Ingeniaritza Konputazionala eta
Sistema Adimentsuak Unibertsitate Masterra

Máster Universitario en Ingeniería Computacional
y Sistemas Inteligentes

Konputazio Zientziak eta Adimen Artifiziala Saila

Departamento de Ciencias de la Computación e Inteligencia Artificial

Master Tesia
Tesis de Máster

Lithium-Ion Battery Remaining Useful Life Prediction

Beñat Larrarte Lizarralde

Zuzendaritza
Dirección

Josu Ceberio Uribe

Elixabete Ayerbe Olano

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Konputazio Ingeniaritza eta Sistema Adimentsuak Unibertsitate Masterra
Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo Fin de Máster

Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

Lithium-Ion Battery Remaining Useful Life Prediction

Beñat Larrarte Lizarralde

Advisors
Josu Ceberio Uribe
Elixabete Ayerbe Olano

22 June 2021

Acknowledgments

It's been 15 weeks since I started working on this project and fortunately it seems that my computer science carrier will continue in this way. Thus, I would like to thank Basque Country University (EHU/UPV) and CIDETEC for making this placement possible.

Although I had an initial background on time series forecasting, this project has been a challenge due to my lack of knowledge about rechargeable batteries field. Even so, we have been able to overcome the issue working together. According the experience of working full time in a long time project, sometimes it looks like surfing a wave. When you are on the crest, everything goes fluently and you feel motivated and happy with the work fulfilled. On the other hand, when you are facing a problem, it is vital to find your way to come back. Here, I would like to show my gratitude to project tutors Elixabete Ayerbe Olano and Pablo Cereijo García from CIDETEC, and Josu Ceberio Uribe, from EHU, for taking part of this work and assisting during the stay.

Finally, I would like to show my appreciation to all that I work with for considering me just as another one, and for giving me the space and time I need to feel comfortable. Therefore, I feel gratified and pleased with the work performed during these weeks.

Summary

Accurate prediction of the remaining useful life (RUL) of lithium-ion batteries (LIB) can improve the durability, reliability, and maintainability of battery system operation in different scenarios such as, electric vehicles. To achieve high-accuracy RUL prediction, it is necessary to develop an effective method for long-term nonlinear degradation prediction and quantify the uncertainty of the prediction results. The RUL prediction for lithium-ion batteries (LIB) is a challenging task due to long-term dependencies among the capacity degradation.

In this work, we propose RUL prediction models based on long short-term memory (LSTM). It is employed to learn the long-term dependencies among the degraded capacities of lithium-ion batteries. In order to estimate RUL, an introduction to Severson fast-charging battery dataset are done, complemented by outlier detection and feature selection considering multiple measurable data such as, the evolution of internal resistance and temperature.

Using Severson LIB dataset, we verify the accuracy of the proposed LSTM-based RUL prediction taking into consideration different variables (univariate, bivariate and multivariate). Conducted experiments reveal that the proposed univariate LSTM model outperforms bivariate and multivariate LSTM model learnt. Furthermore, proposed univariate model achieve less that 0.1 of mean absolute error (MAE) in short- and long-term prediction. In spite of this, a sign of overfitting is observed that leads to further investigation.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

In the first chapter, an introduction to CIDETEC and project framework is presented. Moreover, the background of battery life prediction as well as the goals of the work are introduced.

1.1 Cidetec

CIDETEC is a private organization for applied research founded in 1997. Located in the city of Donostia-San Sebastián, CIDETEC is comprised of three international technological reference institutes in Energy Storage, Surface engineering and Nanomedicine.

CIDETEC Energy Storage is specialised in creating new battery technologies according to specific challenges, and its ultimate transference to the industry. The institute has the capacity to develop complete products and processes and offers material validation, pilot manufacture, pack engineering and battery testing services. CIDETEC overall workforce consists of 180 employees, 95 % of whom are university graduates and 45 % PhD holders. Its volume of activity came up to 13.1M€ in 2018.

Energy Storage involves up to 57 specialized researchers distributed into two technological units: Materials for Energy –approx. 2/3 of the workforce with a background of chemistry, electrochemistry, mathematics and materials- and Systems Engineering - mainly electrical and mechanical engineers-. As a result of this approach, CIDETEC Energy Storage cooperates closely with the industry in the context of bilateral, direct contract research and product development projects, both at the National and International level.

CIDETEC Energy Storage is heavily involved in the European batteries community, including several EU level platforms and associations where energy storage and its applications is at the core. We are active member in several industrial and research associations and platforms like the ETIP Batteries Europe, the European Battery Alliance (EBA), EGVIA, EARPA, EMIRI or EERA.

Moreover, it is also regular participant in H2020 European funded projects, frequently from the coordination of international consortia –currently CIDETEC Energy Storage is coordinator of iModBatt, iHeCoBatt, and E-Magic projects, plus one more from the last CC-BAT-2019 calls-. We accumulate more than 15 participations in EU funded R+D battery

projects under FP7 and H2020. Finally, CIDETEC Energy Storage is involved as well in the core team of BATTERY 2030+, the initiative for a long-term large-scale research program towards the European battery of the future.

1.2 Project context

This project is a set on to the Battery Interface Genome – Materials Acceleration Platform (*BIG-MAP*)[1]. This project is part of the large-scale and long-term European research initiative *BATTERY 2030+* [2]. BIG-MAP propose a radical paradigm shift in battery innovation, which will lead to a dramatic speed-up in the battery discovery and innovation time; reaching a 5-10 fold increase relative to the current rate of discovery within the next 5-10 years. In addition, BIG-MAP relies on the development of a unique *R&D* infrastructure and accelerated methodology that unites and integrates insights from leading experts, competences and data across the entire battery (discovery) value chain with Artificial Intelligence (*AI*), High Performance Computing (*HPC*), large-scale and high-throughput characterization and and autonomous synthesis robotics. In short, BIG-MAP aims to reinvent the way we invent batteries and to develop core modules and Key Demonstrators of a Materials Acceleration Platform specifically designed for accelerated discovery of battery materials and interfaces (Fig. 1.1).

This will not be achievable in the three years of this project, but the BIG-MAP consortium has identified a set of Specific Objectives and 12 Key Demonstrators for the 3-year ramp-up phase that will develop and demonstrate the infrastructural backbone needed to achieve a 5-10 fold acceleration in the discovery process. In summary, the specific objectives are:

- Develop the scientific and technological building blocks and models for accelerated battery discovery.
- Deliver cross-cutting initiatives to ensure the implementation and use of project data and results throughout the battery discovery value chain.
- Disseminate information/innovations to the battery community throughout the Battery value chain, to boost EU advanced battery development and manufacturing.

1.3 Battery cycle life prediction

In 2021, the battery industry will mark the 30th anniversary of a remarkable scientific invention that led to great commercial success, and the awarding of the 2019 Nobel Prize in Chemistry: *the rechargeable lithium-ion battery*. Three decades of performance improvements have occurred because of innovative research, with enhanced manufacturing efficiency bringing about mass-market penetration. Nowadays, lithium-ion batteries (LIB) are widely used as power sources for many types of systems, including consumer electronics, electric vehicles (*EVs*), owing to their high energy density, high galvanic potential, and long lifetime.

[1] BIG-MAP url : https://www.big-map.eu/.
[2] BATTERY 2030+ url : https://battery2030.eu/.

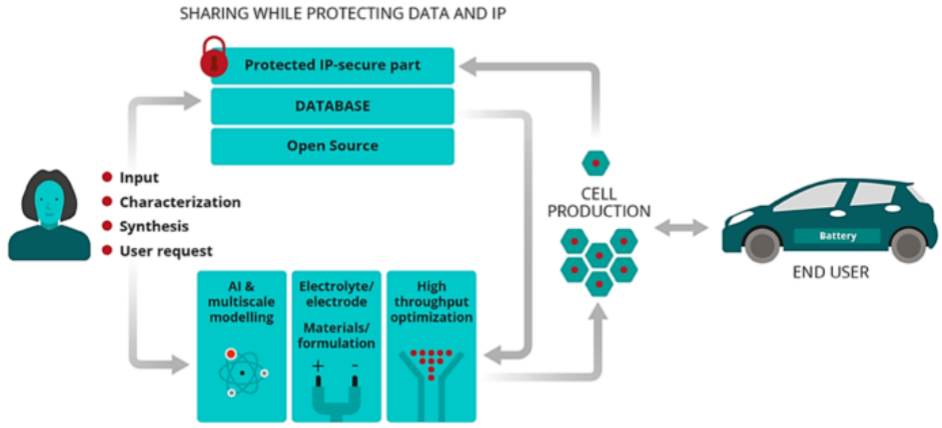
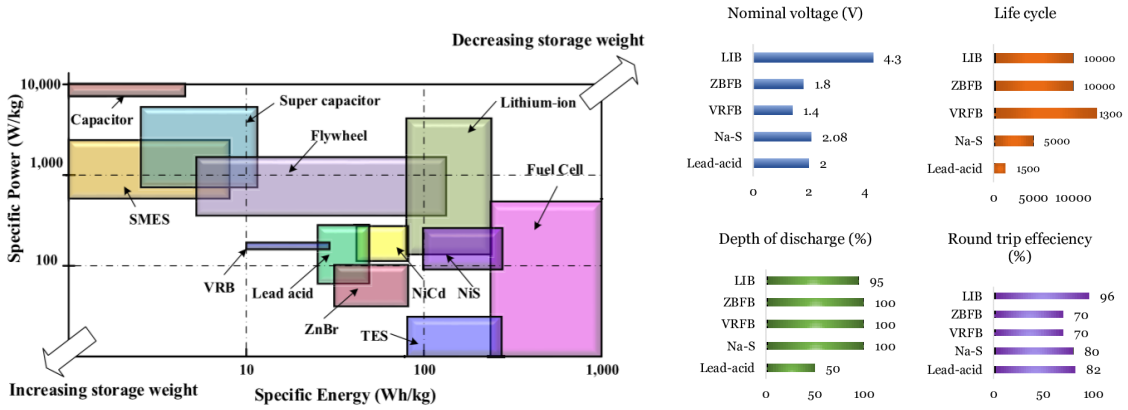


Figure 1.1: A illustration of the BIG-MAP project.

Moreover, with new legislation ending the sale of new petrol and diesel vehicles in the UK by 2030, there is a need to further reduce costs and improve performance to convince consumers that electrification can meet their usage requirements. Improvements are needed in the electric vehicle range, to increase the speed with which batteries can be charged and deliver power (which is enabled by power density levels) and, of course, to safety during both operation and storage.

Figure 1.2a compares the different battery energy storage system where LIB is dominant with the regard to specific power (W/kg) and specific energy (Wh/kg). Besides, the performance comparative analysis between LIB and other EV batteries in terms of nominal voltage, life cycle, depth of discharge and efficiency demonstrates that LIB appears to be a better choice for EV application, as shown in Figure 1.2b.



(a) Specific energy and power comparison among BESS Acronyms. **(b)** Characteristics of different EV batteries

Figure 1.2: Comparison of LIB against other batteries [1].

However, their performance degrades over repeated charge/discharge cycles. For many applications, failure is considered to occur when the battery's capacity decreases below 80%

of its initial capacity. This way, battery useful life or remaining useful life (RUL) is the area of predicting the number of charge–discharge cycles until the battery reaches end-of-life (EOL). As an important indicator of battery health, RUL is closely linked to the reliability and durability of battery system operation. Thus, RUL prediction can provide important information for the battery management system and maintenance. Because the battery's capacity and power tends to drop much faster, its performance is thus unreliable after this point, and it should be replaced. Despite this unreliability on high demanding tasks, they can be used in less necessity assignments.

Battery degradation is a collection of events that leads to a loss of performance over time. It is a successive and complex set of dynamic chemical and physical processes, slowly reducing the amount of mobile lithium ions or charge carries [6]. In order to explain battery degradation, it is useful to first consider what is a cell, where batteries are collectives of individual cells.

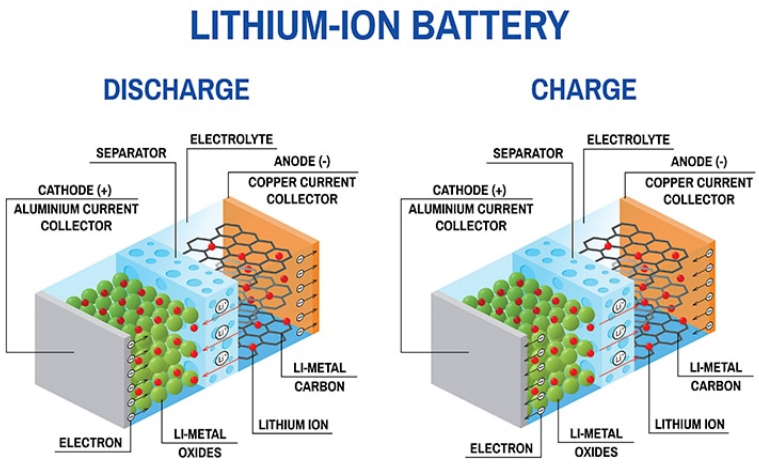


Figure 1.3: Simplified view of a lithium-ion cell composition and functionality.

A cell is composed by two opposing electrodes, cathode and anode, impregnated by an electrolyte solvent and electrically isolated by a porous separator, as a way of preventing a short circuit, Figure 1.3.

In a cell, chemical energy is transformed into electricity through oxidation and reduction processes. When a component of the cell oxidizes, it releases electrons that leave the core of the material, travel through an external circuit, and rejoin the cell through the other end, reducing the material of the opposite electrode. Therefore, it is necessary that the materials be easily reducible and oxidable.

In this way, the electrical charge step is pairing up to a chemical reaction. This type of reaction is known as *redox*, where there is a transfer of electrons from one species to another. These electronic transfers take place in two half-reactions, in which the anode and the cathode participate. In the anode the oxidation of some species is occurred, loss of electrons, while in the cathode the reduction of other species, gain of electrons.

It is important to highlight that the roles of each of the electrodes are exchanged each

time the cell starts charging or discharging. That is why it is not suggested to use cathode and anode expressions in rechargeable cells.

For maximising cells performance, an optimised coating structure or architecture is required. This is directly dependent on a uniform distribution of sub-components with desired properties, and is critical to ensuring complete mass and charge transport during operation. However, there are numerous chemical, electrochemical and physical processes that occur during operation of the battery that can lead to incomplete charge/mass transfer. This invariably results in degradation and eventual failure. It is said that the capacity fade is caused by numerous modes of degradation that lead to an impaired ability of the battery to store power. It is generally motivated by a combination of these three events.

- The loss of electrolyte.
- The loss of the lithium ions in the electrode as a result of mechanisms such as Solid Electrolyte Interphase (*SEI*) growth or/and lithium plating.
- The electrode active materials becoming unavailable for participation in the electrochemistry of the cell.

Regarding battery useful life prediction, the aim is to protect lithium-ion user against unexpected battery dead or in worst cases the cell explosion. This becomes much unpredictable and susceptible when the cell discharge capacity is lower than 80% of nominal capacity. Besides this goal, it could be used also to optimize cell investigation. If a reliable predictive model is achieved, instead of doing a complete charging/discharging experiment of a cell until the threshold, the model will predict the cell durability, reducing time costs of tests.

Motivated by the different contexts of applications, remaining useful life prediction methods for batteries can be categorized into two main families: model-based methods and data-driven methods [7]. Model-based methods use mathematical models to capture the long-term dependencies of battery degradation, which are combined with advance filtering techniques, such as the particle filter (*PF*) algorithm, to predict the battery RUL [8][9]. The filtering process can be used to update the model parameters by evaluating the importance of the data points adaptively based on the measured signals. However, due to the lack of knowledge of the degradation evolution, the mathematical model is usually constructed by fitting the degraded capacities of lithium-ion batteries, but this step could lead to overfitting. Therefore, the model is able to fit the training data accurately but its prediction accuracy is frequently low.

Although significant progress has been made in model-based RUL prediction methods in recent years, two drawbacks still. First, there is no accurate aging model that can be used as a basis for RUL prediction. Second, RUL prediction accuracy based on the most widely used filter method, PF, is limited by the particle degeneracy problem.

In addition to model-based methods, a new branch in the prediction paradigm has emerged with the application of data-driven methods which do not need an explicit mathematical model to describe the degradation evolution of batteries and are only dependent on historical degradation data. Therefore, it should be noted that the uncertainty quantification of prediction results is an important factor affecting final decision-making [10] [11]. These

methods try to extract key degradation information from the data points by a specific learning algorithm.

1.4 Goals

The main goal in this work is to study the relevance of battery charging/discharging experiments features on battery life expectancy. Within the main goal, the first aim is to optimize CIDETEC's feature measurement equipments in order to, firstly, identify the relevant vaiables to collect, and secondly, discover variables that where not considered important, but might be useful in the prediction task.

The second aim is to investigate the connection between collected features with the measurement equipment, and battery durability. To that end, a predictive model is created to forecast the discharge capacity until the 80% of the nominal capacity. This way, apart from the first goal, this enables to speed up the charging/discharging capacity if the predictive model is good enough to rely on.

In order to carry out previous goals, a complete understanding of the working environment is relevant in order to figure out internal relationship between features and perceive incoherent data, such as, outliers. For the purpose of the work, the dataset by Severson et al. [12] will be used. Then the dataset must be analyzed and preprocessed to get an optimal features to train the model. Afterwards, the best models of the literature will be studied and the most relevant ones will be considered for experimental evaluation. Finally, the prediction will be done combining selected data and models.

1.5 Structure of the document

Figure 1.4 shows the structure of this project based the previous mentioned goals. This way, each of the stages represents an important task during the experiment. Moreover, in order to introduce the manuscript, each color represents a chapter. Therefore, it is composed of 5 chapters, each one describing the following:

- Chapter 2 : This chapter introduce the data used in this investigation, in addition to their features and characteristics.
- Chapter 3 : In this chapter the data is preprocessed. This means that a depth analysis of raw data is done in order to, first, detect and correct outliers, and second do an feature selection.

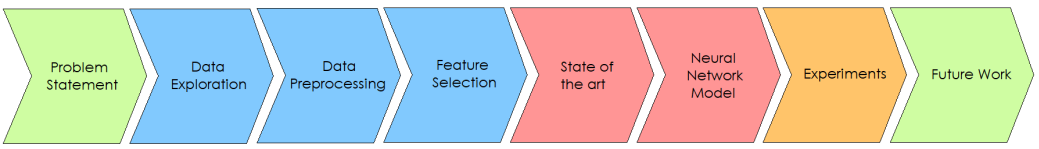


Figure 1.4: Illustration of the project structure divided on stages, where each box represents an important event during the investigation.

- Chapter 4 : This chapter describes the learning scenario of the problem, complemented by the state-of-the-art and the proposed model.
- Chapter 5 : This chapter introduce the experimental framework and four experimental studies in order to a better understanding of the predictive model.
- Chapter 6 : In this last chapter a summary of the work is made. Not limited to that, lines for future improvements are described.

CHAPTER 2

Severson dataset

In this chapter, a broad description of the problem statement, complemented by the Severson dataset [12] is provided. Moreover, an exploratory data analysis is also included, as well as the experiments conditions and irregularities.

2.1 Data collection

Severson dataset [12] consists of 135 commercial lithium-ion batteries cycled to failure under fast-charging conditions. The aim is to simulate a real high demand environment, and approximately 96.700 cycles have been recorded, the largest publicly available for nominally identical commercial lithium-ion batteries cycled under controlled conditions. These lithium-ion phosphate (LFP)/graphite cells, manufactured by A123 Systems (APR18650M1A), were cycled in horizontal cylindrical fixtures on a 48-channel Arbin LBT potentiostat in a forced convection temperature chamber set to 30°C. The cells have a nominal capacity of 1.1 Ah and a nominal voltage of 3.3 V.

