah and the error over the test dataset is 0.0308 Ah. The methodology proposed in Chapter 5, provides a fast and easy way to accurately predict cell capacity in applications where the battery cannot be connected to a charging line. Given the simplicity of the models implemented, the presented methodology can easily be embedded in BMS since the computational effort required to run such models is minimal.

As extensively reported, the problem of determining the health of a battery is complex and there is not yet a universally accepted methodology which is suitable to be applied to all the different contexts which can appear. In the PhD research, the main focus has been of finding easy methodologies to estimate battery SOH in a special context, with different battery management strategies compared to the passenger cars. The research brings to implement two works with different characteristics, but anyone with particular strengths that make it suitable to be applied in the reference context.

Bibliography

[1] U.S. Department of Energy. *Batteries for Electric Vehicles*. 2017.

[2] 2024 Fédération Internationale de l'Automobile. *2024 Formula 1 Technical Regulation*. URL: www.fia.com/sites/default/files/fia_2024_formula_1_technical_regulations_-_issue_3_-_2023-12-06.pdf.

[3] Eduardo Redondo-Iglesias, Pascal Venet and Serge Pelissier. 'Calendar and cycling ageing combination of batteries in electric vehicles'. In: *Microelectronics Reliability* 88-90 (2018), pp. 1212–1215. ISSN: 0026-2714. DOI: https://doi.org/10.1016/j.microrel.2018.06.113.

[4] Gaizka Saldaña et al. 'Empirical calendar ageing model for electric vehicles and energy storage systems batteries'. In: *Journal of Energy Storage* 55 (2022), p. 105676. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2022.105676.

[5] Anthony Barré et al. 'A review on lithium-ion battery ageing mechanisms and estimations for automotive applications'. In: *Journal of Power Sources* 241 (2013), pp. 680–689. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2013.05.040.

[6] Prashant Shrivastava et al. 'Review on technological advancement of lithium-ion battery states estimation methods for electric vehicle applications'. In: *Journal of Energy Storage* 64 (2023), p. 107159. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2023.107159. URL: https://www.sciencedirect.com/science/article/pii/S2352152X2300556X.

[7] M. bere et al. 'Critical review of state of health estimation methods of Li-ion batteries for real applications'. In: *Renewable and Sustainable Energy Reviews* 56 (2016), pp. 572–587. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.11.042. URL: https://www.sciencedirect.com/science/article/pii/S1364032115013076.

[8] David Vincent Ragone. 'Review of Battery Systems for Electrically Powered Vehicles'. In: *SAE MOBILUS* (1968). DOI: https://doi.org/10.4271/680453.

[9] Bradley A. Johnson and Ralph E. White. 'Characterization of commercially available lithium-ion batteries'. In: *Journal of Power Sources* 70.1 (1998), pp. 48–54. ISSN: 0378-7753. DOI: https://doi.org/10.1016/S0378-7753(97)02659-1. URL: https://www.sciencedirect.com/science/article/pii/S0378775397026591.

[10] Shin-ichi Tobishima and Jun-ichi Yamaki. 'A consideration of lithium cell safety'. In: *Journal of Power Sources* 81-82 (1999), pp. 882–886. ISSN: 0378-7753. DOI: https://doi.org/10.1016/S0378-7753(98)00240-7. URL: https://www.sciencedirect.com/science/article/pii/S0378775398002407.

[11] Mario Marinaro et al. 'Bringing forward the development of battery cells for automotive applications: Perspective of R&D activities in China, Japan, the EU and the USA'. In: *Journal of Power Sources* 459 (2020), p. 228073. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2020.228073. URL: https://www.sciencedirect.com/science/article/pii/S0378775320303761.

[12] A.H. Zimmerman. 'Self-discharge losses in lithium-ion cells'. In: *IEEE Aerospace and Electronic Systems Magazine* 19.2 (2004), pp. 19–24. DOI: 10.1109/MAES.2004.1269687.

