An innovative machine learning strategy for Lithium-Ion batteries Remaining Useful Life prediction and State of Health estimation

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Iacopo Marri**

Student ID: 10750914
Advisor: Prof. Francesco Amigoni
Co-advisors: Prof. Loredana Cristaldi
Academic Year: 2021-22

Extended summary

Lithium-Ion batteries have become the most widespread energy storing system, and are used as a power source for many applications. Industries, as well as private users, have many interests in optimizing the way lithium batteries are used. First of all safety: batteries behavior is very sensitive to the condition of usage, and it may happen that they overheat, or even burst if they are, for example, used under a voltage range not suitable for it, or at a degradation level beyond the fixed failure threshold; some Tesla Model S caught fire due to a short-circuit of the lithium-ion cells in 2013 [28]; an electric scooter blew up during charging in 2018 [16].
There are also financial reasons: lithium-ion batteries are very expensive components, and optimizing their use conditions helps to maximize their life spawn, and minimize costs. Prognostic measurements on the duration of the batteries and on their failure, help schedule the appropriate maintenance on time, optimizing the availability of the system. This thesis gives a contribution to this optimization process by providing one solution for estimating the State of Health (**SoH**) of the batteries, and one solution for predicting the Remaining Useful Life (**RUL**). The SoH is a value that measures the health of the battery by taking into account the remaining capacity, compared to the nominal capacity that the same battery had when it was new. Knowing with accuracy the SoH of a battery improves the accuracy in which other parameters, like the State of Charge (**SoC**), are measured. For instance, having a precise measure for the SoC allows the user to use the device in the proper range of charge, and helps to plug or to unplug in the correct way, optimizing safety and battery life.
From the industrial point of view, having an accurate prediction for RUL allows saving money by scheduling appropriate maintenance on time, reducing the downtime of the service, and reducing the risk of dangerous component failures which may endanger the assets of the industry.

In literature, many approaches have been used for SoH estimation and RUL prediction, like model-based or statistic approaches, but now in the big data era, data-driven and machine learning solutions are the most appealing and promising. In particular support vector machine and random forest algorithms, and recurrent neural networks gave the

most promising results.
In our thesis, we decided to use support vector regression (**SVR**) algorithm for the SoH estimation task, and an **LSTM neural network** for RUL prediction. The performance of our solutions has been validated on two publicly available datasets: one from the Prognostic Center of Excellence at NASA and another from a MIT-Toyota cooperation. The NASA dataset is small if compared to the MIT-Toyota, for this reason, we decided to use support vector machine (for regression, namely support vector regression), which is able to perform also with a few data, to work on the NASA dataset. The MIT-Toyota dataset is way bigger and allowed us to produce more sound results.

The first approach we had to RUL prediction, before to start developing our solutions, has been to replicate the work proposed in [3]. They proposed a hybrid approach that mixes a physic model component along with an echo state network (**ESN**), for predicting RUL using the NASA dataset. They start from a capacity degradation model called **Double Exponential**, from which they derived a semplified version called **Single Exponential**. They use this model to estimate the state space of the batteries up to a fixed prediction cycle, and from this state-space they generate another dataset, representing the degradation of the battery up to the prediction cycle k. The descriptor of this dataset is the number of cycles from 1 to k. Finally, they use this sequence to train the network and use the network to predict the rest of the degradation curve of the battery.
We've been able to state that this approach is not solid since the network is not able to learn the full degradation pattern of the battery, because in training phase are used only data about the first phase of the degradation, The approach could give more interesting results if more batteries are used to train the network, and their complete degradation profile is used.