The objective of the research work that generated this data is to optimize fast charging for lithium-ion batteries. As such, all cells in this dataset are charged with a one-step or two-step fast-charging policy. First a cell is charged with one-step policy until a specific state-of-charge (SOC, %) at which the currents switch. The second current steps ends at 80% SOC, after which the cells charge at 1C CC-CV, (Fig. 2.1). The upper and lower cutoff potentials are 3.6 V and 2.0 V, respectively, which are consistent with the manufacturer's specifications. Finally, all

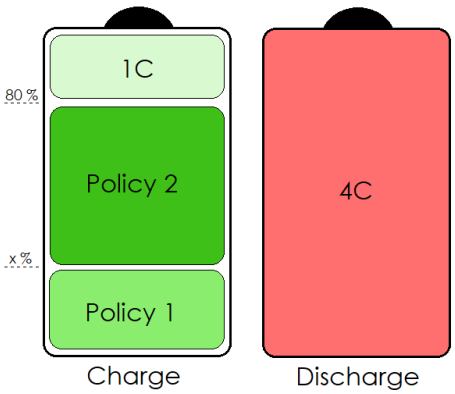


Figure 2.1: Illustration of the experimental design, where *x*, *policy1* and *policy2* attributes are different in each cell experiment.

cells discharge at 4C and rests are placed after reaching 80% SOC during charging and after discharging.

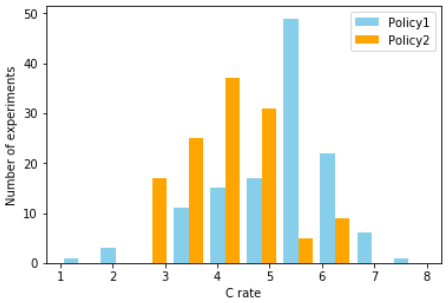
As a way to start analysing data, these two charging policies are illustrated to gain information about the connection of charging conditions and final battery life. It is believed that with high and hard charging and discharging conditions, battery capacity will fall faster, thus getting less number of useful cycle.

It is seen that each of these policies separately do not show any evidences of this relationship between hard charging/discharging conditions and battery life. Therefore, with the aim of gaining useful knowledge, a weighted mean of charging C rate is done, applying the Eq. 2.1, to inference all the information to create a new one that overalls the experiment charging conditions, as the discharge conditions is always the same.

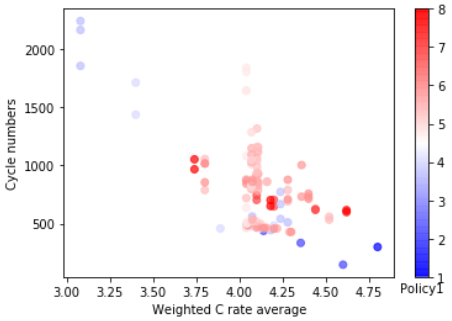
$$\text{Weighted C rate average} = policy1 * x + policy2 * y + 1 * 0.2 \tag{2.1}$$

where x is specific state-of-charge between 0 and 1 to describe the usage of *policy 1*, which is different every experiment. While y is the percentage usage of *policy2* $(y = 0.8 - x)$. This way, $x + y + 0.2 = 1$ is preserved and an overall charging C rate average is calculated.

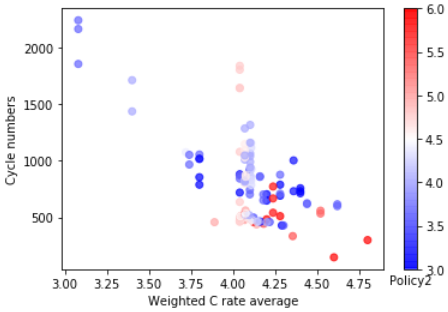
Figure 2.2a shows a visual point of view of the policies used in these experiments. As



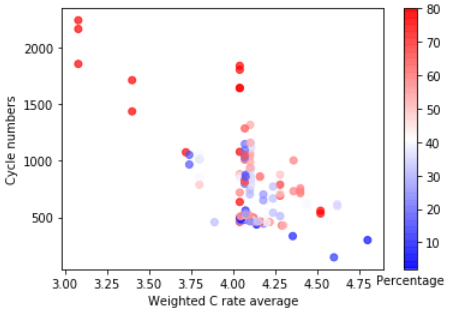
(a) Histogram of C rates used on Severson fast-charging experiments.



(b) A scatter graph where each experiment is illustrated taking into account the durability and the weighted C rate average. While the color introduces the charging policy1.



(c) A scatter graph where the color of each points illustrate the second charging policy used on the experiments.



(d) Relation between the charging C average rates and the number of cycles before end of life.

Figure 2.2: Illustrations of charging policy diversity and its relationship with final battery life.

the experiment consists of fast-charging conditions, low C rates are rarely tested. Note that in order to have fast-charging environment, high C rates must be provided. That's why most of policies are between 3C and 6C, far form common 1C condition.

According to other three Figures 2.2b, 2.2c and 2.2d, a scatter graph is illustrated to describe the connection between C rate used and the durability of the cell. As it was expected, an inverse correlation between high charging policies and battery life. This means, that using high C rates decreases the number of useful cycles of the cell.

Although the weighted C rate average is used to illustrate each fast-charging experiment, in each scatter, the color introduce the charging variables, policy1, policy2 and the percentage of switch. It is interesting to highlight, as it is introduced on Figure 2.2d, that those with high durability use mainly one policy, as the switching percentage is very high.

However, those sequence of experiments with weighted C rate average around 4C were not foreseen. Searching for a reason to describe this results, a lack of replicability is found on *Severson* dataset. In other words, from the observed data and in that particular context (Severson et al. [12]), we conclude that using the same cell and conditions, the ability to get similar results is not guaranteed.

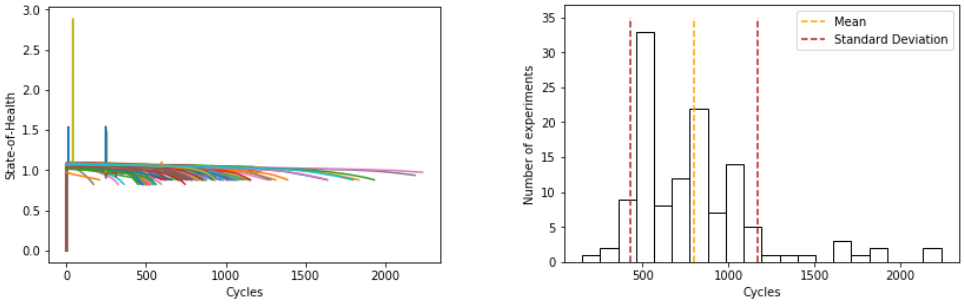
In order to ensure the replicability, a set of tests in the same conditions must be done and compare their results. Despite in theory seems to be easy, in real life conditions might be, somehow, waste of time when a big amount of time is spent to ensure only this, or just because the study has a limited time. That is why usually very few test are done in each conditions, but gaining uncertainty on the way.

Table 2.1 gives example of test done to create this dataset. There the battery cycle life and charging conditions are described. In total 135 tests have been done, divided in 69 different experiments, taking into account that all cells are identical. Moreover, only 25 of 69 tests have more that one repetitions. Not only that, most of them have acquired very varied lifetimes, for instance, *7C(30%)-3.6C* charging policies cells have reached 627 and 966 cycles, in which the difference is huge. In contrast, *5.6C(19%)-4.6C* have obtain almost the same results.

This could be a reason for those experiments around 4C charging average rate. On the other hand, in order to get a more reliable dataset, more repetitions should be done to

Table 2.1: Description of few cells included in dataset, taken from *Severson Supplementary Information (Supplementary Table 9.)* [5].

Cell barcode	Cycle life	Charging policy
EL150800460514	1852	3.6C(80%)-3.6C
EL150800460486	2160	3.6C(80%)-3.6C
EL150800460623	2237	3.6C(80%)-3.6C
EL150800460642	**625**	7C(30%) -3.6C
EL150800463229	**966**	7C(30%) -3.6C
EL150800737276	**817**	5.6C(19%) -4.6C
EL150800737319	**816**	5.6C(19%) -4.6C
EL150800460518	300	1C(4%) -6C
EL150800460634	511	3.6C(30%) -6C



(a) Discharge capacity of all experiments. There are multiple machine error outliers getting more capacity than the maximum of these cells, (1.1 Ah).

(b) Histogram of cell lifetime. Point out that the mean is 798 and most experiments are placed between standard deviation.

Figure 2.3: *Severson* dataset battery cycle life results.

calculate the shape of the variations.

Following with the exploratory analysis, Figure 2.3a illustrates the lifetime of all experiments. On one hand, discharge capacity evolution of all the tests is shown. The degradation is exponential, first the degeneration seems to be meaningless as the capacity seems to be almost maintaining. However, increasing the number of fast charging/discharging cycles, this ability of preserving is loosing until, at the end, the capacity looks like an abyss.

Despite the regular evolution, some unexpected vertical lines appear on the figure. This irregularities have been considered as outliers and they should be taken into account on the data preprocessing in the next chapter.

On the other hand, an histogram is displayed in Figure 2.3b in order to aggregate the lifetime of experiments. It is noteworthy to point out that the minimum life expertise of a cell in these fast-charging dataset is about 128 cycles and the maximum of 2237. The mean is 800 cycles and the standard deviation is 371. Although this numbers seems too low when comparing to a commercial battery lifetime, it is important to highlight that in real usage these batteries will last more cycles, as these experiments are done in a fast charging conditions.

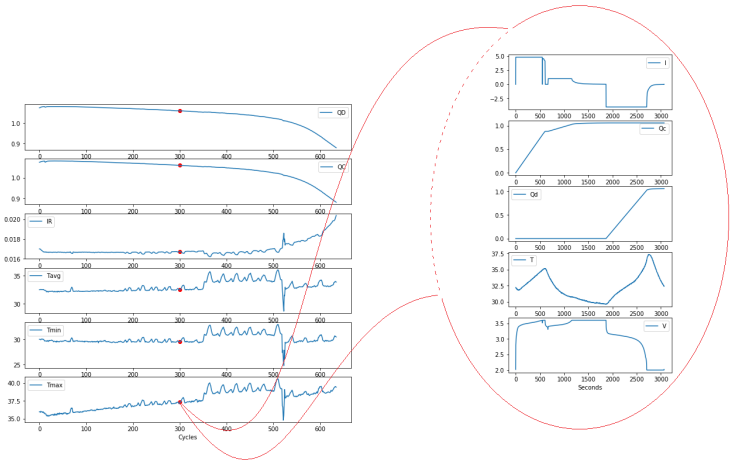
2.2 Exploratory data analysis

In this section all the features of the dataset are, and can be divide in two groups. Firstly, there are those characteristics that represents the whole experiments, such as, the evolution of the capacity or the temperature progression. Each instance of these time series represents a complete fast charging/discharging cycle, named as "features of an experiment".

On the other hand, there are variables that describe the cycles. Features that describe the evolution of internal characteristics of a cycle are named as "features of a single cycle". These features introduces a second dimension to the learning scenario as they add additional information to each variable in each cycle value.

To sum up, Figure 2.4 illustrates these two groups of features. Firstly, on the left, "features of an experiment" are introduced as a multivariate time series with 6 features,

where a time series is multivariate when it has more than one features. Whereas, on the right, another multivariate time series is shown to represent second type of features. Taking into consideration that there are 135 experiments stored in the database, the present learning scenario can be considered as a "bi-dimensional multivariate multiple time series" problem.



(a) Illustration of a two dimensional multivariate time series.

Figure 2.4: A visual picture of an experiment divided in two types of features.

In order to illustrate each of the features, *b2c2* experiment and its second cycle are chosen. In this test 2C and 5C one-step and two-step fast-charging policies are used, respectively. 2% is the threshold to switch these strategies and in total 438 cycles are achieved before the getting 80% of nominal discharge capacity.

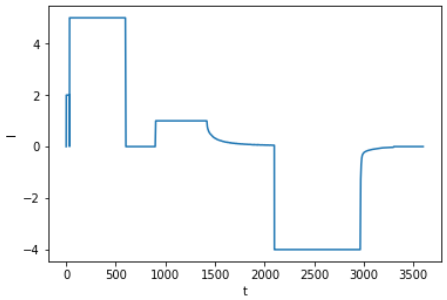
2.2.1 Features of a single cycle

The experiments consists of fast-charging and discharging repetitions in which little by little discharge capacity fades until the cell is considered dead. Meanwhile in each cycle time step 5 features are recorded, current, voltage, charging- and discharging-capacity and temperature.

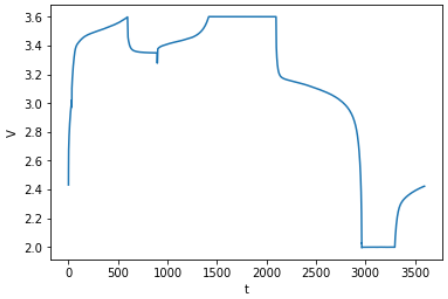
Figure 2.5 shows the current and voltage values at every instant. Regarding current (I) evolution, the charging and discharging policies can be figured out. Note that at the beginning of the cycle, the cell must be charged and then discharged. Therefore, at the beginning of the current time series, three current values are constantly measured. First one-step policy (in Figure 2.5a, 2C) is shown, then second policy (5C) and finally 1C condition. This way, the minimum value is the discharging C rate value set as -4 in all cases, this means, 4C at discharge. On the other hand, the maximum value of current (I) feature is different each time as the charging policies varies in each test, and the maximum value of the time series is the maximum C rate used in the experiment.

Moving to voltage progression, highest and lowest points are lead to experiments upper and lower cutoff, which are 3.6V and 2.0V respectively.

Regarding the capacity, Figure 2.6 expose the cell charging situation. Figure 2.6a shows the charging strategy of one cycle while Figure 2.6b all cycles together. Note that on the

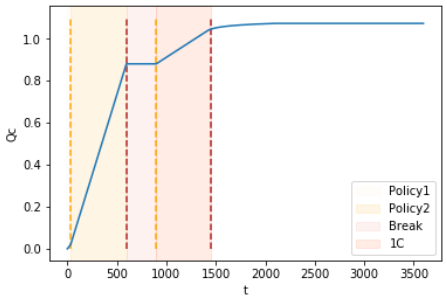


(a) Current value on each time step. Positive values represents cell is charging, while negative ones discharging process. On the other hand, 0 Ah represents break of the current process.

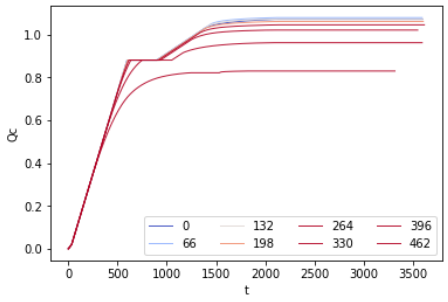


(b) Illustration of voltage used on cycle measurements. Point out, it is not possible put them constant both features at the same time.

Figure 2.5: Illustration of a battery current and voltage on a experiment.



(a) Charging capacity curve of a single cycle. It is divided among different activity. After that discharging process starts.



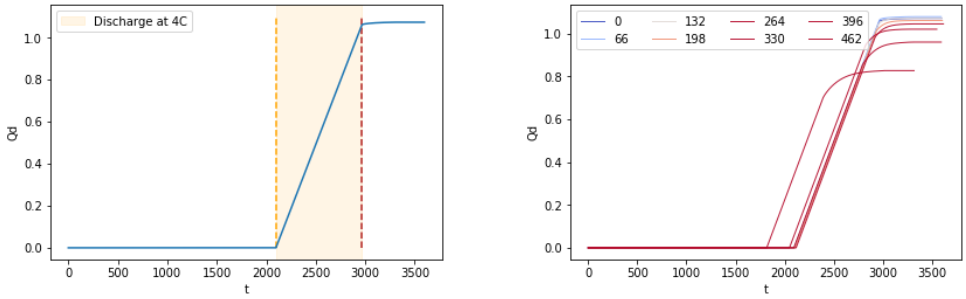
(b) Charging capacity degradation over the experiments. 8 different cycles are shown to describe the evolution through time.

Figure 2.6: Battery charging evolution during a cycle and the whole experiment.

first cycles the cell is charged properly, reaches the maximum capacity (around 1.1Ah). However, as it is illustrated in the second figure, during the experiment it is not possible to charge totally, this means that there is a degradation in the course of a test.

On the other hand, Figure 2.7 describes discharging capacity progression. Note that always, both capacities, charging and discharging, goes from 0Ah to 1.1Ah, the nominal capacity of these cells, to illustrate the capacity evolution during these processes. Figure 2.7a shows the progression of capacity of the first cycle, highlighting the 4C rate is used in the whole process. Finally, discharge capacity degradation is illustrated on the Figure 2.7b. In the same way, maximum range decreases over time and repetition. As a result, when it reaches 80% of total capacity, the cell is taken as unstable and, thus, it is suggested to be replaced.

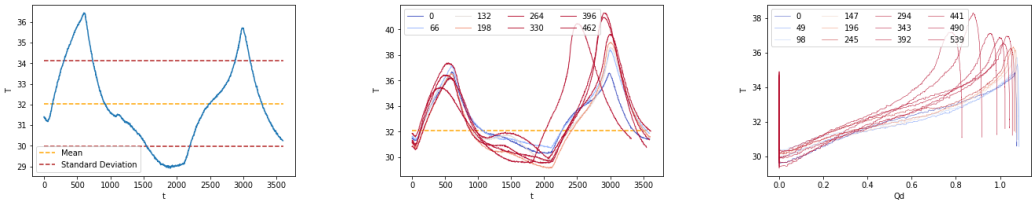
Regarding internal temperature attribute, Figure 2.8 introduces graphical point of view of the feature during a cycle and a complete test. Charging and discharging hard C rates causes the growth of the temperature achieving those two peaks. On the contrary, the break freshen internal heat between them, as it is illustrated on Figure 2.8a and 2.8b.



(a) Charging capacity curve of a single cycle. It is divided among different activity. After that discharging process starts.

(b) Charging capacity degradation over the experiments. 8 different cycles are shown to describe the evolution through time.

Figure 2.7: Battery discharging evolution during a cycle and the whole experiment.



(a) Temperature curve of a single cycle displaying the mean and standard deviation horizontal lines.

(b) Internal temperature evolution over the experiments. Orange horizontal line represents first cycle mean temperature as a way to have a better view of the growth.

(c) Illustration of the relationship during the experiment between temperature and discharge capacity. Showing a connection within detriment of capacity with increment of the heat.

Figure 2.8: Temperature evolution during a cycle and the whole experiment.

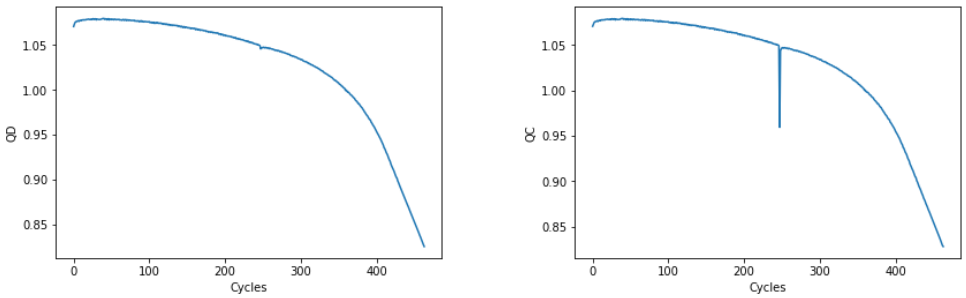
Not limited to that, we observe that, during the experiment the temperature is increasing until 42°C in this example, as it is shown in Figure 2.8b. Finally, internal connection of temperature and discharge capacity is described in Figure 2.8c, concluding that there are a relationship between the incremental of the temperature and the capacity detriment.