[13] J. Vetter et al. 'Ageing mechanisms in lithium-ion batteries'. In: *Journal of Power Sources* 147.1 (2005), pp. 269–281. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2005.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0378775305000832.

[14] A. de Guibert and A. Hermelin. 'Ageing of rechargeable lithium batteries'. In: *Journal of Power Sources* 14.1 (1985), pp. 57–64. ISSN: 0378-7753. DOI: https://doi.org/10.1016/0378-7753(85)88011-3. URL: https://www.sciencedirect.com/science/article/pii/0378775385880113.

[15] Rui Xiong, Linlin Li and Jinpeng Tian. 'Towards a smarter battery management system: A critical review on battery state of health monitoring methods'. In: *Journal of Power Sources* 405 (2018), pp. 18–29. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2018.10.019.

[16] Zuolu Wang et al. 'A review on online state of charge and state of health estimation for lithium-ion batteries in electric vehicles'. In: *Energy Reports* 7 (2021), pp. 5141–5161. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2021.08.113.

[17] Sunil K. Pradhan and Basab Chakraborty. 'Battery management strategies: An essential review for battery state of health monitoring techniques'. In: *Journal of Energy Storage* 51 (2022), p. 104427. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2022.104427. URL: https://www.sciencedirect.com/science/article/pii/S2352152X22004509.

[18] Abbas Fotouhi et al. 'A review on electric vehicle battery modelling: From Lithium-ion toward Lithium-Sulphur'. In: *Renewable and Sustainable Energy Reviews* 56 (2016), pp. 1008–1021. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.12.009.

[19] Akash Basia et al. 'Review on State of Health estimation methodologies for lithium-ion batteries in the context of circular economy'. In: *CIRP Journal of Manufacturing Science and Technology* 32 (2021), pp. 517–528. ISSN: 1755-5817. DOI: https://doi.org/10.1016/j.cirpj.2021.02.004. URL: https://www.sciencedirect.com/science/article/pii/S1755581721000249.

[20] Dai Haifeng, Wei Xuezhe and Sun Zechang. 'A new SOH prediction concept for the power lithium-ion battery used on HEVs'. In: *2009 IEEE Vehicle Power and Propulsion Conference*. 2009, pp. 1649–1653. DOI: 10.1109/VPPC.2009.5289654.

[21] Xuezhe Wei, Bing Zhu and Wei Xu. 'Internal Resistance Identification in Vehicle Power Lithium-Ion Battery and Application in Lifetime Evaluation'. In: *2009 International Conference on Measuring Technology and Mechatronics Automation*. Vol. 3. 2009, pp. 388–392. DOI: 10.1109/ICMTMA.2009.468.

[22] Jürgen Remmlinger et al. 'State-of-health monitoring of lithium-ion batteries in electric vehicles by on-board internal resistance estimation'. In: *Journal of Power Sources* 196.12 (2011). Selected papers presented at the 12th Ulm ElectroChemical Talks (UECT):2015 Technologies on Batteries and Fuel Cells, pp. 5357–5363. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2010.08.035. URL: https://www.sciencedirect.com/science/article/pii/S0378775310013534.

[23] Laurence A. Middlemiss et al. 'Characterisation of batteries by electrochemical impedance spectroscopy'. In: *Energy Reports* 6 (2020), pp. 232–241. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2020.03.029. URL: https://www.sciencedirect.com/science/article/pii/S2352484720303103.

[24] Uwe Tröltzsch, Olfa Kanoun and Hans-Rolf Tränkler. 'Characterizing aging effects of lithium ion batteries by impedance spectroscopy'. In: *Electrochimica Acta* 51.8 (2006), pp. 1664–1672. ISSN: 0013-4686. DOI: https://doi.org/10.1016/j.electacta.2005.02.148. URL: https://www.sciencedirect.com/science/article/pii/S0013468605007899.