The next step is to build a model with SVR algorithm for SoH estimation, and train it with the 3 batteries available in NASA dataset. In particular, is important to state that SoH normally can be measured with a method called Coulomb counting, but at one condition: the battery must go through a full charge + discharge cycle. This condition is not reliable in many real applications, for example electric vehicles. So we developed a feature-based approach that estimate the SoH by extracting features from a small portion of the charging curve, which is a condition that is more likely to happen in real cases. This is called partial charging curve approach.
We extracted the charging time from different voltage steps, in different voltage ranges. We picked the optimal voltage ranges and steps by picking the one with the lowest cross-validation error. The optimal values were 3.9-4 V range, and 0.5 V steps, so the final descriptors of the data were the charging time from 3.9 V to 3.95 V, and the charging

time from 3.95 V to 4 V, for every cycle. The validation R^2 is 0.95 which is an excellent result, but the solution has to be validated also on bigger datasets.

In the next step, we perform RUL prediction over the MIT-Toyota dataset, using a bidirectional LSTM neural network. We decided on this approach since RUL prediction is a task for which is important to take into account the past history of the sequence, to somehow infer how the battery is going to behave in the future, and from this, predict an appropriate RUL.
Despite the characteristic of neural networks of begin able to find meaningful features and internal representation, we decided to use our knowledge to extract features that are human interpretable. Starting from what [23] discovered, we adapted 3 of the features they elaborated so that they can be computed for any prediction cycle, and we can have a continuous RUL prediction from the beginning to the end of life of the batteries. The first two features for cycle k are the **variance** and the **minimum** of the curve obtained by subtracting the discharge curves of Q as a function of V, at cycles 10 (reference cycle) and cycle k. These two features have the property of representing on a linear space the input data, battery-wise. The third feature is the sum of the average cycle temperature, from cycle 2 to cycle k, that we call **temperature** feature. PCA analysis allowed us to eliminate the second feature, the minimum since it contained the same information of the variance feature.
We tuned the network with a 5-fold cross-validation and tested it over 10 samples obtaining excellent RMSE values for almost all of them. Exceptions are made by those batteries whose life is much longer than the average of the batteries in the dataset, since not enough data is present to describe them, and also the LSTM limitation about the length of the sequences that can be memorized must be taken into account.

The final step of the thesis has been to apply the SVR solution to estimate the SoH on Toyota-MIT dataset, that, unlike NASA dataset, has many different policies for the charging phase of the batteries. Those policies vary in the charging current and in other important charging details. This makes it impossible to adopt the same partial charging curve approach we had, so instead, we used the same feature we just described for RUL prediction, with the difference that we computed them considering only a small part of the discharge curve of Q(V), instead of the full curve. This is called partial discharge curve. In particular, we picked the interval from 2.8 V to 3.1 V. The fist important thing to note is that the variance feature, which in the case of the full discharge curve generates a linear pattern, with partial discharge curve it loses most of its information, and it is almost totally noise. On the other end, the minimum feature keeps almost entirely its information. So in the final SVR model, we used two features: the minimum feature and the temperature feature. We again validated the models with a 5-fold cross-validation. The results show an R^2 value of 0.97 which is beyond our expectations. Some of the samples present noise in the estimation, which is due to the noisy nature of the features used.

Anyway, the partial discharge curve assumption is more strong than the partial charge curve, in fact for the majority of the real applications, an approach based on the charging curve would be preferred. Unfortunately, no available dataset at the moment is big enough, and with the right charging pattern, to allow us to properly test the solution on the partial charging curve.
What can be done in the future to improve this work, is to further reduce the discharge interval to extract the feature, and explore different voltage ranges to make the assumption as weak as possible.
Regarding RUL prediction, a possible future development is to adopt a sliding window approach to crop the longest sequences to smallest ones, and overcome the memory limitations of LSTM Network. Another approach to be carried out, in order to fully exploit the power of Neural networks, is to quit the feature-based approach and feed the network with sequences of raw measurements instead.