2.2.2 Features of an experiment

After introducing all the features of a single cycle, there are six more attributes that corresponds to each particular experiment. However, most of them are a sum up of previous information as these features describes a set of cycles.

The most important property is illustrated in Figure 2.9 accompanied with charging energy progression. In the current work and as the key of the investigation, discharge capacity evolution is the feature which will be forecast. Although the nominal capacity is 1.1Ah in these cells, during the experiments this value is rarely reached. Nevertheless, in this study, as well as, in the *Severson* [12] article, nominal capacity will be used to calculate the battery life threshold.

Moreover, charging capacity progression an anomalous data is being detected. Point



(a) Evolution of discharge capacity during the test. Note that at the beginning the energy is kept, whereas at the end is a steep curve.

(b) Charge capacity progression. Although it is seems very similar, discharge evolution will be used to forecast battery life.

Figure 2.9: Discharge and charge capacity evolution during a experiment.

out that each value of this time series corresponds with a cycle, this means that the whole cycle should be considered as an outlier and special treatment should be done in order not to send unwanted noise to the data-driven model, as it is studied in Chapter 3.

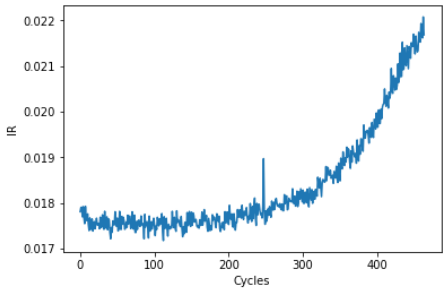
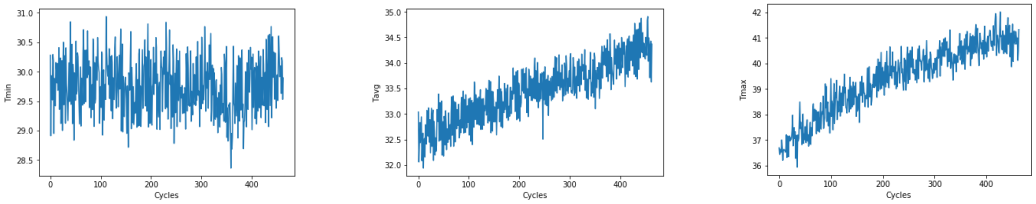


Figure 2.10: Internal resistance evolution during a experiment.

Moving on, even though most of these attributes are a summary of all cycles, *Internal Resistance (IR)* is calculated taking into account the current and the voltage in each cycle. Moreover, while capacity is often used to determine the life of a battery, the end of life could be also settle down from IR. Although in this work discharge capacity (QD) is used.

Opposed to the capacity evolution, internal resistance has a incremental behaviour, as introduced in Figure 2.10. However, this series has small and continues ups and downs that complicates the prediction, similar to a noisy time series. Nevertheless, it will be used as additional helpful feature to the model.

Finally, Figure 2.11 describes the temperature changes through the experiment. As it is expected, the minimum temperature measured is around 29°C, less than the nominal temperature set to 30°C, while the maximum is quantified to be 42°C. Besides, the average and maximum temperature series seem to be interesting to take into account to predict the cell discharge capacity.



(a) The evolution of the minimum temperature measure.

(b) The progression of the average temperature calculated from each cycle.

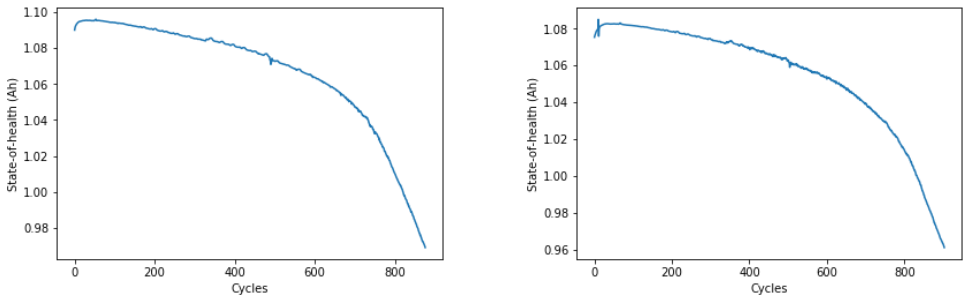
(c) The evolution of the maximum temperature quantified.

Figure 2.11: Minimum, average and maximum temperature evolution during a test.

2.3 Experiments' irregularities

During the experimental process, some irregularities were occurred that lead to unusual values. These experimentation deviations are described on the available *web page* [1] to download the dataset itself. According to what notified:

- The test in channel 13, 19, 21, 22 and 31 for the first batch and 33 and 41 of third batch were terminated before the cells reached 80% of nominal capacity (the dataset is divided into three batches). Figure 2.12 introduces two examples of those 7 experiments. These limited tests will be considered as anomalies, therefore, as events of interest.



(a) The degradation of state-of-health of the *b1c8* cell. Note that 80% of 1.08Ah is 0.86Ah.

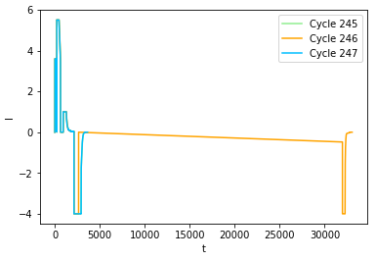
(b) The evolution of state-of-health of the *b1c10* cell.

Figure 2.12: Example of a cells which is not reached to 80% discharge capacity through the test.

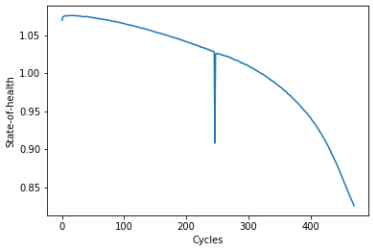
- Unfortunately, in the first batch, the temperature control is somewhat inconsistent, leading to variability in the baseline chamber temperature.
- During some tests uncontrollably stops were come about. In the first batch, the computer automatically restarted twice. As such, there are some time gaps in the data. Similarly in the second batch, the computer automatically restarted, affecting all tests (around cycle 250 for most policies). This effectively lead to around an 8-hour

[1] Project - Data-driven prediction of battery cycle life before capacity degradation : https://data.matr.io/1/.

'rest'. Therefore, there are some peaks in capacity as it is illustrated in Figure 2.13 and 2.14.

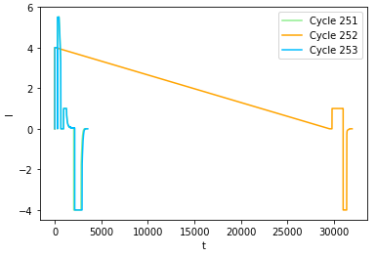


(a) Illustration of the current evolution of the *b2c4* cell at cycle number 246. Note that this halt causes the anomaly on SOH.

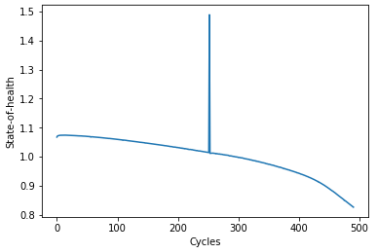


(b) Integrating current curve in discharge the discharge capacity is obtained. Thus, when a halt is occurred, the resulting capacity is miscalculated.

Figure 2.13: Irregularities on SOH due to unexpected halt.



(a) Gaps in cycle *252* of *b2c12* battery, thus, different charging strategy seems to be used.



(b) The consequence of the having gaps during the test of *b2c12* cell. In theory, less time discharging means less discharge capacity.

Figure 2.14: Illustration of the relevance of gaps on the evolution of the SOH.

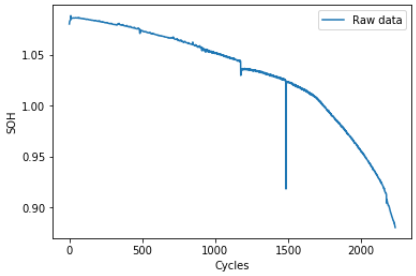
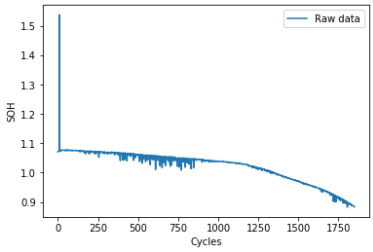


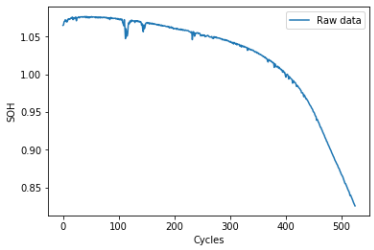
Figure 2.15: The consequence of restarting the computer at *b1c2* cell. Here *3.6C(80%)-3.6C* policy is used and finally 2237 cycles were measured.

- In the same way, also in the second batch, the computer also restarted near the end of *b1c2* cell, Figure 2.15.

- Moreover, in the second batch, upon unloading the cells, the thermocouples was noticed from channels 7 and 21 had fallen off the cell. The consequences of this issue are introduced in the Figure 2.16.



(a) Apart from the big peak, some irregularities are illustrated on the evolution of SOH.



(b) Deviations of the SOH due to thermocouples, note that only in these two test happens these degradations.

Figure 2.16: Outcome of the thermocouples on the state of health.

- Also in the second batch, channel 10 dies quite quickly. Therefore, this cell might be defective, Figure 2.17.

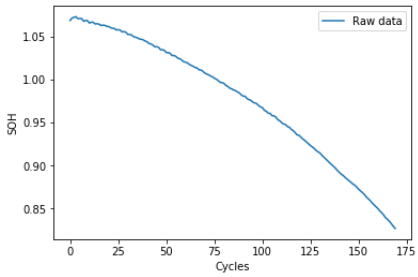
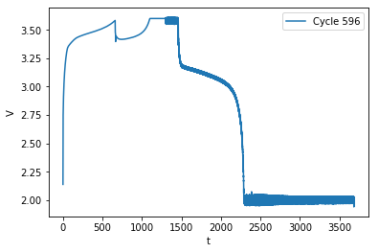
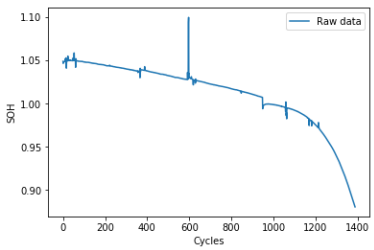


Figure 2.17: It might be considered as defective because it uses a *2C(10%)-6C* policy and it gets only 148. Whereas with a similar policy, *1C(4%)-6C*, is measured 300 cycles.



(a) An example of a noisy voltage cycle. Note that as it was introduced on Figure 2.5 the evolution of the voltage must be a smooth line.



(b) *b3c37* experiment has a huge deviations during the whole test. Note that the big peak is not derived by the noise.

Figure 2.18: Illustration of a noisy voltage and its outcome.

- Moving to the third batch, the cell in channel 46 has noisy voltage profiles, likely due to an electronic connection error, Figure 2.18.
- Finally, also in the third batch, some cells had *OCV* errors (caused by the internal resistance test) that lead to temporary pauses in cycling.

CHAPTER 3

Data preprocessing

In recent years, large amount of data over time is stored in diverse areas of the society due to the recent advances in technology. These data observations are frequently correlated in time, and constitute a time data series. Under this type of data, time series data mining aims to extract all meaningful knowledge from this data, and several mining tasks have been considered in the literature [2]. Among these tasks, the forecasting of real processes on a future horizon can present a number of difficulties arising from the nature of the low quality data and the objective itself. In fact, the data coming from a real system, can present anomalies such as, noise, missing values or useless variables that do not provide useful information.

It is important to note that low quality data have a direct influence on the quality of the predictions, resulting in an increase in this influence when the predictions are made in distant horizons. Without proper treatment of these problems the predictions will be considered to be of insufficient quality and even useless [13]. Thus, the generation of high-quality synthetic information for such inputs from real-world historical time-series through data preprocessing technique is vital to obtain improved accuracy in forecasting. The data preprocessing is followed by input selection to reduce the prediction uncertainty.

The time series of these inputs are often vulnerable to outliers causes several challenges to preprocessing process. Outlier detection has been studied in a variety of application domains such as credit card fraud detection or fault diagnosis in industry. In the first study on this topic [14], two types of outliers in univariate time series were defined: those that affect a single observation, and those affects both, particular and subsequent observations.

Since that study, many definitions of the term *outlier* and numerous detection methods have been proposed in the literature. However, to this day, there is still no consensus on the term. Despite this, from a classical point of view, [15], an outlier is "*an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*" Thus, outliers can be thought as observations that do not follow the expected behavior. This way, in this study all the irregular or unexpected points will be considered as outliers.

These irregularities can have two different meanings, and the semantic distinction between them is mainly based on the interest of the analyst or the particular scenario

considered. Most of them are unwanted data related to noise or erroneous measurements, which should be deleted or improve the data quality and generate a cleaner dataset. However, although not following usual behaviour, there are some outliers that explain real and interesting phenomena. These intriguing observations are often referred as anomalies [16].

Moreover, especially in the area of time series data, many researchers have aimed to detect and analyze unusual but interesting phenomena. Fraud detection is an example of this because the main objective is to detect and analyze the outlier itself.

The process of detection of these data points is known as outlier detection or anomaly detection. If one of these processes is followed by outlier correction, it is called the preprocessing of data [17].

This way, outlier detection techniques in time series data vary depending on the input data type, the outlier type, and the nature of the method. Therefore, Figure 3.1 introduces an overview of the resulting taxonomy.

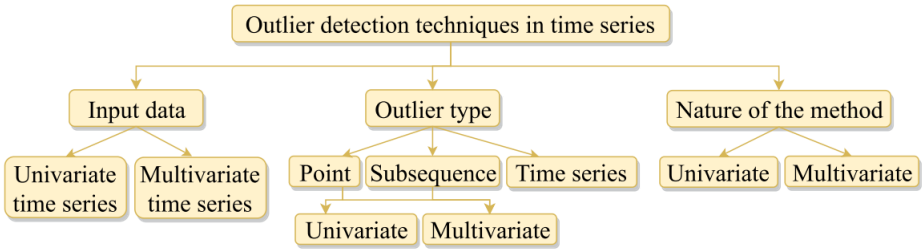


Figure 3.1: Proposed taxonomy of outlier detection techniques in time series data [2].

While input and output data refers to the search space and the type of data detected within. Nature of the method describes to the essence of the detection method employed, univariate or multivariate. A univariate detection method only considers a single time-dependent variable, whereas a multivariate detection method is able to simultaneously work with more than one time-dependent variable.

Regarding *Severson* data preprocessing, because of the relationship among all the features, it is vital to correct all connected features to each outlier. Thus, in this work, first outlier cycles are going to be located using the evolution of the discharge capacity, as regular values have a smooth progression. Then, these cycles will be corrected on all the features due to their internal connections.

3.1 Outlier detection

As other data mining tasks, each data type has different preprocessing methods due to their characteristics. Thus, there are also preprocessing methods that apply to data that are not time-series. The existing outlier detection and correction methods for a univariate time series as input and points as output can be categorized into four major types, defined as under:

- *Statistical-based methods*: In these type of methods, the data points are modeled using a stochastic distribution and outliers are detected based on the relationship with

the distribution model. Here, the outliers are those points that do not agree with or conform to the underlying model for the data.

  - *Parametric methods*: These methods assume the time-series to follow a specific parametric distribution model.
  - *Non-parametric methods*: On the contrary, these methods do not assume a specific parametric distribution. But, they are not suited for multivariate nor large datasets as intensive computations are involved.

- *Density-based methods*: The density-based outlier detection methods work on the core principle that an irregularity can be found in a low-density region. Thus, non-outliers are assumed to appear in dense neighborhoods. Despite this, they are not applicable to univariate data and there is not any approach to correct outliers.
- *Distance-based methods*: These methods detect outliers by computation of the distances between points, where the data points far from their nearest neighbor are considered outliers. Similarly, they are incapable of correcting outliers and they are not suited to apply on multivariate nor large series.
- *Cluster-based methods*: Clustering-based techniques usually rely on the usage of clustering methods to describe data behaviour. However, these methods are not applicable to univariate data nor capable for correcting outliers.

For example, *k-nearest neighbor (kNN)-based approach* is one of the most widely used distance-based methods, because its ease application to various kinds of data without knowing the distribution or characteristics of the data. The method segregates the entire time-series into several groups or windows [18].

Other methods are based on the definition for the concept of anomaly, where it is a point that significantly deviates from its expected value. Therefore, given a univariate time series, a point in time t can be declared an outlier if the distance to its expected value is higher than a threshold γ:

$$|x_t - \hat{x}_t| > \gamma \tag{3.1}$$

where x_t is the observed data and $\hat{x}_t$ is its expected value.

The methods based on the strategy described in Equation 3.1 are denominated *model-based* methods and the technique to approach $\hat{x}_t$ *estimation models*. The most simple models are based on constant models, where basic statistics such as the median or the Median Absolute Deviation (MAD) are used to obtain $\hat{x}_t$.

Other estimation-based techniques intend to identify data points that are unlikely if a certain fitted model or distribution is assumed to have generated the data. For instance, some authors model the structure of the data using smoothing methods such as *B-spline smoothing-based approach*. This method is based on the modelling of inherent patterns of a time-series to detect the presence of outliers, [19]. The data can be modeled as

$$X = m(t) + \epsilon \tag{3.2}$$

where $m(t)$ is the underlying function and ϵ is the error term. The most important task is to find a suitable estimate of the function $m(t)$ using the collected data.

Some other univariate outlier detection methods analyze all the residuals obtained from different models to identify the outliers. For instance, ARIMA models, linear regression or Artificial Neural Netowrks (ANNs).

In contrast, statistical-based *sliding window prediction-based approach* will be used in this study. It is a prediction-based method which uses the nearest neighborhood of data, [20]. Considering a time-series X, the $k^{'th}$ nearest neighborhood $n_i^{k'}$ of width $2k'$ of a data point x_i at a time instant i is defined as,

$$n_i^{k'} = \{x_{i-2k'}, x_{i-2k'+1}, ..., x_{i-1}\} \tag{3.3}$$

where $n_i^{k'}$ is one-sided window which considers $2k'$ data points before x_i.

A both-sided window which considers *k'* data points before x_i and *k'* data points after x_i is given as,

$$n_i^{k'} = \{x_{i-k'}, x_{i-k'+1}, ..., x_{i-1}, x_{i+1}, x_{i+2}, ..., x_{i+k'}\} \tag{3.4}$$

In general, one-sided window is preferred over both-sided window in case of outlier detection as *k'* data points after x_i may have undetected and uncorrected outliers. The predicted value of x_i i.e., x_i', based upon one-sided nearest neighborhood $n_i^{k'}$ is defined as

$$x_i' = \frac{\sum_{j=1}^{2k'} w_{i-j} x_{i-j}}{\sum_{j=1}^{2k'} w_{i-j}} \tag{3.5}$$

where w_{i-j} is the weight of data point x_{i-j}, which is inversely proportional to the distance between points x_i and x_{i-j}, which is given by

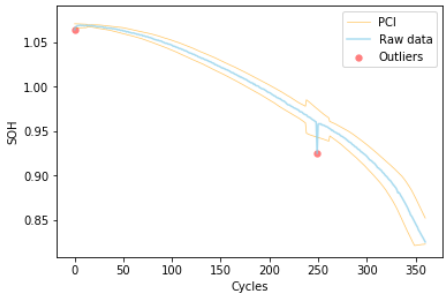
$$w_{i-j} = \frac{1}{|x_i - x_{i-j}|} \tag{3.6}$$

The greater the distance between the data points, the smaller is the weight indicating that the dependence of x_i on x_{i-j} is insignificant and vice versa. Based on the prediction on Equation 3.5, predicted confidence interval (PCI) is calculated for x_i.