[25] Chaofan Li et al. 'SOH estimation method for lithium-ion batteries based on an improved equivalent circuit model via electrochemical impedance spectroscopy'. In: *Journal of Energy Storage* 86 (2024), p. 111167. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2024.111167. URL: https://www.sciencedirect.com/science/article/pii/S2352152X24007515.

[26] Marvin Messing, Tina Shoa and Saeid Habibi. 'Estimating battery state of health using electrochemical impedance spectroscopy and the relaxation effect'. In: *Journal of Energy Storage* 43 (2021), p. 103210. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.103210. URL: https://www.sciencedirect.com/science/article/pii/S2352152X21009087.

[27] Alexandros Ch. Lazanas and Mamas I. Prodromidis. 'Electrochemical Impedance Spectroscopy-A Tutorial'. In: *ACS Measurement Science Au* 3.3 (2023), pp. 162–193. DOI: 10.1021/acsmeasuresciau.2c00070. URL: https://doi.org/10.1021/acsmeasuresciau.2c00070.

[28] Akram Eddahech, Olivier Briat and Jean-Michel Vinassa. 'Determination of lithium-ion battery state-of-health based on constant-voltage charge phase'. In: *Journal of Power Sources* 258 (2014), pp. 218–227. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2014.02.020. URL: https://www.sciencedirect.com/science/article/pii/S0378775314002031.

[29] Zhonghua Yun and Wenhu Qin. 'Remaining Useful Life Estimation of Lithium-Ion Batteries Based on Optimal Time Series Health Indicator'. In: *IEEE Access* 8 (2020), pp. 55447–55461. DOI: 10.1109/ACCESS.2020.2981947.

[30] Jingwen Wei, Guangzhong Dong and Zonghai Chen. 'Remaining Useful Life Prediction and State of Health Diagnosis for Lithium-Ion Batteries Using Particle Filter and Support Vector Regression'. In: *IEEE Transactions on Industrial Electronics* 65.7 (2018), pp. 5634–5643. DOI: 10.1109/TIE.2017.2782224.

[31] Xiaoyu Li et al. 'State-of-health estimation for Li-ion batteries by combing the incremental capacity analysis method with grey relational analysis'. In: *Journal of Power Sources* 410-411 (2019), pp. 106–114. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2018.10.069.

[32] Alejandro Gismero et al. 'Electric vehicle battery state of health estimation using Incremental Capacity Analysis'. In: *Journal of Energy Storage* 64 (2023), p. 107110. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2023.107110.

[33] Ira Bloom et al. 'Differential voltage analyses of high-power, lithium-ion cells: 1. Technique and application'. In: *Journal of Power Sources* 139.1 (2005), pp. 295–303. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2004.07.021.

[34] Jiangong Zhu et al. 'Investigation of lithium-ion battery degradation mechanisms by combining differential voltage analysis and alternating current impedance'. In: *Journal of Power Sources* 448 (2020), p. 227575. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2019.227575.

[35] V. Bobanac, H. Bašić and H. Pandžić. 'A method for deriving battery one-way efficiencies'. In: *Journal of Energy Storage* 73 (2023), p. 108815. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2023.108815. URL: https://www.sciencedirect.com/science/article/pii/S2352152X23022120.

[36] D.U. Sauer. 'BATTERIES | Charge-Discharge Curves'. In: *Encyclopedia of Electrochemical Power Sources*. Ed. by Jürgen Garche. Amsterdam: Elsevier, 2009, pp. 443–451. ISBN: 978-0-444-52745-5. DOI: https://doi.org/10.1016/B978-044452745-5.00052-6.

[37] Jianqiang Kang et al. 'A novel way to calculate energy efficiency for rechargeable batteries'. In: *Journal of Power Sources* 206 (2012), pp. 310–314. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2012.01.105. URL: https://www.sciencedirect.com/science/article/pii/S0378775312002066.

[38] Eduardo Redondo-Iglesias, Pascal Venet and Serge Pelissier. 'Efficiency Degradation Model of Lithium-Ion Batteries for Electric Vehicles'. In: *IEEE Transactions on Industry Applications* 55.2 (2019), pp. 1932–1940. DOI: 10.1109/TIA.2018.2877166.