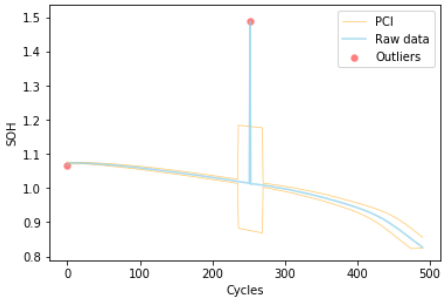
Algorithm 1 Sliding window prediction-based outlier detection

1: **for** $i = 2k' + 1$ until $i = n$ **do**
2: Compute the $k^{'th}$ nearest neighborhood n_i^k
3: Compute the prediction value
4: Compute the PCI for x_i
5: **if** x_i is not within the PCI **then**
6: Replace the data point x_i with its predicted value $\hat{x}_t$.
7: **end if**
8: **end for**

If x_i does not lie within PCI, it is treated as an outlier. The corrupted data point x_i is replaced by the corrected data, which is the predicted value x_i'. This detection and correction process continues until the last data point of the time-series is checked. The flowchart for preprocessing using SWP is as given in Algorithm 1. Figure 3.2 illustrates the methodology. Although a correction is possible, in this occasion, it will be used only to detect rare cycles.



(a) The evolution of the discharge capacity during a complete test. Note that SWP precisely detect the outlier at cylce 249 of *b2c3* cell.

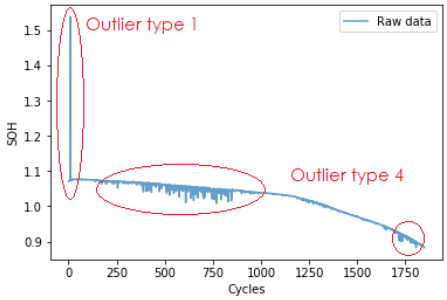


(b) SOH of *b2c12* experiment after and before applying preprocessing. Highlight that the PCI of the whole time series is computed before the correction.

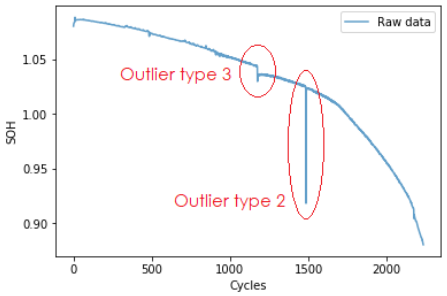
Figure 3.2: Performance of Sliding Prediction on State-of-Health.

3.1.1 Real Outliers

In order to evaluate the detection algorithm and taking into account that there is no information about whether such points are outliers or not in reality, the only way to classify whether a data point is a true outlier or not is to check the near by data points and see whether it follows similar characteristics. Thus, experiments by experiments, 109 outliers on 61 different time series were detected by an expert on battery life evolution. As a result, 56% of the experiments, in the observed data, suffer from at least one unexpected value in the series. Moreover, 15 of them (24%) have more than one outlier.



(a) Illustration of outliers type 1 and 4 on *b1c0* test.



(b) An example of outliers type 2 and 3 on *b1c2* experiment.

Figure 3.3: Performance of Sliding Prediction on State-of-Health.

These outliers can be organized according to their behaviour in four groups, clarified on Figure 3.3. First and second types are those with significant peak, whereas, the third group

consists of small drops on the discharge capacity and, finally, fourth collection introduce other small irregularities.

Regarding first and second groups, 20 outliers are classified as it is described on Table 3.1. These two categories are caused by the unexpected restart of the computer and by gaps in the data as introduced in Figures 2.13 and 2.14.

Table 3.1: The exact experiments and cycles are described for those 20 outliers, where those 5 first group experiments are represented in bold.

Experiment	Cycle	Experiment	Cycle	Experiment	Cycle	Experiment	Cycle
b1c0	10	b1c2	1485	b1c4	1125	b1c5	907
b1c18	38	b2c3	249	b2c4	246	b2c6	257
b2c10	250	b2c11	249	**b2c12**	252	b2c13	245
b2c20	248	b2c22	246	b2c28	249	b2c29	246
b2c37	246	b2c42	246	**b2c44**	247	**b3c37**	596

Thirdly, 7 small peaks were detected as outliers, presented on Table 3.2. Although they show less irregularity on the behaviour of the time series and the inability to recover the previous capacity, these cycles are also taken as outliers. As an example, one of these was illustrated on the Figure 2.15 on the 1175 cycle, before the huge drop at cycle number 1485.

Table 3.2: Seven outliers from the third group are detailed.

Experiment	Cycle	Experiment	Cycle	Experiment	Cycle	Experiment	Cycle
b1c1	1177	b1c2	1175	b1c4	1125	b3c6	550
b3c24	295	b3c35	356	b3c37	950		

In regard to the fourth outlier type, the Table 3.3 describes smaller deviation during the experiments. In total there are 36 outliers that should be corrected.

Table 3.3: More 36 irregularities are introduced. Note that *b1c0* experiments there is a range of cycles, that are taken into account as one.

Experiment	Cycle	Experiment	Cycle	Experiment	Cycle	Experiment	Cycle
b1c0	250-900	b1c0	1715	b1c0	1725	b1c0	1735
b1c1	485	b1c2	485	b1c3	500	b1c7	569
b1c8	491	b1c9	886	b1c20	12	b1c21	12
b2c0	72	b2c17	112	b2c17	144	b2c17	232
b2c28	104	b2c28	125	b2c30	247	b2c31	250
b2c32	357	b2c33	247	b2c35	247	b2c39	247
b2c40	64	b2c40	338	b2c40	358	b2c47	257
b3c37	16	b3c37	53	b3c37	60	b3c37	366
b3c37	1058	b3c37	1059	b3c37	1062	b3c37	1063

Finally, it seems to have been a data collection issue in some of the experiments, due to this 46 experiments have their first cycle, in all features, set to 0. Thus, the outlier detection

method should equally take into account. As a result, the expert of battery life evolution has detected 109 cycles with this type of outliers.

3.1.2 Evaluation

The preprocessing methods existing in the literature differ from each other in their preprocessing capabilities. Therefore, a comparison and the basis for the analysis must be settled down. To do so, outliers are introduced manually, taking into account that there are only 135 experiments, and then the method applied to the data, and its performance is analyzed. Four indicators are used as defined as:

- **True positive (TP)** : The data points identified correctly as outliers by the method.
- **False positive (FP)** : The genuine data points marked as outliers by the algorithm.
- **False negative (FN)** : The outliers that are not identified by the method.
- **True negative (TN)** : The data points correctly identified as non outliers by the method.

Based on these indicators, three bases for comparison are defined as follows:

- **Precision (P)** : It is the percentage of outliers detected correctly, out of the total number of data points marked as outliers by the algorithm. The higher the precision a the better is the performance of the method.

$$Precision = \frac{TP}{TP + FP} \tag{3.7}$$

- **Recall (R)** : It is the percentage of outliers detected correctly by the method with respect to the total number of outliers present in the dataset. The method with the highest recall value is considered the best.

$$Recall = \frac{TP}{TP + FN} \tag{3.8}$$

- **F-measure (F)** : Is the harmonic mean of precision and recall. Its significance lies in the comparison of methods when two methods have contradictory P and R values. In this case, the method with higher F-measure is considered better.

$$\text{F-measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.9}$$

In this work a simplified version of the *sliding window prediction-based method* is evaluated, which has two hyperparameters to tune: one is the size of the both-sided window and the predicted confidence interval calculated as:

$$PCI = x'_i \pm (\gamma * std) \tag{3.10}$$

where x'_i is the predicted value and *std* means the standard deviation of the window. This way, there are two tuning parameters, the length of the window and the γ value.

- The window is used to estimate the real value. The larger is the window, the more different values will be taken into account, thus, rarer outlier will be detected. This means, that less outleirs will be tracked. It is studied that a very small percentage of total time series must be taken into consideration, in this project, in total 3.5% will be used.

- γ value: It is also used to make the model more or less sensitive. The greater is the value, the less outliers will be detected.

The model was tuned taking into account previously considered as real outliers. This way, Table 3.4 introduce three different models, with three γ value as the window length was already set. In addition, Figure 3.4 introduce the performance of each tuned model.

The table confirms the relevance of γ on model's performance. Despite it could be believed that detecting and correcting more than one thousand cycles might be dangerous, in order to preserve the reality of the database, in practice, as the correction will not change the value of false positive points. Therefore, the model with the biggest *F measure* will be chosen, in this case, with the γ tuning variable set to 1.5

Table 3.4: Evaluation of SWP model with different γ value.

γ	TP	FN	FP	Precision	Recall	F measure
1.5	**108**	**1**	**1082**	**0.091**	**0.991**	**24.056**
1.9	102	7	429	0.199	0.935	12.15
2.5	89	20	208	0.299	0.816	9.123

Figure 3.4 is divided in rows and columns, in a matrix style. Each column represents a different experiment, *b1c32*, *b2c40* and *b3c37* respectively. Whereas each row introduces a γ tuning value, where the first row illustrates only raw values and second, third and fourth rows shows the performance SWP with γ value equal to 1.5, 1.9 and 2.5.

In the view of the results, due to the sensitivity of the model, increasing the γ value, decreases the number of detected outliers. Therefore, false positive outlier are also decreased, while the chances to get false negatives increases.

To sum up, it is decided to use the *Sliding window prediction-based* method to detect outlier cycles during the experiments, taking into account 3.5% of the time series, as window, to predict the value with the predicted confidence interval (PCI) set as $1.5 * std$. As the correction method will not modify much those false positive values.

3.2 Outlier correction

As it was mentioned before, each outlier detected with sliding window corresponds to a cycle. Thus, all the data from "features of a cycle" must be corrected and on "features of an experiment" only that point.

Although missing values and outliers are different, the correction method is quite similar, to replace the missing value or the outlier with reasonable values without influencing the predictive model so much. This way, the replacement or imputation is a huge area, where

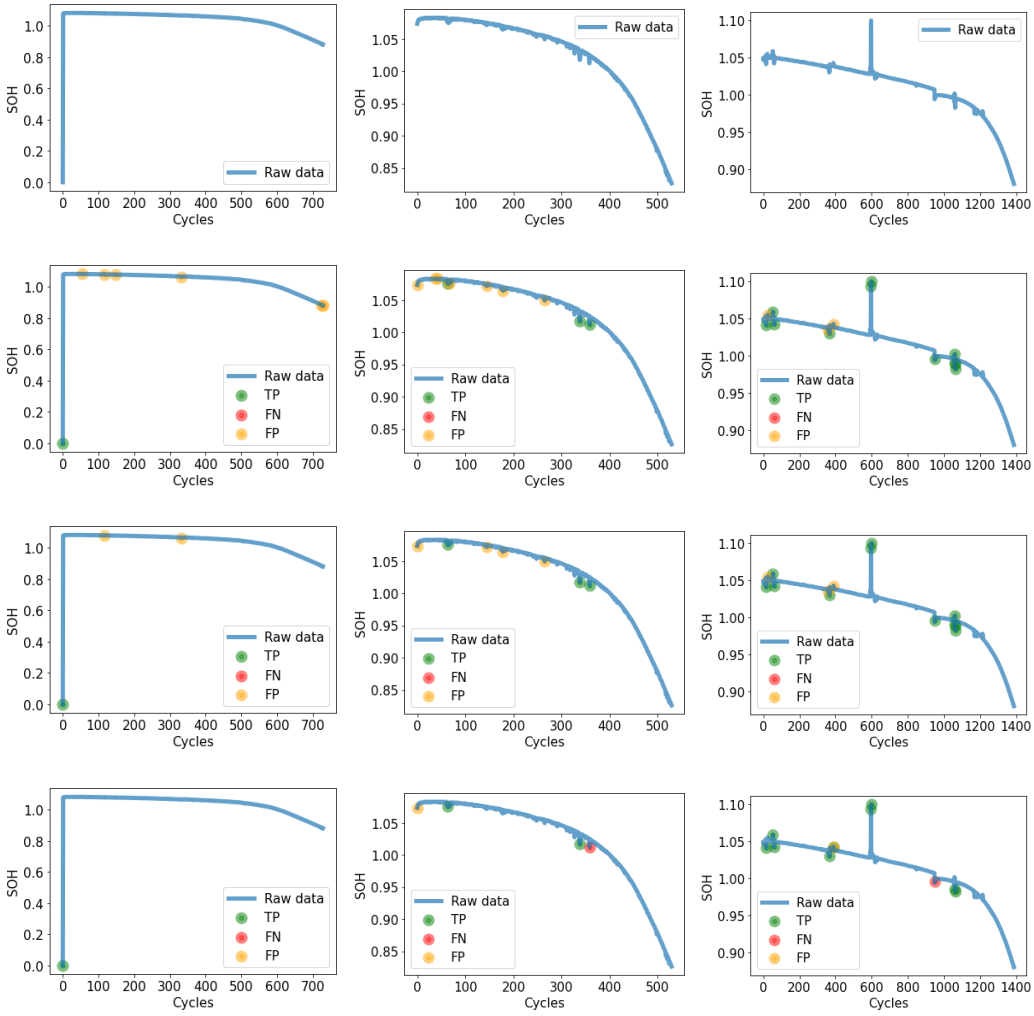


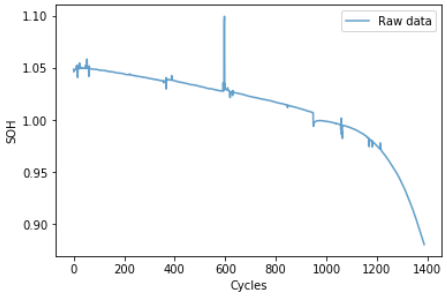
Figure 3.4: The performance of SWP with γ parameter set as 1.5, 1.9 and 2.5 (rows) in *b1c32*, *b2c40* and *b3c37* experiments (columns).

lots of research has already been done, for instance Multiple Imputation [21] or Nearest Neighbor [22] methods. In the research field of imputation, univariate time series are a special challenge. Most of the sufficiently well performing standard algorithms rely on inter-attribute correlations to estimate values. Effective univariate algorithms instead need to make use of the time series characteristics [23].

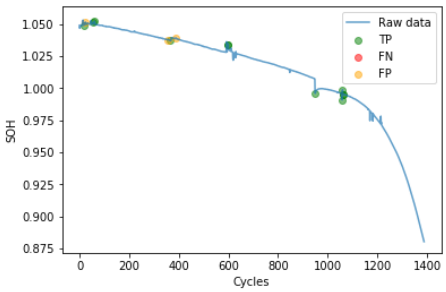
On the other hand, imputation methods fall primarily into two broad classifications: traditional and modern techniques. Traditional techniques such as simple deletion, averaging, or regression estimation are limited but still used in many cases. On the other hand, modern approaches such as multiple imputation and maximum likelihood routines, have proved superior and are gaining favour [24].

Despite of recent growth of new approaches, due to the slow evolution of the LIB features, a simple traditional methodology will be used. The irregular value or cycle, on features of a single cycle, will be replaced by the nearest regular point or repetition as it is illustrated on Figure 3.5. Although replacement strategy is considered, the mean of

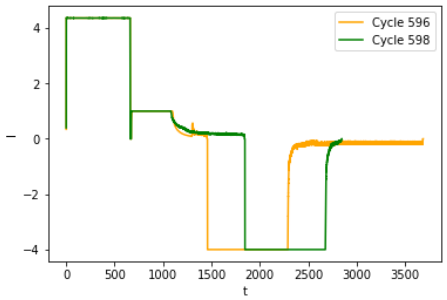
nearest neighbours also was taken into account. However, due to the slow evolution of the discharge capacity time series, minor difference has been achieved between these two strategies.



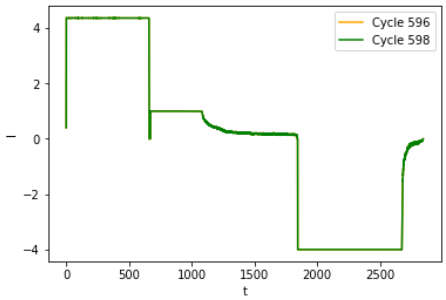
(a) The evolution of the discharge capacity of *b2c42* experiment without any preprocessing, note big irregularity on cycles 596 and 597.



(b) Corrected discharge capacity of *b2c42* test. Green points represents true positive error before the correction.



(c) The cause of the irregular peak on capacity on cycle 596, there is not any rest after charging.



(d) Irregular current cycle replaced by the next nearest regular neighbour as other cycles's features, in this case cycle number 598.

Figure 3.5: Performance of Sliding Prediction on State-of-Health.

3.3 Feature selection

In order to effectively use machine learning methods, preprocessing data is essential. Feature selection is one of the most frequent and important technique in data preprocessing, and has become an indispensable component of the machine learning process [25]. It is also known as variable selection, attribute selection, or variable subset selection in machine learning and statistics. It is the process of detecting relevant features and removing irrelevant, redundant, or noisy data. This process speeds up data mining algorithms, improves predictive accuracy, and increases comprehensibility. Irrelevant features are those that provide no useful information, and redundant features provide no more information than the currently selected features [26]. Therefore, the correct use of feature selection algorithms for selecting features improves inductive learning, either in term of generalization capacity, avoiding over-fitting, learning speed, or reducing the complexity of the induced model.

Feature selection (FS) techniques are typically classified into three groups: wrapper methods, filter methods and embedded methods [27]. The main difference between them is that wrapper and embedded methods are specific for the used classifier, while filter methods

are independent of the employed classifier. Due to the aim of this study is prediction, a filter method will be used.

In the context of multivariate time series forecasting, all of the filter methods try to find the subset of predictor time series that best improves the prediction accuracy of a target time series, in this case, being the evolution of discharge capacity. Which the selection is usually based on correlation of a dataset. Far from highly used Principal Component Analysis (PCA) [28], due to time series peculiarities other techniques must be applied, for instance model-based correlation between two time series or cross-correlation. Cross-correlation is the comparison of two different time series, while auto-correlation is the comparison of a data with itself.

In spite of different type of correlation methods, *Pearson correlation*, *Spearman correlation* and *Mutual information* will be used in feature selection in this work. Firstly, *Pearson* correlation quantifies the linear relationship between two features where a change in one feature is associated with a proportional change in the other feature. It is the covariance of two features divided by the product of their standard deviations [29]. The Pearson correlation coefficient, r_{pear}, between two resource metrics x_i an x_j is given by

$$r_{pear} = corr(x_i, x_j) = \frac{\sum_{t=1}^{T}(x_i^t - \mu_i)(x_j^t - \mu_j)}{\sqrt{\sum_{t=1}^{T}(x_i^t - \mu_i)^2}\sqrt{\sum_{t=1}^{T}(x_i^t - \mu_i)^2}} \tag{3.11}$$

where μ_i and μ_j represent the average value of the features x_i and x_j, respectively. Pearson correlation gives a value in $[-1, +1]$, where $+1$ denotes maximum positive correlation and -1 specifies negative correlation between the two variables. Note that the Pearson correlation coefficient is symmetric: $corr(x_i, x_j) = corr(x_j, x_i)$.

To select significant features from a set of features, a subset Z is redefined recursively. Initially, $Z = \{\}$. The candidate resource metric whose Pearson correlation coefficient value for the desired resource metric is greater than a threshold is selected [30].

As regards to the *Spearman* correlation, it measures the strength and direction of monotonic association between two variables which tend to change together but not necessarily at a constant rate. Let x_i and x_j be two resource metrics having T observations each. A rank of each value in the resource metrics x_i and x_j is obtained by assigning 1 to the lowest value, 2 to the next lowest and so on. The correlation coefficient is computed as:

$$r_{spear} = 1 - \frac{6\sum_{t=1}^{T}(c^t)^2}{T(T^2 - 1)} \tag{3.12}$$

where c^t is the difference in ranks of two features at time t. Spearman correlation gives a value between $+1$ and -1. If time series x_j tends to increase when time series x_i increases, the correlation coefficient is positive. If X_j tends to decrease when x_i increases, the coefficient is negative. A Spearman correlation of zero indicates that there is no tendency for X_j to either increase or decrease when x_i increases [31]. Similar to Pearson correlation coefficient, only those features greater than a threshold will be selected.