[39] Wenhua H. Zhu et al. 'Energy efficiency and capacity retention of Ni-MH batteries for storage applications'. In: *Applied Energy* 106 (2013), pp. 307–313. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2012.12.025. URL: https://www.sciencedirect.com/science/article/pii/S0306261912009075.

[40] A.-I. Stroe, V. Knap and D.-I. Stroe. 'Comparison of lithium-ion battery performance at beginning-of-life and end-of-life'. In: *Microelectronics Reliability* 88-90 (2018). 29th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF 2018), pp. 1251–1255. ISSN: 0026-2714. DOI: https://doi.org/10.1016/j.microrel.2018.07.077.

[41] Fangfang Yang et al. 'A study of the relationship between coulombic efficiency and capacity degradation of commercial lithium-ion batteries'. In: *Energy* 145 (2018), pp. 486–495. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2017.12.144. URL: https://www.sciencedirect.com/science/article/pii/S0360544217321874.

[42] Kun Qian et al. 'State-of-health (SOH) evaluation on lithium-ion battery by simulating the voltage relaxation curves'. In: *Electrochimica Acta* 303 (2019), pp. 183–191. ISSN: 0013-4686. DOI: https://doi.org/10.1016/j.electacta.2019.02.055. URL: https://www.sciencedirect.com/science/article/pii/S001346861930307X.

[43] Issam Baghdadi et al. 'State of health assessment for lithium batteries based on voltage-time relaxation measure'. In: *Electrochimica Acta* 194 (2016), pp. 461–472. ISSN: 0013-4686. DOI: https://doi.org/10.1016/j.electacta.2016.02.109.

[44] C. Uhlmann et al. 'In situ detection of lithium metal plating on graphite in experimental cells'. In: *Journal of Power Sources* 279 (2015). 9th International Conference on Lead-Acid Batteries - LABAT 2014, pp. 428–438. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2015.01.046.

[45] Christian von Lüders et al. 'Lithium plating in lithium-ion batteries investigated by voltage relaxation and in situ neutron diffraction'. In: *Journal of Power Sources* 342 (2017), pp. 17–23. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2016.12.032.

[46] N. E. Galushkin, N. N. Yazvinskaya and D. N. Galushkin. 'Model of Relaxation Processes in Batteries'. In: *ECS Electrochemistry Letters* 4.8 (2015). DOI: 10.1149/2.0091508eel. URL: https://dx.doi.org/10.1149/2.0091508eel.

[47] Lei Pei et al. 'Development of a voltage relaxation model for rapid open-circuit voltage prediction in lithium-ion batteries'. In: *Journal of Power Sources* 253 (2014), pp. 412–418. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2013.12.083. URL: https://www.sciencedirect.com/science/article/pii/S0378775313020624.

[48] Qiaohua Fang et al. 'A State of Health Estimation Method for Lithium-Ion Batteries Based on Voltage Relaxation Model'. In: *Energies* 12.7 (2019). ISSN: 1996-1073. DOI: 10.3390/en12071349. URL: https://www.mdpi.com/1996-1073/12/7/1349.

[49] A. Lamorgese, R. Mauri and B. Tellini. 'Electrochemical-thermal P2D aging model of a LiCoO2/graphite cell: Capacity fade simulations'. In: *Journal of Energy Storage* 20 (2018), pp. 289–297. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2018.08.011.

[50] Dafang Wang et al. 'An electrochemical-thermal model of lithium-ion battery and state of health estimation'. In: *Journal of Energy Storage* 47 (2022), p. 103528. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.103528.

[51] Ali Jokar et al. 'Review of simplified Pseudo-two-Dimensional models of lithium-ion batteries'. In: *Journal of Power Sources* 327 (2016), pp. 44–55. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2016.07.036. URL: https://www.sciencedirect.com/science/article/pii/S0378775316308916.