Finally, regarding *Mutual information*, oppose to the others, it is not a correlation measurement. It quantifies the similarity between to labels of the same data. Despite the data used in this study is continuous, time series, with the aim of simplify, first, it will

be explained with discrete values, for instance the similarity between no time correlated variables.

This way, mutual information between clusters U and V is give as,

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|}, \tag{3.13}$$

where $|U_i|$ is the value that instance i takes from variable U and $|V_j|$ is the the value that instance j takes from variable V. This metric is symmetric, this means that same score will get switching the order of clusters order.

According to time series, in order to apply mutual information on time sereis, it must be based on regression. This function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances as described in [32] and [33].

Regarding *Severson* data, two types of features were defined, which describes a cycle and an experiment. Taking into account that the predictive feature is the discharge capacity during an experiment, first this type of features will be selected. Therefore, *Pearson* and *Spearman* correlation are calculated in each experiment. Figure 3.6 introduces the distribution of Pearson and Spearman correlation to each pair of features, while Figure 3.7 shows the histogram of Mutual information between pair of features among all the experiments.

According to [34] and [35] studies, a strong relationship is considered when the correlation is higher than 0.7. Despite this, a strong relationship not always means a good feature to predict, as it could be very similar. This means, that the model will need more time and memory to train model that it could not be beneficial at forecasting.

Therefore, instead of selecting those features with high interest, the idea in this work will be to refuse those features with lack of relevant information. For instance, as it was expected, charging and discharging capacities have high correlation and similarity. This way, as they share almost all the data, charging capacity does not add new interesting information with respect to discharging capacity.

On the contrary, the three temperature features, broadly, has low interest. As we see, *Tmin* features has very low information with the rest of features and a great number of experiments has near to zero correlation. This way, it seems that to be a noisy feature. Moreover, *Tmax* and *Tavg* have similar behaviours, therefore, these two features will not taken on the feature selection.

In summary, the selected experiments' features are discharge capacity (QD) and internal resistance (IR), and they will be used to learn the models and predict the RUL of lithium-ion batteries. Despite feature selection, the model will be trained with different input features in order to study the relevance of the input information.

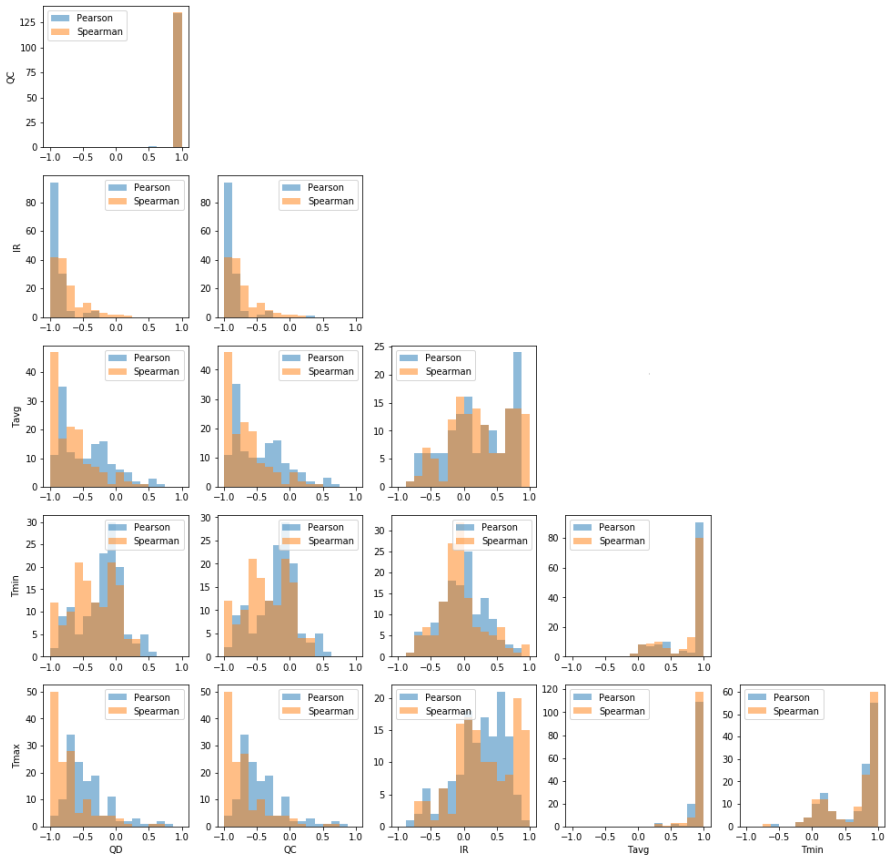


Figure 3.6: Distribution of Pearson and Spearman correlations between pair of features.

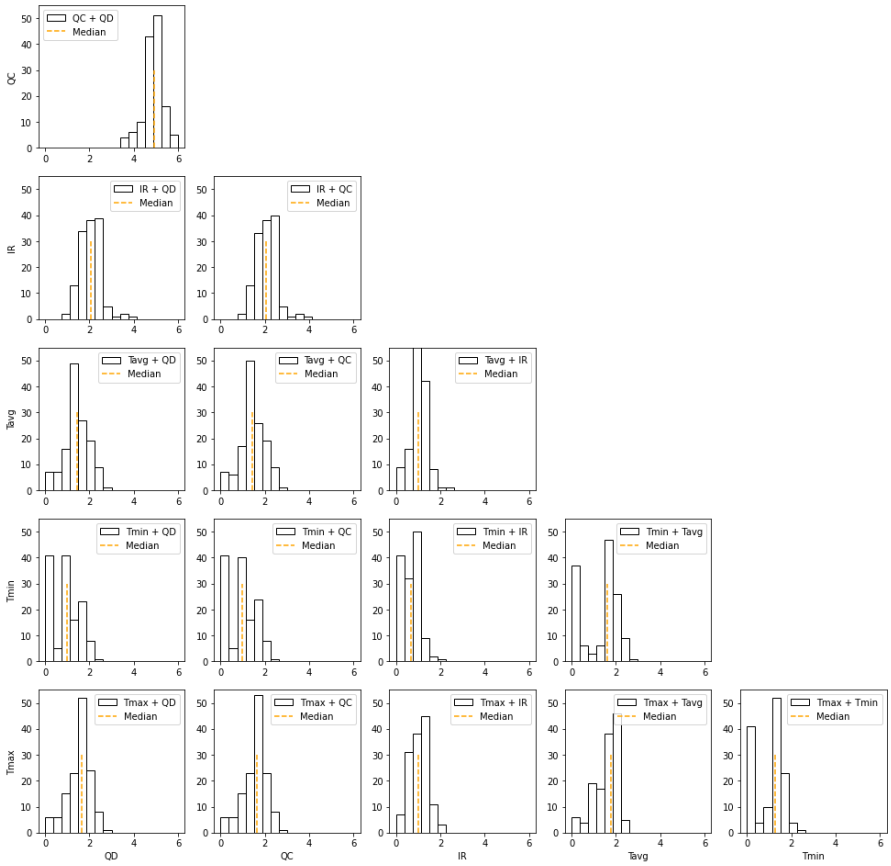


Figure 3.7: Distribution of mutual information between pair of features and among all the experiments.

CHAPTER 4

Models

In this chapter, the introduction of the state-of-the-art, complemented by the learning scenario is provided. In addition, the selected algorithm will be present as well as the proposed learning process and model.

4.1 Learning scenario

Learning scenario of an machine learning task highly depend on the data available and the internal characteristics of the problem. Therefore, the learning environment of lithium-ion battery remaining useful life prediction using [12] is divided in these properties:

- Uni-step forecasting: In each iteration of the model, an instance will be forecast.
- Early prediction : Only first 100 cycles of an experiment will be used to forecast the rest of the evolution of the discharge capacity, until it reaches 80%.
- Multiple time series : Due to lack of past information on predictive time series, the rest of experiments will be used to train the model.
- Second dimension : As optional and additional information, each instance, cycle, of the capacity time series has another dimension with multivariate time series.

4.2 State-of-the-art review

Time series forecasting is the generalization of battery life prediction. This way, fundamentally, the goal of time series prediction is to estimate some future value based on current and past data samples. Mathematically stated as,

$$\hat{x}(t + \Delta_t) = f(x(t - a), x(t - b), x(t - c), ...), \tag{4.1}$$

where, in this example, $\hat{x}$ is the predicted value of a (one dimensional) discrete time series x.

This way, the objective of time series prediction is to find a function $f(x)$ such that $\hat{x}(t)$, the predicted value of the time series at a future point in time, is *unbiased* and *consistent*.

The challenge in *Remaining useful life (RUL)* forecasting of lithium-ion batteries lies in how to store and update the key information from the degradation data via effective learning of long-term dependencies. In this sense, RUL prediction methods for batteries can be classified into two main categories: model-based and data driven methods [36]. (This is an extension of the state-of-the-art presented in the introduction).

Model-based or physics-based methods are approaches that involve the knowledge of a system's failure mechanism (e.g. crack growth) to build a mathematical description of the system's downgrading process in order to estimate the RUL. The numerical models quantitatively characterize a system's behaviour using the first principles. Therefore, model-based methods use mathematical models to capture the long-term dependencies of battery degradations, which are combined with advanced filtering techniques. For instance, this filtering process, *particle filter* (PF) algorithm, can be used to update the model parameters by evaluating the importance of the data points adaptively based on the measured signals.

He et al. [37] used the Dempster-Shafer theory (DST) and PF methods to predict the RUL of batteries. DST is used to initialize model parameters, and PF is then used to update the system parameters and forecast the RUL based on the available data through battery capacity monitoring. To further improve modeling accuracy, *Xing et al.* [38] developed an ensemble model to characterize the capacity degradation and predict the RUL batteries. Due to lack of knowledge of the degradation evolution, the highly complex chemical reactions inside the lithium-ion battery (LIB), the mathematical model is usually constructed by fitting the degraded capacities of LIB, but this step could lead to overfitting. This means that, the approach is able to fit the training data precisely but its prediction accuracy is low. In addition, the state of lithium-ion battery is highly vulnerable to working temperature, circuit, load and other environmental factors, therefore it is difficult to establish an accurate model for lithium-ion battery degradation [39].

The data-driven method has recently drawn significant interest in lithium-ion battery RUL prediction research area. Yet most data-driven methods are concentrated on making short-time forecasting for same battery based on its historical data , few existing models can realize multi-battery prediction by using data from multiple batteries. The advances in *Artificial Intelligence* (AI) and Deep Learning introduce new data-driven approaches to this issue. Especially *Deep Neural Network* (DNN) are suitable for high complex non-linear fitting by training multi-layer artificial neural networks, and can achieve better accuracy for complex prediction problems such as multi-battery RUL estimations [40].

Specifically, the data-driven approach does not require specific knowledge of the issue. Statistical techniques and/or machine learning algorithms are used to observe parameter changes, isolate faults and estimate RUL for failure diagnosis prediction. In other words, data-driven methods rely solely on historical data rather than requiring precise mathematical models of battery degeneracy or internal mechanisms of the battery. Thus, these techniques are flexible and be applied to problems with similar format, even when the underlying physics are different.

Hu et al. [41] proposed a data-driven forecasting system for by using a combination of sample entropy and sparse Bayesian predictive modeling. *Hong et al* [42] presented a novel performance degradation assessment method for bearing based on ensemble empirical

mode decomposition (EEMD) and Gaussian mixture model (GMM).

Sasha et al. [43] and *Zhou et al.* [44] uses inner parameters of the storage to build a Relevance Vector Machine (RVM) model and uses Particle filter(PF) algorithm to determine the adaptive variables of the RVM model to predict the decline of the lithium battery. However, the long-term forecasting ability of the RVM algorithm is poor, so it is difficult to obtain decent RUL estimation outcome by directly using the RVM model.

Moving on, the kernel regression approach models the non-linear relationship between the measurable features (e.g., cell terminal voltage, current and temperature) and the cell capacity by means of kernel functions. Kernel regression techniques that were employed to evaluate the capacity of Li-ion battery include support vector machine (SVM) and relevance vector machine (RVM). For instance, *Nuhic et al.* introduced SVM to forecast lithium-ion battery's state of health (SoH) and RUL [45], and *Liu et al.* proposed RVM algorithm, an online training method, to improve the accuracy of RUL prediction [46]. *Patil et al.* have desired an online multistage SVM method utilizing battery voltage and temperature data as characteristic parameters to improve the precision of RUL prediction [47]. Nevertheless, these algorithms require a large amount of past data to learn the patterns of cell capacity degradation, and in case battery degeneration data are deeply nonlinear, forecasting accuracy may not be guaranteed.

To overcome these limitations, neural network (NN) techniques have been recently employed to predict the lithium-ion battery (LIB) RUL. The NN approaches basically builds a web structure of interconnected "neurons" to model the dependency between the measurable features and the cell capacity. For example, *Liu et al.* introduced the adaptive recurrent neural network (RNN) for predicting RUL by estimating the dynamic state of cell [48]. Using NASA's battery data on lithium-ion degradation [49], the authors verified that RNN works more precisely than RVM and PF techniques. However, RNN has the long-term dependency problem as the number of cycles increases [50]. Very recently, *Zhang et al.* [51] used long short-term memory (LSTM) to reflect long-term memories of battery degradation tendency, as it is studied that this NN can maintain and update timing information, thereby achieving excellent performance in long-term time series prediction.

Long short-term memory (LSTM) is a type of *Recurrent Neuronal Network (RNN)* which is a subdivision of *Artificial Neural Network (ANN)* which are widely applied to model sequential data like time series or natural language. Moreover, deep learning models have been proposed for many other applications such as: image, speech, video, and audio reconstruction, natural language understanding, sentiment analysis, question answering, and language translation [3].

Even so, these LSTM predictions did not capture the *capacity regeneration*, the irregular capacity increase, which cannot be well predicted by using capacity data only. In addition, LSTM basically requires a large number of parameters for training. To overcome the shortcoming of LSTM with a large number of training variables, *Xiao et al.* [52] used gated recurrent unit (GRU) to estimate the state of charge, and *Song et al.* [53] used GRU to predict the battery RUL.

In order to significantly improve the forecasting precision even in the presence of capacity regeneration, *Park et al.* [54] proposed two learning methods using LSTM. They leverage the multichannel measurable data of voltage, current and temperature charging profiles from Battery Management System (BMS) whose patterns vary with cycles as aging.

In order to use this various data efficiently, unlike the traditional LSTM prediction that matches input layer with output layer as one-to-one structure (the same size of capacity vectors for input and output), they match, also, input layer with output layer as many-to-one structure.

Although the massive use of machine learning and deep learning approaches, *Hu et al.* [55] proposed a genetic algorithm (GA) to develop a simple, but an accurate model. This model is composed by a k-nearest neighbor (kNN) regression to build a non-linear kernel regression model, with the aim to capture the complex dependency of the capacity on the features. In addition to this, a particle swarm optimization (PSO) is adapted to finding the optimal combination of feature weights for creating a kNN regression model that minimizes the cross validation (CV) error in the capacity estimation.

In addition to the previous, hybrid approaches that combine two or more model-based or data-driven approaches has been also proposed [56]. Particularly, this type of schemes try to overcome the limitations of a single algorithm which results in many cases to an improvement of the prediction performance. Generally, there are two kinds of hybrid systems, one combines model-based and data-driven, while the other combines different kinds of data-driven approaches.

The first type of hybrid approach can be divided into the following two categories:

1. Data-driven predictors are employed to provide the predicted information for the models. Because there are no measurements in the prediction phase, the predicted values from the data-driven approach can be used by the state observer to update model parameters in the prediction phase. Liu et al. [57] use a data-driven approach to predict future measurements, and these predicted measurements are added to the PF to update model parameters.

2. A data-driven approach is proposed to overcome the limitations of filtering methods. The selection of the initial value of the parameters affects the convergence of the filtering algorithm. Therefore, He et al. [58] initialize the model parameters based on *Dempster–Shafer theory*, update the model parameters using the PF filtering algorithm based on the existing capacity degradation data, and achieve the RUL prediction.

The second type of hybrid approach combines two or more types of data-driven approaches. Hu et al. [59] combine multiple data-driven algorithms based on a weighted-sum formulation and the weighting scheme determines the weights of each algorithm. Because ensemble learning is still challenging for the fusion of different data-driven approaches, here we mainly focus on the first type of hybrid approach.

Note that the model-based approach needs complex mechanism modeling, and it is difficult to build a high-accuracy model to complex electrochemical systems like batteries. The data-driven approach avoids the modeling process, and its prediction accuracy depends on historical observation data. However, this approach needs a large amount of data, thus inducing high computational complexity. Overall, developing an effective hybrid approach, which combines the benefits of both approaches to improve accuracy and confidence might be very interesting.

4.3 Selected Algorithm

The *Long Short-Term Memory (LSTM)* neural network is selected to study the behaviour of different features and hyperparameter combinations, due to the recent LSTM outperforming investigations [60]. This election is not only motivated by large number of algorithms that started to apply RNNs, such as LSTMs, to time series forecasting, but also, it is as far as we have investigated, the only one that permits to incorporate the all the data that has been considered in the present learning scenario.

Despite this severe decision, other classical models were also taken into account. Given the historical impact of *ARIMA* models in time series forecasting [61][62], *Vector Autoregression Moving Average regressor (VARMA)* model was also considered [63]. In addition, as a result of studies of remaining useful life (RUL) of lithium-ion batteries (LIB) in last few years using *Support Vector Regressor* (SVR) was also considered [64].

However, due to newborn area of using multiple time series to forecast, limitations were found on open source machine learning libraries, *statsmodels* for ARIMA models and *scikit* for SVR, to train with more than one dataset, as in this case each experiment must be considered as different dataset or time series. In spite of this restriction, it seems that the constraints come from repositories and not from models as there are signs of optimism after reading the use of multiple time series as input of SVR model, although it is not a trivial task [65]. As a result of this restriction and the aim of studying features importance of LIB RUL prediction, a decision of using LSTM models was taken.

4.3.1 MLP

Multi Layer Perceptron (MLP) is the simplest model of ANN. Even though particular model architectures might have variations depending on different problem requirements, MLP models consist mainly of three layers: input, hidden and output, Figure 4.1. Thanks to this simplicity, it could be easily used in wide range of learning scenarios.

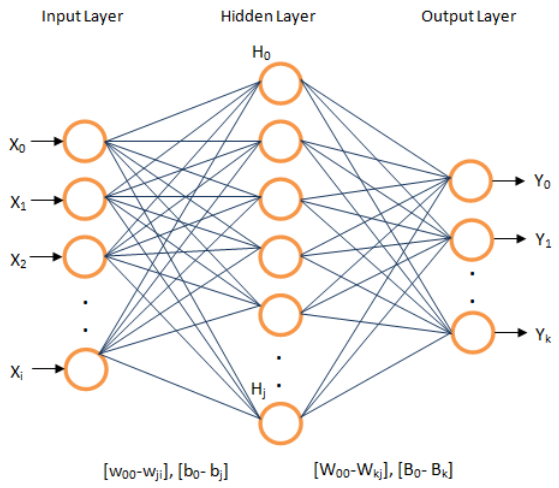


Figure 4.1: Topology of a Multilayer Perceptron Neural Network.