[52] Jamie Gomez et al. 'Equivalent circuit model parameters of a high-power Li-ion battery: Thermal and state of charge effects'. In: *Journal of Power Sources* 196.10 (2011), pp. 4826–4831. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2010.12.107.

[53] Manh-Kien Tran et al. 'A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters'. In: *Journal of Energy Storage* 43 (2021), p. 103252. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.103252.

[54] Xiaolei Bian, Longcheng Liu and Jinying Yan. 'A model for state-of-health estimation of lithium ion batteries based on charging profiles'. In: *Energy* 177 (2019), pp. 57–65. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2019.04.070.

[55] Zhong Ren and Changqing Du. 'A review of machine learning state-of-charge and state-of-health estimation algorithms for lithium-ion batteries'. In: *Energy Reports* 9 (2023), pp. 2993–3021. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2023.01.108.

[56] Kai Luo et al. 'A review of deep learning approach to predicting the state of health and state of charge of lithium-ion batteries'. In: *Journal of Energy Chemistry* 74 (2022), pp. 159–173. ISSN: 2095-4956. DOI: https://doi.org/10.1016/j.jechem.2022.06.049.

[57] Alvin Barbier et al. 'Analysis of Real-Driving Data Variability for Connected Vehicle Diagnostics'. In: *IFAC-PapersOnLine* 55.24 (2022). 10th IFAC Symposium on Advances in Automotive Control AAC 2022, pp. 45–50. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.10.260.

[58] Chen Ling. 'A review of the recent progress in battery informatics'. In: *npj Computational Materials* 8 (2022), p. 33. DOI: https://doi.org/10.1038/s41524-022-00713-x.

[59] Warren S. McCulloch and Walter Pitts. 'A logical calculus of the ideas immanent in nervous activity'. In: *Mathematical Biophysics* 5 (1943), pp. 115–133. DOI: https://doi.org/10.1007/BF02478259.

[60] A. L. Samuel. 'Some Studies in Machine Learning Using the Game of Checkers'. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[61] Frank Rosenblatt. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65 6 (1958), pp. 386–408. URL: https://api.semanticscholar.org/CorpusID:12781225.

[62] K. Fukushima. 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36 (1980), pp. 193–202. DOI: doi.org/10.1007/BF00344251.

[63] Hopfield JJ. 'Neural networks and physical systems with emergent collective computational abilities'. In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

[64] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[65] Kurt Hornik, Maxwell Stinchcombe and Halbert White. 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[66] Christopher Watkins. 'Learning From Delayed Rewards'. In: (Jan. 1989).

[67] Tin Kam Ho. 'Random decision forests'. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[68] Corinna Cortes and Vladimir Vapnik. 'Support-vector networks'. In: *Machine Learning* 20 (1995), pp. 273–297. ISSN: 0893-6080. DOI: https://doi.org/10.1007/BF00994018. URL: https://link.springer.com/article/10.1007/BF00994018.

[69] Sepp Hochreiter and Jürgen Schmidhuber. 'Long Short-Term Memory'. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[70] David Silver et al. 'Mastering the game of Go without human knowledge'. In: *Nature* 550 (2017), pp. 354–359. DOI: https://doi.org/10.1038/nature24270. URL: https://www.nature.com/articles/nature24270.

[71] Michael Eisenstein. 'Artificial intelligence powers protein-folding predictions'. In: *Nature* 599 (2021), pp. 706–708. DOI: https://doi.org/10.1038/nature24270. URL: https://www.nature.com/articles/d41586-021-03499-y.

[72] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[73] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[74] Randy Jalem, Masanobu Nakayama and Toshihiro Kasuga. 'An efficient rule-based screening approach for discovering fast lithium ion conductors using density functional theory and artificial neural networks'. In: *J. Mater. Chem. A* 2 (3 2014), pp. 720–734. DOI: 10.1039/C3TA13235H. URL: http://dx.doi.org/10.1039/C3TA13235H.