The number of neurons on each layer and the number of layers are one of many

hyperparameters of the network. In general, each neuron in the hidden layers has *input (x)*, *weight (w)*, and *bias (b)* terms. In addition, each neuron has a nonlinear activation function, which produces a cumulative output of the preceding neurons, as introduced in Equation 4.2.

$$y_i = \sigma\Big(\sum_i w_i x_i + b_i\Big) \tag{4.2}$$

where y_i is the output of the neuron and σ represents the activation function. The most common used nonlinear activation functions are *sigmoid*, *hyperbolic tangent*, *Rectified Linear Unit*, *leaky-ReLU*, *swish* and *softmax* [66].

The MLP, as other NN models, learning stage is implemented using *backpropagation* technique. It is based on the idea of propagating the learning errors of output layers back to the preceding layers. Optimization algorithms are used to find the optimum parameters/variables of the NNs. After that, they are used to update the weights of the connections between between the layers. Different optimization algorithms have been developed: Stochastic Gradient Descent (SGD), SGD with Momentum, Adaptive Gradient Algorithm (AdaGrad), Root Mean Square Propagation (RMSProp), and Adaptive Moment Estimation (ADAM) [67] [68] [69].

Gradient descent is an iterative method to find optimum parameters of the function that minimizes the cost function. For instance, SGD is an algorithm that randomly selects a few samples for each iteration instead of the whole dataset. While ADAM is an updated version of RMSProp that uses running averages of both the gradients and second moments of the gradients. ADAM combines the advantages of RMSProp (works well in online and non-stationary settings) and AdaGrad (works well with sparse gradients) [69].

As it is introduced in the Figure 4.2, the effect of backpropagation is transferred to the previous layers [70]. Using the chain rule, layers that are deeper into the network go through continuous matrix multiplication in order to compute their derivatives.

In a network of n hidden layers, n derivatives will be multiplied together. If the derivatives are large then the gradient will increase exponentially as they propagate down the model until they eventually explode, called as *exploding gradient problem*. Similarly, if the derivatives are small then the gradient will decrease exponentially, until it eventually vanishes, named as *vanishing gradient problem*.

In the case of exploding gradients, the accumulation of large derivatives results in the model being very unstable and incapable of effective learning. Large changes in the models weights creates a very unstable network, which at extreme values the weights become so large that is causes overflow resulting in *NaN* weight values of which can no longer be updated. However, the accumulation of small gradients results in a model that is incapable of learning meaningful insights since the weights and biases of the initial layers, which tends to learn the core features from the input data, will not be updated effectively. In the worst scenario the gradient will be 0 which in turn will stop the network will stop further training.

As other machine learning model, the important issue in the MLP are the hyperparameters of the networks, which are the variables of the network that affect the network architecture and performance of the networks. The number of hidden layers, number of units in each layer, regularization techniques, network weight initialization, activation

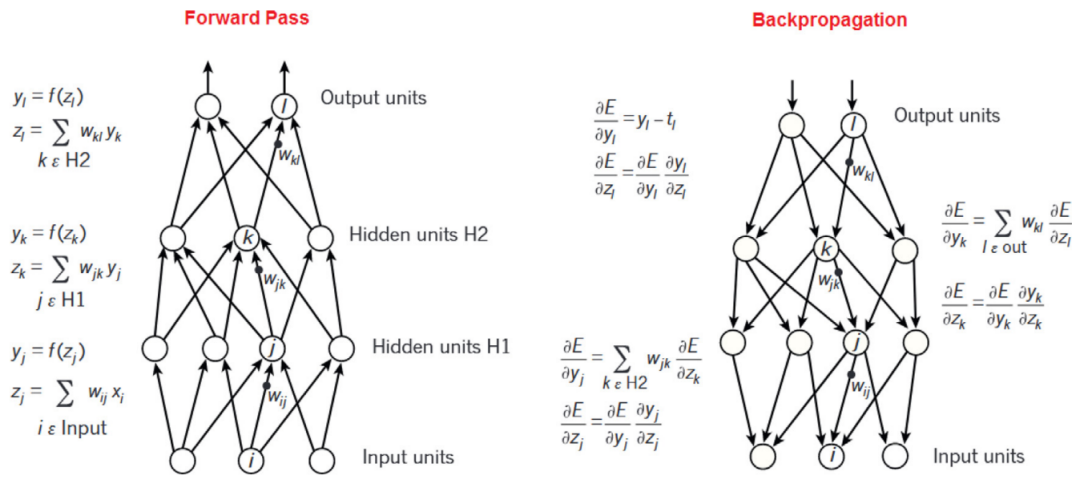


Figure 4.2: Deep multi layer neural network forward pass and backpropagation [3]

functions, learning rate, decay rate, momentum values, number of epochs, batch size and optimization algorithms are the hyperparameters of a MLP. Choosing better hyperparameter values/variables for the network results in better performance. Therefore, finding the best hyperparameters for the methods is a significant issue.

4.3.2 RNN

Moving on, as it has been mentioned before, LSTMs are a type of Recurrent Neural Network (RNN). This deep learning network is used for time series or sequential data, such as language and speech. Although it is possible to face the sequential issue with other NN models, RNNs are preferred due to their ability to include longer time periods. Unlike multi layer perceptron, recurrent networks use internal memory to process incoming inputs. This way, as stated in the literature, RNNs are good at predicting the next character in text, language translation applications, and sequential data preprocessing as time series [71].

The RNN model architecture consists of different numbers of layers and different types of units in each layer. The main difference between RNN and MLP is that each RNN unit takes the current and previous input data at the same time. The output depends on the previous data in the RNN model. RNNs process input sequences one by one at any given time during their operation. The units in the hidden layer hold information about the history of the input in the *state vector*. When the output of the units in the hidden layer is divided into different discrete time steps, an RNN is converted into a MLP, as illustrates in Figure 4.3, [3].

RNNs can be trained using the Backpropagation Through Time (BPTT) algorithm. As well as usual learning method, optimization algorithms are used to adjust the weights. With the BPTT learning method, the error change at time t is reflected in the input and weights of the previous t times. The difficulty of training an RNN is that the RNN structure has a backward dependence over time. Therefore, RNNs become increasingly complex as the learning period increases. Although the main aim of using an RNN is to learn long-term dependencies, studies in the literature show that when knowledge is stored for long time

periods, it is not easy to learn with an RNN [72]. To solve this particular problem, LSTMs with different structures of ANN have been developed [3].

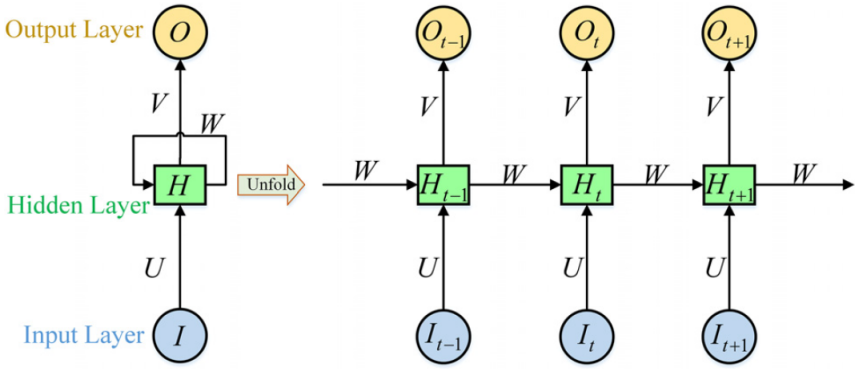


Figure 4.3: Illustration of a Recurrent Neural Network structure folded and unfolded.

where, I is the input layer, H the hidden layer and O the output. Matrix U connects the input layer and the hidden layer, matrix V connects the hidden layer and the output layer, and matrix W connects the hidden layer at different times [73].

Hyperparameters of the RNN also define the network architecture, and the performance of the network is affected by the parameter choices, as in DMLP case. The number of hidden layers, number of units in each layer, regularization techniques, network weight initialization, activation functions, learning rate, momentum values, number of epochs, batch size (*minibatch* size), decay rate, optimization algorithms, model (*Vanilla RNN*, *Gated Recurrent Unit (GRU)*, *LSTM*), and sequence length are the hyperparameters of a RNN.

4.3.3 LSTM

Long Short-Term Memory (LSTM) [74] are a type of RNN where the network can remember both short term and long term values. LSTM networks are the preferred choice of many deep learning model developers when facing complex problems such as automatic speech and handwritten character recognition. LSTM models are mostly used with time-series data. Their applications include Natural Language Processing (NLP), language modeling, language translation, speech recognition, sentiment analysis, predictive analysis, and financial time series analysis [75] [76]. With attention modules and auto encoder (AE) structures, LSTM networks can be more successful in time series data analysis, such as language translation [77].

LSTM networks consist of LSTM units. LSTM units merge to form an LSTM layer. An LSTM unit is composed of cells, each with an input gate, output gate, and forget gate as introduced in Figure 4.4. The goal of these gates is to regulate the information flow. With these features, each cell remembers the desired values over arbitrary time intervals.

- *Forget gate*: The forget gate f_t decides whether to retain or forget the previous information. The decision-making is completed by the *sigmoid* activation function (denoted as σ) of the forget gate, which is composed of the inputs vector I_t and

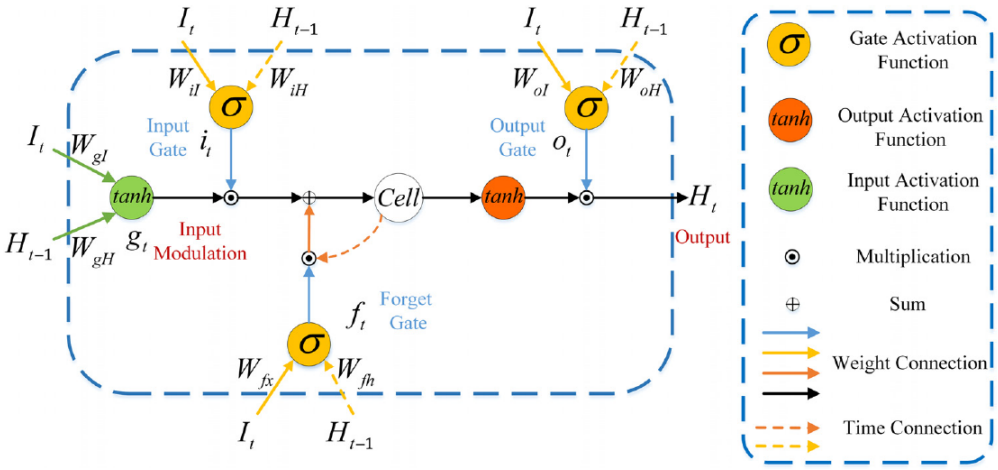


Figure 4.4: A global view of the components of the LSTM model [4].

previous hidden state H_{t-1} and outputs a value between $0-1$. The concrete equation is described as follows:

$$f_t = \sigma(W_{fI} I_t + W_{fH} H_{t-1} + b_f) \tag{4.3}$$

- *Input gate*: The input gate i_t decides whether to update the state using the current input or not. Concretely, this step consists of two parts: A *tanh* activation function is chosen to form the new memory g_t, and the activation function *sigmoid* of the input gate determines which information in the new memory g_t can be updated into the cell state,

$$g_t = tanh(W_{gI} I_t + W_{gH} H_{t-1} + b_g) \tag{4.4}$$

$$i_t = \sigma(W_{iI} I_t + W_{iH} H_{t-1} + b_i) \tag{4.5}$$

By combining Equations 4.3, 4.4 and 4.5, the cell state at the previous moment s_{t-1} can update to the cell state at the current moment s_t

$$s_t = g_t \times i_t + s_{t-1} \times f_t \tag{4.6}$$

- *Output gate*: Finally, the output gate o_t is designed to decide which information is converted from the cell state s_t into current hidden layer H_t. Concretely, after the cell state s_t passes through an activation function *tanh*, its value is placed between -1 and 1 and then multiplied by the output of the activation function *sigmoid* of the output gate, and the information of the cell state can be transferred to the hidden layer

$$o_t = \sigma(W_{oI} I_t + W_{oH} H_{t-1} + b_o) \tag{4.7}$$

$$H_t = o_t \times tanh(s_t), \tag{4.8}$$

where, for all equations, σ and *tanh* are denoted as the *sigmoid* and *tanh* activation functions. W_{fI}, W_{iI}, W_{gI}, W_{oI} are weight matrices between the input layer I_t and the hidden layer H_t at time *t*. W_{fH}, W_{iH}, W_{gH}, W_{oH} are hidden layer weight matrices between current moment t and previous moment $t-1$. And finally, b_f, b_i, b_g, b_o are bias matrices.

This way the cell memory enables LSTMs to handle long-term dependencies, because information can remain in the memory for many steps. Several LSTM layers can be stacked on top of each other in a model or having a different architecture, such as, *bidirectional LSTM* [78]. However, the last LSTM layer is followed by a traditional fully connected dense layer (output layer), which the number of output neurons corresponds to how many values (features or steps) are going to be predicted in each repetition.

LSTM models are a specialized version of RNN. Thus, the weights updates and preferred optimization methods are the same. In addition, the hypeparameters of LSTM are just like those of RNN.

4.4 Proposed RUL prediction

The proposed LSTM-based prediction framework is divided in two main tasks, firstly, in order to train the model, data must be organize in a specific way to go into it. Then, the proposed model will take care of training internal weights and to forecast the degradation of discharge capacity.

Figure 4.5 introduces the learning process of the framework. Firstly, the time series of each battery experiment are reorganized and considered in a supervised learning scenario. This means that a set of finite D_k data, of instance k and length L (called as *window*), will be matched with the next cycle, in this case, D_{k+1}. This means that, the model will try to predict the future value, D_{k+1}, from L history data at each iteration. This way, the predictive model will learn internal patterns and connections between past data and the future value. Note that in order to continue forecasting, it is necessary to predict every features used on the training to feed the model. That's why it is used the same notation for each window data and future value, D.

Highlight that as the data of each battery is reorganized, all the supervised data pairs can be come together to create a huge supervised dataset within different fast-charging experiments, named as multiple time series forecasting. Therefore, taking 100 cycles of window to predict the next cycle, there are more than 120.000 supervised pairs, which most of them will be used for training the model.

Due to time limitation, a hold-out strategy was decided to use opposed to widely known *k-fold cross-validation*. In this work, randomly 80% of the batteries were chosen to train, 108 batteries, and the other 20% for testing, 27 batteries. Although the goal of those testing fast-charging experiments, or batteries, is to evaluate the performance of the model, the first cycles of these batteries can not be forecast as the model needs an initial starting point. This way, this initial data will be used to for training and, in the next Chapter 5, the significant of the amount of data used for initializing will be studied. Note that this

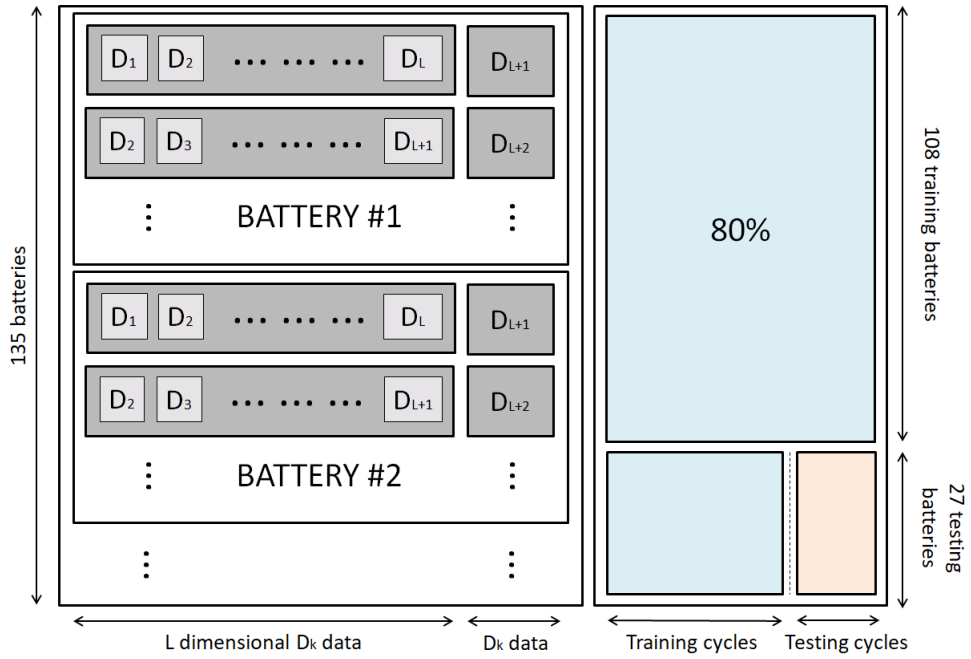


Figure 4.5: Input format and configurations for training.

information is illustrated on the lower right part of the Figure 4.5, named as *training cycles*, where the first cycles of testing batteries are colored in blue, as the rest of training data.

In regarding to the proposed model, in this work a stacked LSTM neural network is presented in Figure 4.6. The predictive model is divided in two main parts, first two LSTM layers are stacked with the aim of understanding the patterns of the input time series. While on the second part, a dense layer is used to generate the future value from the internal information generated by those LSTM layers.

This way, in the training task, all the *supervised* pairs will be used to adjust all the inside weights of the model. To do so, the window will be used as the input data and the future value, the next cycle, as the output data. However, on the testing or forecasting task, the last window of the training data of each battery will be used as the starting point and in each repetition the predicted cycle will be including on the window, as it is illustrated in Figure 4.7. For instance, if the goal is to predict 500 cycles, the model will need 500 iterations to predict the whole time series, as the model predicts only the next cycle in each repetition. Moreover, due to predicted cycles must include on the window, and as the window has a finite length (in this work of 100 cycles), the window will easy be completed by predicted cycles, getting a huge uncertainty in upcoming predictions. Carrying on with the example, 400 cycles of those 500, will be forecast taking into account a complete uncertainty window.

As regards to the input and output data, aforementioned, the same features, D, must be used, although in the input data a set of this data is used. According to which features used on the prediction, a interesting debate appears. From one side, if the objective is to predict from a particular dataset, for instance *Severson* dataset [12], apriori more features will be beneficial to the predictive neural network model in order to get an accurate forecasting. However, from the other side, if the objective is to generalize the model to other battery

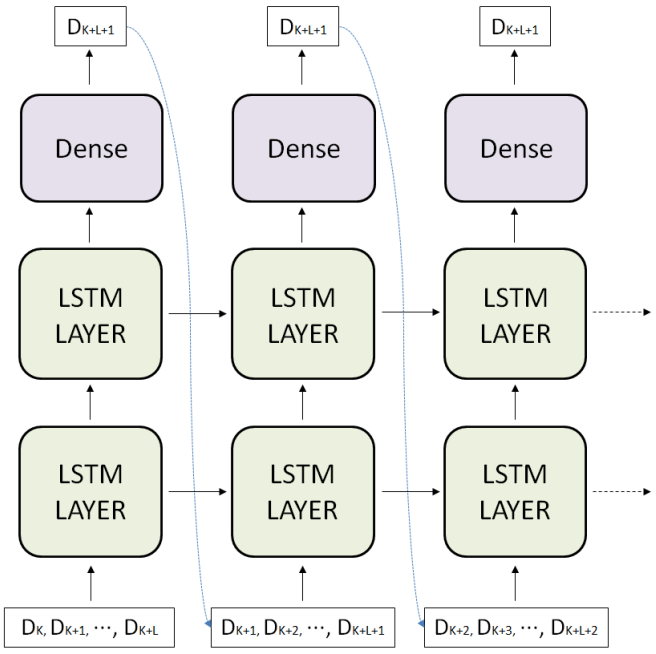


Figure 4.6: An illustration of the proposed LSTM neural network.