[75] Austin D. Sendek et al. 'Holistic computational structure screening of more than 12000 candidates for solid lithium-ion conductor materials'. In: *Energy Environmental Sciencies* 10 (1 2017), pp. 306–320. DOI: 10.1039/C6EE02697D. URL: http://dx.doi.org/10.1039/C6EE02697D.

[76] Austin D. Sendek et al. 'Machine Learning-Assisted Discovery of Solid Li-Ion Conducting Materials'. In: *Chemistry of Materials* 31.2 (2019), pp. 342–352. DOI: 10.1021/acs.chemmater.8b03272. eprint: https://doi.org/10.1021/acs.chemmater.8b03272. URL: https://doi.org/10.1021/acs.chemmater.8b03272.

[77] Zhisen Jiang et al. 'Machine-learning-revealed statistics of the particle-carbon/binder detachment in lithium-ion battery cathodes'. In: *Nature Communications* 11.1 (2020). ISSN: 2041-1723. DOI: 10.1038/s41467-020-16233-5. URL: http://dx.doi.org/10.1038/s41467-020-16233-5.

[78] Orkun Furat et al. 'Mapping the architecture of single lithium ion electrode particles in 3D, using electron backscatter diffraction and machine learning segmentation'. In: *Journal of Power Sources* 483 (2021), p. 229148. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2020.229148. URL: https://www.sciencedirect.com/science/article/pii/S0378775320314403.

[79] Lukas Petrich et al. 'Crack detection in lithium-ion cells using machine learning'. In: *Computational Materials Science* 136 (2017), pp. 297–305. ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2017.05.012. URL: https://www.sciencedirect.com/science/article/pii/S0927025617302422.

[80] Tianyu Li et al. 'Cost, performance prediction and optimization of a vanadium flow battery by machine-learning'. In: *Energy & Environmental Science* 13.11 (2020), pp. 4353–4361. ISSN: 1754-5706. DOI: 10.1039/d0ee02543g. URL: http://dx.doi.org/10.1039/d0ee02543g.

[81] Peter M. Attia et al. 'Closed-loop optimization of fast-charging protocols for batteries with machine learning'. In: *Nature* 578.7795 (Feb. 2020), pp. 397–402. ISSN: 1476-4687. DOI: 10.1038/s41586-020-1994-5. URL: http://dx.doi.org/10.1038/s41586-020-1994-5.

[82] Gyouho Cho, DI ZHU and Jeffrey Campbell. 'A Comparative Study of Recurrent Neural Network Architectures for Battery Voltage Prediction'. In: *SAE Technical Paper Series*. FFL. SAE International, 2021. DOI: 10.4271/2021-01-1252. URL: http://dx.doi.org/10.4271/2021-01-1252.

[83] Giacomo Capizzi, Francesco Bonanno and Giuseppe M. Tina. 'Recurrent Neural Network-Based Modeling and Simulation of Lead-Acid Batteries Charge-Discharge'. In: *IEEE Transactions on Energy Conversion* 26.2 (2011), pp. 435–443. DOI: 10.1109/TEC.2010.2095015.

[84] Ruxiu Zhao et al. 'A Compact Methodology Via a Recurrent Neural Network for Accurate Equivalent Circuit Type Modeling of Lithium-Ion Batteries'. In: *IEEE Transactions on Industry Applications* 55.2 (2019), pp. 1922–1931. DOI: 10.1109/TIA.2018.2874588.

[85] Jichao Hong, Zhenpo Wang and Yongtao Yao. 'Fault prognosis of battery system based on accurate voltage abnormity prognosis using long short-term memory neural networks'. In: *Applied Energy* 251 (2019), p. 113381. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2019.113381.

[86] Di Zhu, Jeffrey Joseph Campbell and Gyouho Cho. 'Battery Voltage Prediction Using Neural Networks'. In: *2021 IEEE Transportation Electrification Conference & Expo (ITEC)*. 2021, pp. 807–812. DOI: 10.1109/ITEC51675.2021.9490081.

[87] Zhong Ren and Changqing Du. 'A review of machine learning state-of-charge and state-of-health estimation algorithms for lithium-ion batteries'. In: *Energy Reports* 9 (2023), pp. 2993–3021. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2023.01.108. URL: https://www.sciencedirect.com/science/article/pii/S235248472300118X.

[88] Carlos Vidal et al. 'Machine Learning Applied to Electrified Vehicle Battery State of Charge and State of Health Estimation: State-of-the-Art'. In: *IEEE Access* 8 (2020), pp. 52796–52814. DOI: 10.1109/ACCESS.2020.2980961.

[89] Kristen A. Severson et al. 'Data-driven prediction of battery cycle life before capacity degradation'. In: *Nature Energy* 4 (2019), pp. 383–391. ISSN: 0378-7753. DOI: https://doi.org/10.1038/s41560-019-0356-8.

[90] Gae-won You, Sangdo Park and Dukjin Oh. 'Real-time state-of-health estimation for electric vehicle batteries: A data-driven approach'. In: *Applied Energy* 176 (2016), pp. 92–103. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2016.05.051. URL: https://www.sciencedirect.com/science/article/pii/S0306261916306456.

[91] Meng Wei et al. 'State-of-health estimation and remaining useful life prediction of lithium-ion batteries based on extreme learning machine'. In: *Journal of Physics: Conference Series* 1983.1 (July 2021), p. 012058. DOI: 10.1088/1742-6596/1983/1/012058. URL: https://dx.doi.org/10.1088/1742-6596/1983/1/012058.

[92] Shuzhi Zhang et al. 'Synchronous estimation of state of health and remaining useful lifetime for lithium-ion battery using the incremental capacity and artificial neural networks'. In: *Journal of Energy Storage* 26 (2019), p. 100951. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2019.100951. URL: https://www.sciencedirect.com/science/article/pii/S2352152X19307340.

[93] Sahar Khaleghi et al. 'Online health diagnosis of lithium-ion batteries based on nonlinear autoregressive neural network'. In: *Applied Energy* 282 (2021), p. 116159. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.116159. URL: https://www.sciencedirect.com/science/article/pii/S0306261920315671.

[94] Zhiquan Cui et al. 'State of health estimation for lithium-ion battery based on the coupling-loop nonlinear autoregressive with exogenous inputs neural network'. In: *Electrochimica Acta* 393 (2021), p. 139047. ISSN: 0013-4686. DOI: https://doi.org/10.1016/j.electacta.2021.139047. URL: https://www.sciencedirect.com/science/article/pii/S0013468621013372.

[95] Weihan Li et al. 'Online capacity estimation of lithium-ion batteries with deep long short-term memory networks'. In: *Journal of Power Sources* 482 (2021), p. 228863. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2020.228863.

[96] Seongyoon Kim et al. 'Forecasting state-of-health of lithium-ion batteries using variational long short-term memory with transfer learning'. In: *Journal of Energy Storage* 41 (2021), p. 102893. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.102893. URL: https://www.sciencedirect.com/science/article/pii/S2352152X21006101.

[97] Yaxiang Fan et al. 'A novel deep learning framework for state of health estimation of lithium-ion battery'. In: *Journal of Energy Storage* 32 (2020), p. 101741. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2020.101741. URL: https://www.sciencedirect.com/science/article/pii/S2352152X20315784.

[98] Lucian Ungurean, Mihai V. Micea and Gabriel Cârstoiu. 'Online state of health prediction method for lithium-ion batteries, based on gated recurrent unit neural networks'. In: *International Journal of Energy Research* 44.8 (2020), pp. 6767–6777. DOI: https://doi.org/10.1002/er.5413. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.5413. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.5413.

[99] Jiangong Zhu et al. 'Data-driven capacity estimation of commercial lithium-ion batteries from voltage relaxation'. In: *Nature Communications* 13.1 (2022). DOI: 10.1038/s41467-022-29837-w. URL: https://doi.org/10.1038%5C%2Fs41467-022-29837-w.