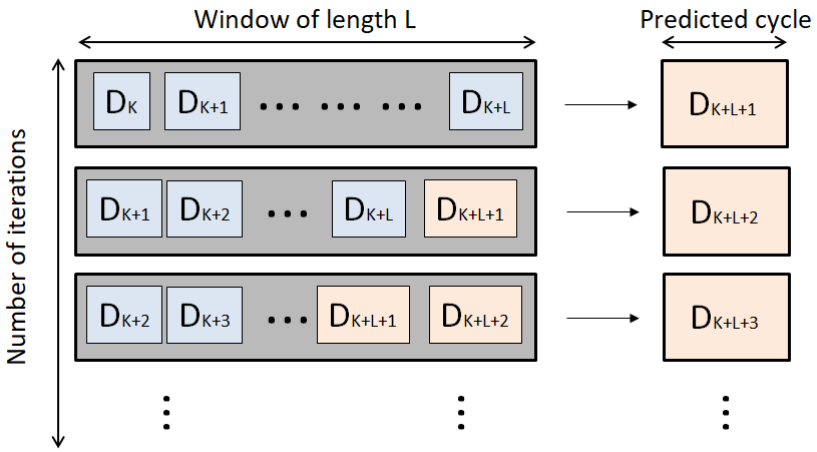


Figure 4.7: An illustration of forecasting process, where predicted cycles, oranges, are introduced on the window in each iteration. Note that blue color represents the real or raw cycles.

Remaining Useful Life (RUL) prediction, it could be hard to find another dataset with the same features, as the infrastructure of generating the data will rarely be the same. For example, NASA's dataset [49] only has the evolution of the discharge capacity (QD) and the current (I), voltage (V) and temperature (T) used on each cycle.

Apart from this, the goal of this work is focused on the importance of each feature on the performance of the proposed model, Section 1.4. Therefore, in this investigation three different data will be use to forecast the evolution of the discharge capacity (QD). Firstly,

only this feature will be used to forecast itself, called as univariate time series (Eq. 4.9a). Secondly, due to the feature selection done in Section 3.3, discharge capacity and internal resistance will be used, named as bivariate time series (Eq. 4.9b). Finally, all the features that explains an experiments will be used, called as multivariate time series (Eq. 4.9c).

$$D_k = QD_k \tag{4.9a}$$

$$D_k = \begin{cases} QD_k \\ IR_k \end{cases} \tag{4.9b}$$

$$D_k = \begin{cases} QD_k \\ IR_k \\ QC_k \\ Tmax_k \\ Tavg_k \\ Tmin_k \end{cases} \tag{4.9c}$$

where k represents a cycle.

Unfortunately, the second dimension or features referred to a cycle will not be taken into account, due to time limitation. Nevertheless, as described in the final chapter, this line will stand as one of the initial future lines to continue this work.

CHAPTER 5

Computational studies

In order to obtain a preliminary picture of the performance of the studied LSTM models, in this chapter, a battery of experiments are conducted. To that end, first, the experimental framework is presented, which settles down the environment used in the four experiments that are described later on.

5.1 Experimental framework

The code is implemented by Tensorflow in Python [79] and executed on a machine with the next characteristics: Intel i5-4590 CPU with 3.30 Ghz and 8GB memory, and the average training time is around 20 minutes. Although the predictive model was proposed, for the purpose of improving its performance some hyperparameters must be tuned. Despite the list of hyperparameters can be long, in this work the most important ones will be tuned and for others the most common values according to *Severson* data [12] will be used.

Before explaining the hyperparameters, it is interesting to highlight that all the training data were normalized using *Max normalization* technique, Equation 5.1, to redefine all the data between 0 and 1. The main reason of using this normalization is that, this way, the evolution of discharge capacity (QD) is converted to widely used state-of-health (SOH) feature, a visual feature to easily control the threshold of 80%, as it is equal to 0.8. The max normalization is defined as:

$$\text{Max normalization} = \frac{value}{max} \tag{5.1}$$

where *max* refers to the maximum value of each feature. For example, as nominal capacity of these batteries is 1.1 Ah, the maximum value, *max*, of discharge capacity is set to this value.

According to the activation functions in the LSTM model, *ReLU*, Eq. 5.2, function has been considered. Each layer will consists of 50 units and Adam optimizer [69] is used to learn the parameters of the model. Similarly, the dense layer will also use *ReLU* as activation function, but the number of neurons depends on the number features. This

means that in order to predict a feature, a neuron is needed. Moreover, as regards to training hyperparameters, the batch size is set to 800 and 25 epoch are settled down.

$$ReLU(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & z \leq 0 \end{cases} \tag{5.2}$$

In relation to loss function, it is said that there is not a single loss function that works for all kind of data. This means that it is suggested to attempt with different loss function to identify the best one for the learning scenario, as it can depend on great number of factors including the presence of outliers, the machine learning algorithm or ease of finding the derivatives. In this work *Mean Square Error (MSE)* is chosen after having a try with *Huber Loss*, *Log-Cosh Loss* and *Mean Absolute Error*. Aforementioned MSE [80] is the most commonly used regression loss function, in which it is the sum of squared distances between the target variable, y_i, and predicted values $\hat{y}_i$, Eq. 5.3.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{5.3}$$

Finally, as a way to evaluate the forecasting *Mean Absolute Error (MAE)* [81] metric will be used. In spite of this, there are other metrics that could be used, such as, *R-squared*, *Mean Squared Error (MSE)* or *Mean Absolute Percentage Error (MAPE)* metric. The reason why MAE is used in this work is because its easy interpretability and the possibility to compare different predictions with different lengths, as MAE calculates the error you can expect from the forecast on average, Eq. 5.4.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{5.4}$$

where the resulting error is between 0 and 1. Zero value means a perfect prediction, while increasing the MAE value the prediction is getting worse. This way, for instance, 0.3 error means that the difference between the prediction and the real value is about 30%.

Note that in this work the accuracy of the model is calculated by comparing the preprocessed raw data and the predicted data, this way, the evaluation is purely based on time series, rather than the significance of the time series. In other words, although it could be possible to measure the performance of predictive model by the difference between the real and predicted last cycle above the threshold, in this project this idea was discharged as not all the predicted time series end under the limit.

To sum up, taking into consideration the proposed model with its hyperparameters, *Severson* dataset and experimental framework, 4 consecutive experiments have been carried out. Since each of them analyzes a particular aspect of the performance of the LSTM model considered, the content has been divided in separated sections.

5.2 Experiment 1: The performance of the LSTM model

The objective of the first experiment is to show the performance of predictive model in different scenarios. To do so, those three input data will be taken into account, univariate,

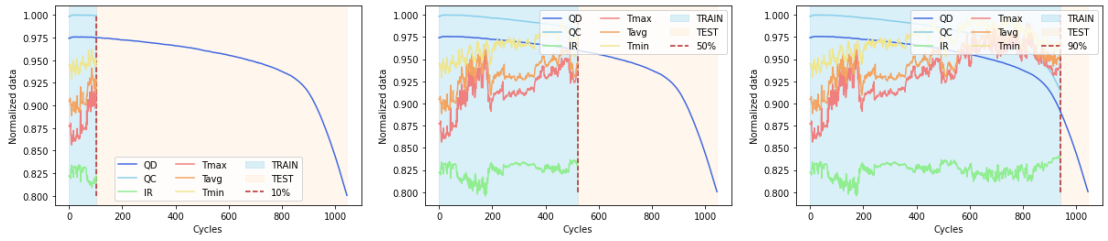
Equation 5.5a; bivariate, Equation 5.5b, and multivariate, Equation 5.5c.

$$D_k = QD_k \tag{5.5a}$$

$$D_k = \begin{cases} QD_k \\ IR_k \end{cases} \tag{5.5b}$$

$$D_k = \begin{cases} QD_k \\ IR_k \\ QC_k \\ Tmax_k \\ Tavg_k \\ Tmin_k \end{cases} \tag{5.5c}$$

On the other hand, as it was explained on Figure 4.5 and Section 4.4, the number of initial cycles of testing batteries will be varied to predict the rest of the time series, as it is illustrated in Figure 5.1. For instance, in Figure 5.1a only 10% of the data is used for training and the rest for testing or forecasting. Similarly, Figure 5.1b uses 50% of the data to train and Figure 5.1c 90% of the time series. Although in the illustrations a multivariate time series, with 6 features, are displayed for training and discharge capacity for testing, it is just for a better illustration as they must be the same. Moreover, despite the figure gives an example of multivariate time series to train, as all the features are included, not necessarily all of them will be used.



(a) Illustration of using 10% for initializing or training.

(b) An example of testing 50% the fast-charging battery experiment.

(c) Illustration of using 90% of testing batteries for training.

Figure 5.1: Examples of using 10, 50 and 90% of testing batteries for training.

This way, Figure 5.2 shows the performance of the predictive model taking into account three input data types, univariate, bivariate and multivariate, and the number of initializing data, in percentage. In addition, as the training process has randomness, 10 repetitions have been done in each permutation between input data and the amount of training data on those testing data (3 input data types, 9 different training data and 10 repetitions). This way, the mean value is represented as a point and the shading represents the inter-quartile range, between first and third quartile.

Several conclusions can be drawn from figure, firstly, the proposed model's performance is inferior when more features are included. Although, it is believed that neural networks can distinguish between useful and useless features to give more importance to some features, apparently, the proposed LSTM model is unable to do the right comprehension of the data.

Secondly, as it is expected, increasing the length of the prediction or reducing the number of training data for those testing batteries, prediction is getting worse. In other

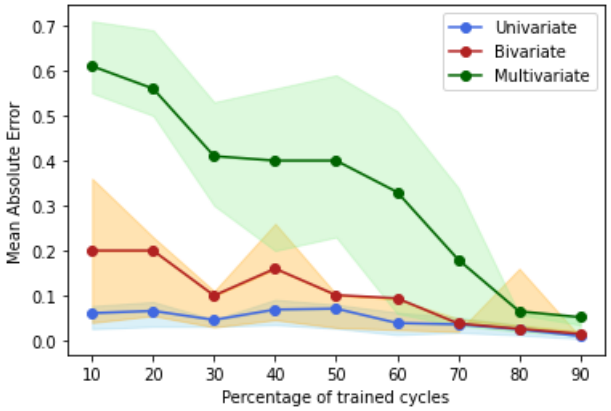
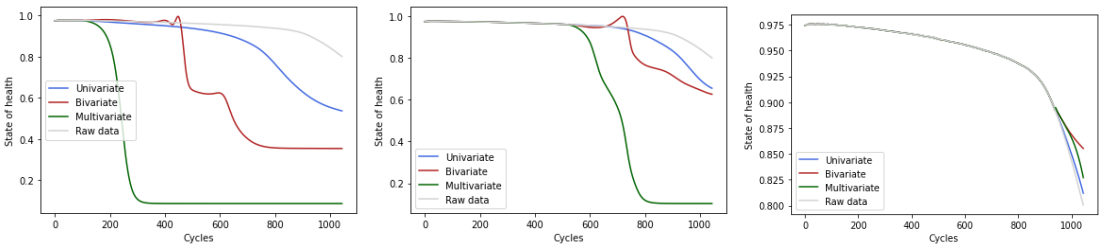


Figure 5.2: The performance of proposed LSTM model with three different inputs.

words, the forecast is easier when the goal is to predict the near future, rather than long future. Specifically, we see that the multivariate case is the one that worst performance shows, followed by bivariate, and being the univariate the best performing option, under 10% of mean error on all scenarios. The number of predictive values in each instance, hence the number of features, could be a reason for this inferior performance, as the number of features increases, the number of possible error values will also rise. Moreover, this possible inaccuracies will grow while when the greater the length of forecasting since the predicted cycles are included on the window.

5.3 Experiment 2: Describing the predicted time series

In the second experiment, the forecast time series are illustrated in order to better understand the model's performance. Figure 5.3 introduces the mean of those 10 repetitions in each scenario.



(a) Illustration of predictive ability of three model after 10% of training data.

(b) The output of forecasting 50% of the time series.

(c) Illustration of model's performance to predict the last 10% of the cycles.

Figure 5.3: Performance of proposed model using univariate, bivariate and multivariate time series, and training the *b3c43* battery on three scenarios, 10, 50 and 90% of initialization.

Firstly, it can be seen that when the objective is to forecast near future or in this case training with 90%, Figure 5.3c, the performance of proposed three models are similar. However, increasing the number of predicted cycles and taking into consideration that

proposed LSTM model forecast only one cycle in each iteration (Fig. 4.7), the uncertainty is heavily introduced when a great number of iterations are done, as it can be observed in Figures 5.3a and 5.3b, thus getting worse results. This way, the initial believe of challenge on long distance prediction is reaffirmed.

Secondly, taking into account the results of MAE, Figure 5.2, and these predictions, univariate time series' superiority over others is reflected. On the contrary, multivariate time series in the worst. Among possible reasons to explain this behaviour, it seems that the proposed model outperforms with less features, and included features achieve an opposite outcome although the feature analysis and feature is already done to avoid this situation.

Thirdly, in spite of error calculated on univariate time series using 10% and 90% seem similar (Fig. 5.2), from battery life prediction an important difference between them is perceived. In other words, MAE metric measures the mean error of all predicted time series and it does not quantify the error at the end. Therefore, despite the dissimilarity at the end of predicted time series in Figure 5.3a, as a great number of initial cycles are accurately predicted, a small overall MAE error is obtained.

Finally, the major difference between proposed multivariate forecasting and the others is that almost always the prediction drops very fast and a constant 0 value is forecast for the rest of the time series, while the other two models try to forecast the initial slow capacity degradation. This way, the MAE error increases so much as the length is also increasing, Figure 5.2.

5.4 Experiment 3: Over or underfitting?

Due to the unexpected inability to, at least, achieve similar results using more information, the idea of bad training conditions came up. Therefore, the evolution of the *Mean Squared Error (MSE)* loss during the training is displayed in Figure 5.4.

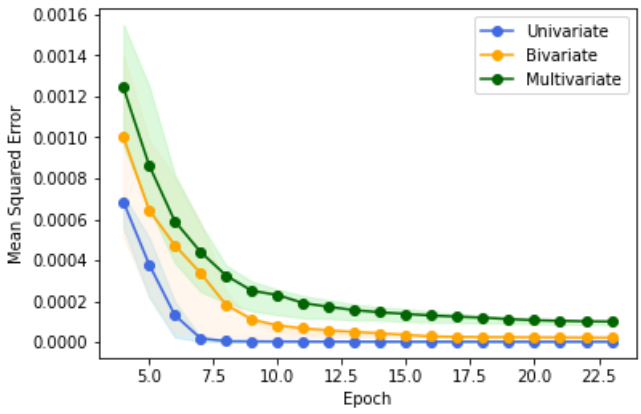


Figure 5.4: The evolution MSE loss during the training. Note that first 4 epoch are not shown to focus on the difference between different models' losses.

Here there are also few notes to highlight. Firstly, there is a substantial difference on the training losses. Univariate time series usually reach at *2e-06* error while multivariate stack at *3e-04*, when they should be similar. Secondly, there might be a small overfitting

on the training as it seems that after 15 epoch the model does not improve. In order to fix this issue, an early stop was proposed to stop the training when the loss of the current epoch was higher than the previous one. However, we find that rarely this situation was happening before 25 epoch. This means that although the improvement is slow or hardly any, an improvement takes place.

On the other hand, the biggest surprise was to find almost zero MSE error during the training. Even though this meaningless error should be an positive point, in this scenario where the predictive model is not as good as expected, this zero error is, at least, confusing. On the search of a possible reason, we came across that there is a huge similarity between the two sides of *supervised learning* pairs, Figure 4.5. Here the interesting point is that as the evolution of the capacity is so slow, the variability is almost none, specially of the first part of the time series.

5.5 Experiment 4: Is preprocessing useful for prediction?

Finally, in the last experiment, the need of the feature selection of the data is revised. Note that in this work the preprocessing consists of outlier detection and correction, as explained on Chapter 3. To do so, already proposed univariate time series forecasting is used to train with first, preprocess data as it is done in previous experiments, and later with data without preprocessing.

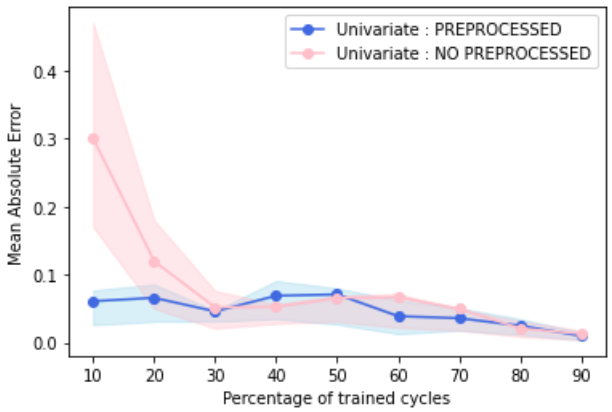


Figure 5.5: Proposed univariate predictive model is trained and testing in different initial data scenarios and with and without preprocessing.

Figure 5.5 introduces the experiment results, similar to Figure 5.2. Like the faculty of neural networks to learn similarly whether or not doing a feature selection, it is also believed that they can master similar knowledge from with or without preprocessing the data. According the results of this experiment, this faculty is not observed when the model is trained with 10% and 20%. In other words, outliers have manifest an error on long-term forecasting, although in short predictions seems not to have much relevance. Therefore, in batteries Remaining Useful Life (RUL) prediction scenarios, as the goal is to predict with very few cycles, the outlier detection and correction is suggested on LSTM based predictive models.

CHAPTER 6

Conclusions and future work

In the last chapter an overall summary and conclusions of the investigation are presented, complemented by organized future works.

6.1 Conclusions

Battery life prediction is important for the prognostics and health of battery management systems. This way, Remaining Useful Life (RUL) prediction can provide the probable failure time of lithium-ion batteries (LIB) in advance. The main challenge of RUL forecasting for LIB lies in how to accurately learn the long-term dependencies over hundreds of cycles based on degradation data. In this work, an initial approach for RUL prediction is presented, where first a background of RUL forecasting complemented with *Severson* [12] dataset is introduced.

Secondly, we develop an outlier detection and correction in order to, on one hand, gain a better understanding of lithium-ion batteries features, and, on the other hand, improve the performance of predictive model. Moreover, a feature selection is proposed to understand relationships between features measured with *Pearson*, *Spearman* correlation, as well as, mutual information.

Finally, we proposed data-driven LSTM-based RUL prediction methods for lithium-ion batteries on different learning scenarios. Experiments on *Severson* fast-charging batteries dataset exhibit an accurate prediction on short-term scenarios, while it is getting worse on long-term predictions. In spite of this, the proposed univariate time series forecasting outperforms on all training scenarios measuring less than 0.1 of MAE and getting an interesting prediction using 100 cycles for an initial approach (Fig. 6.1).

6.2 Future works

Future works are presented in an organized list according to time, where the tasks are planned to be done from top to bottom

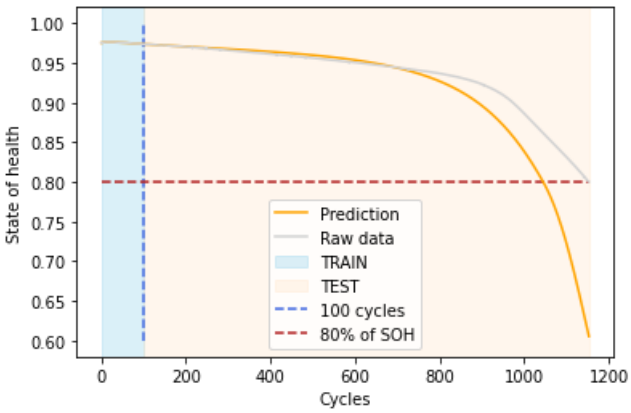


Figure 6.1: An example of a good prediction of proposed univariate model on *b3c19* fast-charging experiment using 100 cycles.