[100] Guodong Fan and Xi Zhang. 'Battery capacity estimation using 10-second relaxation voltage and a convolutional neural network'. In: *Applied Energy* 330 (2023), p. 120308. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2022.120308.

[101] F. Chollet and F. Chollet. *Deep Learning with Python, Second Edition*. Manning, 2021. ISBN: 9781617296864.

[102] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN: 9780071154673. URL: https://books.google.it/books?id=EoYBngEACAAJ.

[103] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[104] T. Hastie, R. Tibshirani and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846.

[105] D. Opitz and R. Maclin. 'Popular Ensemble Methods: An Empirical Study'. In: *Journal of Artificial Intelligence Research* 11 (Aug. 1999), pp. 169–198. ISSN: 1076-9757. DOI: 10.1613/jair.614. URL: http://dx.doi.org/10.1613/jair.614.

[106] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[107] J. R. Quinlan. 'Induction of decision trees'. In: 1.1 (1986), pp. 81–106. ISSN: 1573-0565. DOI: 10.1007/bf00116251. URL: http://dx.doi.org/10.1007/bf00116251.

[108] L. Breiman et al. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN: 9780412048418.

[109] Luís Torgo. 'Regression Trees'. In: *Encyclopedia of Machine Learning and Data Mining*. Springer US, 2016, pp. 1–4. ISBN: 9781489975027. DOI: 10.1007/978-1-4899-7502-7_717-1. URL: http://dx.doi.org/10.1007/978-1-4899-7502-7_717-1.

[110] Leo Breiman. 'Random Forests'. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/a:1010933404324.

[111] Ian J. Goodfellow, Oriol Vinyals and Andrew M. Saxe. *Qualitatively characterizing neural network optimization problems*. 2015. URL: https://arxiv.org/abs/1412.6544.

[112] John Duchi, Elad Hazan and Yoram Singer. 'Adaptive Subgradient Methods for Online Learning and Stochastic Optimization'. In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 2121–2159. ISSN: 1532-4435.

[113] Alex Graves. 'Neural Networks'. In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg, 2012, pp. 15–35. ISBN: 9783642247972. DOI: 10.1007/978-3-642-24797-2_3. URL: http://dx.doi.org/10.1007/978-3-642-24797-2_3.

Acknowledgments

La conoscenza rende gli uomini liberi.

Many people have asked me why I decided to begin a PhD at the age of 30. My answer can be summed up in a single belief: the only true path to freedom is through knowledge. Only by acquiring knowledge can people make well-informed decisions and, in doing so, attain real freedom.

I would like to express my deepest gratitude to all the exceptional individuals who have supported and guided me throughout this transformative journey. Their encouragement and expertise enabled me to successfully complete my doctorate despite the many unforeseen challenges along the way.

First and foremost, I extend my heartfelt thanks to my academic supervisor, Enrico Stalio, and my industrial supervisor, Federico Brusiani, for their invaluable guidance and mentorship. Their insights and support played a pivotal role in shaping my research and helping me navigate this journey.

I am also sincerely grateful to Thierry Barutaud and Peter Culver May for giving me the incredible opportunity to embark on this path.

A special thank you to Ferrari SpA for hosting me throughout my PhD and for allowing me to work on cutting-edge, highly relevant research topics. It has been an extraordinary experience.

To my colleagues, I deeply appreciate the stimulating exchanges of ideas and the personal connections we have built. Your support and collaboration have made this journey even more rewarding.

Finally, my deepest and most heartfelt gratitude goes to my family: my mom, my dad, my sister, my brother-in-law for supporting me in all the difficulties faced along the path. Their belief in me has been a constant source of strength, and I could not have achieved this milestone without them.

A very special mention goes to my newborn niece Beatrice, whose arrival has brought immense joy to our family. Though she is just beginning her journey in this world, her spontaneity and simplicity are already a source of inspiration and a reminder of the beauty of new beginnings.