- **Optimize proposed predictive model:** In order to rely on data-driven predictive model for RUL prediction, it is vital to forecast accurately long distances as the aim is to forecast with as less cycles as possible. Therefore here some suggestions:

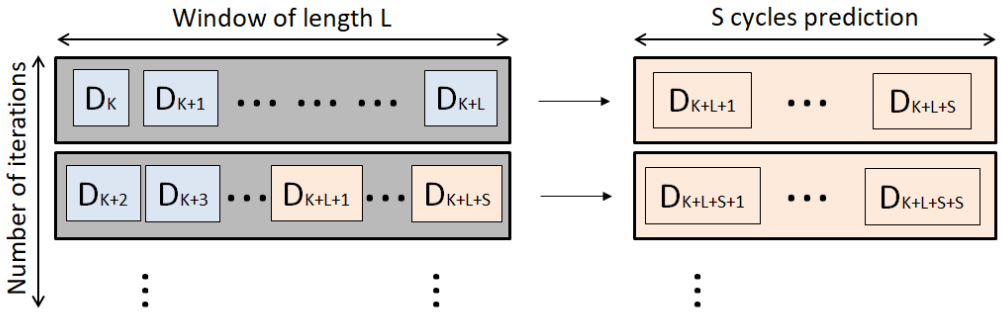
  - A thorough investigation on proposed model LSTM hyperapameters and hierarchies.
  - Train the model with additional information, such as, charging policies or taking into account features that explains a cycle, called in this work as second dimension. *Park et al.* [54] proposed a way of introducing the second dimensionality using this type of data (Eq. 6.1).

$$D_k = \begin{cases} QD_k \\ V_k^1, V_k^2, ..., V_k^S \\ I_k^1, I_k^2, ..., I_k^S \\ T_k^1, T_k^2, ..., T_k^S \end{cases} \tag{6.1}$$

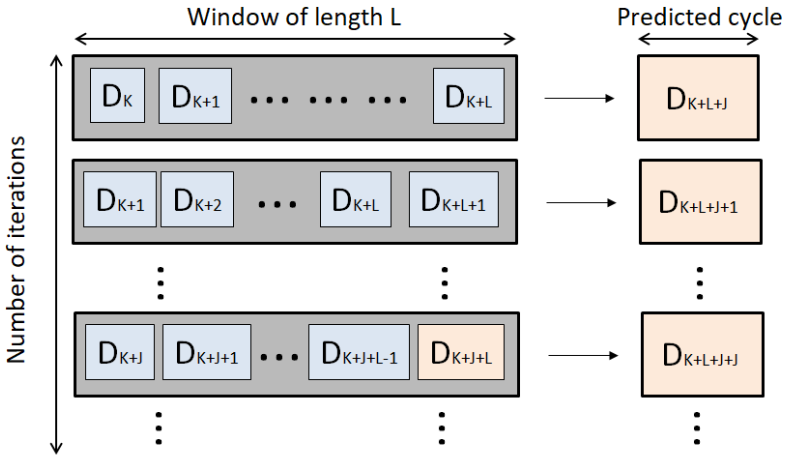
    where V and I corresponds to the distribution of the voltage, (Fig. 2.5b) and current, (Fig. 2.5a); whereas, T represents the evolution of the temperature (Fig. 2.8a).

    Moreover, S is the number of sample points during one charging/discharging profile. According to *Park et al.* [54] investigation: "For efficient learning of LSTM, it is not desired to utilize all data, even if there are many voltage, current and temperature points in the charging process according to the BMS's settigns. We choose S=10 in Equation 6.1, i.e., 10 sample points for each charging profile of voltage, current and temperature as done in [82][83]."
  - Change forecasting strategy: Although in this work the strategy was to predict windows' next cycle in each repetition (Fig. 4.7), a drawback was found and introduced on the third experiment (Section 5.4). The window used for predicting and the next cycle are very similar, getting a meaningless variance and loss error during training. Thus two new strategies consisting on predicting more

than one cycles, *S*, in each repetition (Fig. 6.2a) and forecasting a J distance cycle instead of next cycle (Fig. 6.2b), tries to forecast in a different point of view, such that the a better training is possible.



(a) Introduction of predicting S cycles strategy. Note that S cycles will be predicted in a row and all of them will be included on the window to continue forecasting.



(b) An illustration of the second predictive plan. Highlight that in this strategy more to the past cycles are used to predict current cycle, instead of using prior cycles.

Figure 6.2: Illustration of new forecasting strategies and their application on RUL prediction.

- **Compare:** After optimizing the proposed model, it could be a good chance to validate the performance predictive with different datasets such as, data generated from CIDETEC, NASA's lithium-ion degradation battery data [49] or recently publicated ENPOLITE dataset [84]. On the other hand, it could be of interest to compare with other physic-based or data-driven predictive models, such as, the one developed by Technical University of Denmark (DTU) or with proposed by *Park et al.* [54].
- **Hybrid model:** Finally, it could be intriguing an hybrid model between previously developed physic-based model by CIDETEC and proposed data-driven model in order to, first, learn how to complement both models, and then, analyze and compare the results with other models to opt for the best one for RUL prediction.

Bibliography

[1] Yuejiu Zheng, Minggao Ouyang, Xuebing Han, Languang Lu, and Jianqiu Li. Investigating the error sources of the online state of charge estimation methods for lithium-ion batteries in electric vehicles. *Journal of Power Sources*, 377:161–188, 2018. See pages vii, 3.

[2] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021. See pages vii, 21, and 22.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. See pages viii, 37, 41, and 42.

[4] Xiaosong et al. Hu. A particle filter and long short-term memory fusion technique for lithium-ion battery remaining useful life prediction. *Journal of Dynamic Systems, Measurement, and Control*, 143(6), 2021. See pages viii, 43.

[5] Severson et al. Data-driven prediction of battery cycle life before capacity degradation. Supplementary Information. *Nature energy volume 4, pages 383–391*, 2019. See pages ix, 11.

[6] Melanie Loveridge Stephen Gifford and Martin Dowson. Why Batteries Fail and How To Improve Them: Understanding Degradation To Advance Lithium-Ion Battery Performance. *Faraday Insights*, 2021. See page 4.

[7] Dickson NT How, MA Hannan, MS Hossain Lipu, and Pin Jern Ker. State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review. *IEEE Access*, 7:136116–136136, 2019. See page 5.

[8] Lijun Zhang, Zhongqiang Mu, and Changyan Sun. Remaining useful life prediction for lithium-ion batteries based on exponential model and particle filter. *IEEE Access*, 6:17729–17740, 2018. See page 5.

[9] Jianbo Yu. State-of-health monitoring and prediction of lithium-ion battery using probabilistic indication and state-space model. *IEEE Transactions on Instrumentation and Measurement*, 64(11):2937–2949, 2015. See page 5.

[10] Yujie Cheng, Noureddine Zerhouni, and Chen Lu. A hybrid remaining useful life prognostic method for proton exchange membrane fuel cell. *International Journal of Hydrogen Energy*, 43(27):12314–12327, 2018. See page 5.

[11] Yi Li, Kailong Liu, Aoife M Foley, Alana Zülke, Maitane Berecibar, Elise Nanini-Maury, Joeri Van Mierlo, and Harry E Hoster. Data-driven health estimation and lifetime prediction of lithium-ion batteries: A review. *Renewable and sustainable energy reviews*, 113:109254, 2019. See page 5.

[12] Kristen A Severson, Peter M Attia, Norman Jin, Nicholas Perkins, Benben Jiang, Zi Yang, Michael H Chen, Muratahan Aykol, Patrick K Herring, Dimitrios Fraggedakis, et al. Data-driven prediction of battery cycle life before capacity degradation. *Nature Energy*, 4(5):383–391, 2019. See pages 6, 9, 11, 15, 35, 45, 49, and 55.

[13] Juan Antonio Cortés-Ibáñez, Sergio González, José Javier Valle-Alonso, Julián Luengo, Salvador García, and Francisco Herrera. Preprocessing methodology for time series: An industrial world application case study. *Information Sciences*, 514:385–401, 2020. See page 21.

[14] Anthony J Fox. Outliers in time series. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(3):350–363, 1972. See page 21.

[15] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980. See page 21.

[16] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015. See page 22.

[17] Kumar Gaurav Ranjan, B Rajanarayan Prusty, and Debashisha Jena. Review of preprocessing methods for univariate volatile time-series in power system applications. *Electric Power Systems Research*, 191:106885, 2021. See page 22.

[18] Lianfang Cai, Nina F Thornhill, Stefanie Kuenzel, and Bikash C Pal. Real-time detection of power system disturbances based on k-nearest neighbor analysis. *IEEE Access*, 5:5631–5639, 2017. See page 23.

[19] Jiyi Chen, Wenyuan Li, Adriel Lau, Jiguo Cao, and Ke Wang. Automated load curve data cleansing in power systems. *IEEE Transactions on Smart Grid*, 1(2):213–221, 2010. See page 23.

[20] Yufeng Yu, Yuelong Zhu, Shijin Li, and Dingsheng Wan. Time series outlier detection based on sliding window prediction. *Mathematical problems in Engineering*, 2014, 2014. See page 24.

[21] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004. See page 29.

[22] PM Vacek and T Ashikaga. An examination of the nearest neighbor rule for imputing missing values. *Proc. Statist. Computing Sect., Amer. Statist. Ass*, pages 326–331, 1980. See page 29.

[23] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. Comparison of different methods for univariate time series imputation in r. *arXiv preprint arXiv:1510.03924*, 2015. See page 29.

[24] Faraj Bashir and Hua-Liang Wei. Handling missing data in multivariate time series using a vector autoregressive model-imputation (var-im) algorithm. *Neurocomputing*, 276:23–30, 2018. See page 29.

[25] Alexandros Kalousis, Julien Prados, and Melanie Hilario. Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12(1):95–116, 2007. See page 30.

[26] Vipin Kumar and Sonajharia Minz. Feature selection: a literature review. *SmartCR*, 4(3):211–229, 2014. See page 30.

[27] Hoai Bach Nguyen, Bing Xue, and Peter Andreae. Mutual information estimation for filter based feature selection using particle swarm optimization. In *European Conference on the Applications of Evolutionary Computation*, pages 719–736. Springer, 2016. See page 30.

[28] Fengxi Song, Zhongwei Guo, and Dayong Mei. Feature selection using principal component analysis. In *2010 international conference on system science, engineering design and manufacturing informatization*, volume 1, pages 27–30. IEEE, 2010. See page 31.

[29] Lars Dannecker. *Energy time series forecasting: efficient and accurate forecasting of evolving time series from the energy domain*. Springer, 2015. See page 31.

[30] Shaifu Gupta, Aroor Dinesh Dileep, and Timothy A Gonsalves. A joint feature selection framework for multivariate resource usage prediction in cloud servers using stability and prediction performance. *The Journal of Supercomputing*, 74(11):6033–6068, 2018. See page 31.

[31] Jiaqi Ye, Chengwei Xiao, Rui Máximo Esteves, and Chunming Rong. Time series similarity evaluation based on spearman's correlation coefficients and distance measures. In *Second*

International Conference on Cloud Computing and Big Data in Asia, pages 319–331. Springer, 2015. See page 31.

[32] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004. See page 32.

[33] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014. See page 32.

[34] Shoffi Izza Sabilla, Riyanarto Sarno, and Kuwat Triyana. Optimizing threshold using pearson correlation for selecting features of electronic nose signals. *Int. J. Intell. Eng. Syst*, 12(6):81–90, 2019. See page 32.

[35] Di Liu, Siu-Yeung Cho, Dong-mei Sun, and Zheng-Ding Qiu. A spearman correlation coefficient ranking for matching-score fusion on speaker recognition. In *TENCON 2010-2010 IEEE Region 10 Conference*, pages 736–741. IEEE, 2010. See page 32.

[36] Yongzhi Zhang, Rui Xiong, Hongwen He, and Michael G Pecht. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, 67(7):5695–5705, 2018. See page 36.

[37] Wei He, Nicholas Williard, Michael Osterman, and Michael Pecht. Prognostics of lithium-ion batteries based on dempster–shafer theory and the bayesian monte carlo method. *Journal of Power Sources*, 196(23):10314–10321, 2011. See page 36.

[38] Yinjiao Xing, Eden WM Ma, Kwok-Leung Tsui, and Michael Pecht. An ensemble model for predicting the remaining useful performance of lithium-ion batteries. *Microelectronics Reliability*, 53(6):811–820, 2013. See page 36.

[39] Linxia Liao and Felix Köttig. A hybrid framework combining data-driven and model-based methods for system remaining useful life prediction. *Applied Soft Computing*, 44:191–199, 2016. See page 36.

[40] Lei Ren, Li Zhao, Sheng Hong, Shiqiang Zhao, Hao Wang, and Lin Zhang. Remaining useful life prediction for lithium-ion battery: A deep learning approach. *IEEE Access*, 6:50587–50598, 2018. See page 36.

[41] Xiaosong Hu, Jiuchun Jiang, Dongpu Cao, and Bo Egardt. Battery health prognosis for electric vehicles using sample entropy and sparse bayesian predictive modeling. *IEEE Transactions on Industrial Electronics*, 63(4):2645–2656, 2015. See page 36.

[42] Sheng Hong, Baoqing Wang, Guoqi Li, and Qian Hong. Performance degradation assessment for bearing based on ensemble empirical mode decomposition and gaussian mixture model. *Journal of Vibration and Acoustics*, 136(6), 2014. See page 36.

[43] Bhaskar et al. Saha. An integrated approach to battery health monitoring using bayesian regression and state estimation. In *2007 IEEE Autotestcon*, pages 646–653. Ieee, 2007. See page 37.

[44] Jianbao et al. Zhou. Study on the reconfigurable remaining useful life estimation system for satellite lithium-ion battery. *Chinese Journal of Scientific Instrument*, 34(9):2034–2044, 2013. See page 37.

[45] A. et al. Nuhic. Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods. *Journal of power sources*, 239:680–688, 2013. See page 37.

[46] D. et al. Liu. A health indicator extraction and optimization framework for lithium-ion battery degradation modeling and prognostics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(6):915–928, 2015. See page 37.

[47] Meru et al. Patil. A novel multistage support vector machine based approach for li ion battery remaining useful life estimation. *Applied energy*, 159:285–297, 2015. See page 37.

[48] Jie et al. Liu. An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. Technical report, National Aeronautics And Space Administration Moffett Field CA Ames Research ..., 2010. See page 37.

[49] B. Saha and K. Goebel. Battery data set", nasa ames prognostics data repository (http://ti.arc.nasa.gov/project/prognostic-data-repository). *NASA Ames Research Center, Moffett Field, CA*, 2007. See pages 37, 46, and 57.

[50] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. See page 37.

[51] Yongzhi Zhang and Xiong et al. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, 67(7):5695–5705, 2018. See page 37.

[52] Bin Xiao, Yonggui Liu, and Bing Xiao. Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer. *Ieee Access*, 7:54192–54202, 2019. See page 37.

[53] Yuchen Song, Lyu Li, Yu Peng, and Datong Liu. Lithium-ion battery remaining useful life prediction based on gru-rnn. In *2018 12th international conference on reliability, maintainability, and safety (icrms)*, pages 317–322. IEEE, 2018. See page 37.

[54] Kyungnam Park, Yohwan Choi, Won Jae Choi, Hee-Yeon Ryu, and Hongseok Kim. Lstm-based battery remaining useful life prediction with multi-channel charging profiles. *IEEE Access*, 8:20786–20798, 2020. See pages 37, 56, and 57.

[55] Chao Hu, Gaurav Jain, Puqiang Zhang, Craig Schmidt, Parthasarathy Gomadam, and Tom Gorka. Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. *Applied Energy*, 129:49–55, 2014. See page 38.

[56] Xiujuan Zheng and Huajing Fang. An integrated unscented kalman filter and relevance vector regression approach for lithium-ion battery remaining useful life and short-term capacity prediction. *Reliability Engineering & System Safety*, 144:74–82, 2015. See page 38.

[57] J Liu, W Wang, F Ma, YB Yang, and CS Yang. A data-model-fusion prognostic framework for dynamic system state forecasting. *Engineering Applications of Artificial Intelligence*, 25(4):814–823, 2012. See page 38.

[58] Wei He, Nicholas Williard, Michael Osterman, and Michael Pecht. Prognostics of lithium-ion batteries based on dempster–shafer theory and the bayesian monte carlo method. *Journal of Power Sources*, 196(23):10314–10321, 2011. See page 38.

[59] Chao Hu, Byeng D Youn, Pingfeng Wang, and Joung Taek Yoon. Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. *Reliability Engineering & System Safety*, 103:120–135, 2012. See page 38.

[60] Pedro Lara-Benítez, Manuel Carranza-García, and José C Riquelme. An experimental review on deep learning architectures for time series forecasting. *arXiv preprint arXiv:2103.12057*, 2021. See page 39.

[61] Helmut Lütkepohl. Structural vector autoregressive analysis in a data rich environment: A survey. 2014. See page 39.

[62] Parikshit Gautam Jamdade and Shrinivas Gautamrao Jamdade. Modeling and prediction of covid-19 spread in the philippines by october 13, 2020, by using the varmax time series method with preventive measures. *Results in Physics*, 20:103694, 2021. See page 39.

[63] Mohammad Valipour, Mohammad Ebrahim Banihabib, and Seyyed Mahmood Reza Behbahani. Parameters estimate of autoregressive moving average and autoregressive integrated moving

average models and compare their ability for inflow forecasting. *J Math Stat*, 8(3):330–338, 2012. See page 39.

[64] Meru A Patil, Piyush Tagade, Krishnan S Hariharan, Subramanya M Kolake, Taewon Song, Taejung Yeo, and Seokgwang Doo. A novel multistage support vector machine based approach for li ion battery remaining useful life estimation. *Applied energy*, 159:285–297, 2015. See page 39.

[65] Taichun Qin, Shengkui Zeng, and Jianbin Guo. Robust prognostics for state of health estimation of lithium-ion batteries based on an improved pso–svr model. *Microelectronics Reliability*, 55(9-10):1280–1284, 2015. See page 39.

[66] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. See page 40.

[67] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. See page 40.

[68] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. See page 40.

[69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. See pages 40, 49.

[70] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. See page 40.

[71] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014. See page 41.

[72] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013. See page 42.

[73] Jinglong Chen, Hongjie Jing, Yuanhong Chang, and Qian Liu. Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process. *Reliability Engineering & System Safety*, 185:372–382, 2019. See page 42.

[74] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. See page 42.

[75] Jiawei et al. Liu. Remaining useful life prediction of pemfc based on long short-term memory recurrent neural networks. *International Journal of Hydrogen Energy*, 44(11):5470–5480, 2019. See page 42.

[76] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. See page 42.

[77] Yonghui Wu and Schuster et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. See page 42.

[78] Ahmed Elsheikh, Soumaya Yacout, and Mohamed-Salah Ouali. Bidirectional handshaking lstm for remaining useful life prediction. *Neurocomputing*, 323:148–156, 2019. See page 44.

[79] Martín Abadi and Barham et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. See page 49.

[80] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009. See page 50.

[81] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005. See page 50.

[82] Yohwan Choi, Seunghyoung Ryu, Kyungnam Park, and Hongseok Kim. Machine learning-based lithium-ion battery capacity estimation exploiting multi-channel charging profiles. *IEEE Access*, 7:75143–75152, 2019. See page 56.

[83] Ji Wu, Chenbin Zhang, and Zonghai Chen. An online method for lithium-ion battery remaining useful life estimation using importance sampling and neural networks. *Applied energy*, 173:134–140, 2016. See page 56.

[84] Philipp Dechent, Alexander Epp, Dominik Jo¨st, Yuliya Preger, Peter M Attia, Weihan Li, and Dirk Uwe Sauer. Enpolite: Comparing lithium-ion cells across energy, power, lifetime, and temperature. *ACS Energy Letters*, 6:2351–2355, 2021. See page 57.