Abstract

Nowadays, batteries, and in particular Lithium-Ion batteries, have become enormously used in many systems and applications, and are absolutely the most widespread energy storing system. Optimizing the usage of batteries thus is very important to increase the safety of systems like electric vehicles or portable devices, to reduce economic loss in industrial environments, and to increase the availability of such environment services. An accurate Remaining Useful Life (**RUL**) prediction is key for this optimization process to succeed since it allows us to know battery conditions, and to schedule proper maintenance. At the same time, State of Health (**SoH**) estimation is important to improve the accuracy of other diagnostic measures, like State of Charge (**SoC**).
In this thesis work we propose one approach for SoH estimation, based on Support Vector Regression machine learning algorithm and a smart feature extraction process, finding a good trade-off between applicability, light computation effort, and accuracy of results.
Performances are measured on 2 different datasets, one from the Prognostics Center of Excellence at NASA and another from a MIT-Toyota cooperation. Another approach based on LSTM Recurrent Neural Network is proposed to perform RUL prediction, and its performances are evaluated on the MIT-Toyota dataset.

Keywords: RUL, SoH, aging prediction, neural networks, machine learning, lithium-ion batteries, degradation prognostic.

Abstract in lingua italiana

Oggigiorno, le batterie, ed in particolare le batterie agli ioni di litio, vengono enormemente usate in molti sistemi ed applicazioni, e sono in assoluto il più diffuso sistema di stoccaggio energetico. Ottimizzare l'utilizzo di tali batterie è molto importante per poter aumentare la sicurezza dei sistemi che le adottano, come veicoli elettrici o dispositivi portatili di varia natura, oltre che per ridurre perdite in termini economici nei sistemi industriali ed aumentare la disponibilità dei servizi che tali sistemi offrono. Una accurata predizione della Vita Utile Rimanente (**RUL**) delle batterie è la chiave perchè questo processo di ottimizzazione possa avere successo, perchè tale predizione ci permette di conoscere al meglio le attuali condizioni delle batterie in uso, e programmare un'adeguata manutenzione. Allo stesso tempo, la stima dello stato di salute (**SoH**) delle batterie è importante per migliorare l'accuratezza di altre misure diagnostiche, come ad esempio lo stato di carica (**SoC**).
Questo lavoro di tesi propone un approccio alla stima dello stato di salute, basata sull' algoritmo di machine learning della Regressione a Vettori di Supporto (**SVR**) e su un particolare processo di estrazione delle feature, cercando di stabilire un buon equilibrio tra applicabilità del metodo, leggerezza del carico computazionale e accuratezza dei risultati. Le performance di questo approccio sono misurate su 2 dataset, uno proveniente dal Prognostic Center of Excellence alla NASA e un altro da una cooperazione tra MIT e Toyota. Un altro approccio, basato su una rete neurale ricorrente a moduli LSTM è proposto per effettuare la predizione del RUL, e le performance di questo approccio sono valutate sul dataset MIT-Toyota.

Parole chiave: Vita utile rimanente, Stato di salute, predizione dell'invecchiamento, reti neurali, machine learning, batterie agli ioni di litio, prognostica della degradazione.

Contents

1 | Introduction

Today's world is moving toward a future that is more and more dependent on energy storage systems. We all have multiple battery-powered devices like smartphones, notebooks, tablets, and headphones, and in about 10 years we're all going to drive only battery-powered vehicles. Many industrial fields rely on batteries as energy storage systems too, as well as many critical applications like aircraft and satellites [11]. The global transition to clean and renewable energy heavily depends on our ability to store that energy, which is also the bottleneck of the transition process.

Lithium-Ion batteries are the most used batteries because of their very high density of potential and minor self-discharge, along with a very long lifetime, thus there is a massive demand for them [2].

The high need for batteries and their high cost highlight the need of a Battery Management System (**BMS**) which is capable of monitoring the degradation of the battery with high enough accuracy in order to optimize the battery usage and its cost. In particular is of critical importance to have a BMS which evaluates with precision when failures can happen, in other words, a BMS is able to compute the **State of Health** and **Remaining Useful Life** of batteries early enough to avoid the use of the batteries under dangerous degradation conditions, schedule appropriate maintenance, and therefore avoid unsafe situations, maximize the availability as well as the lifetime of the batteries, and reduce costs. The BMS is mounted on battery-equipped systems, and uses sensors to take measurements during the usage of batteries, like surface **temperature**, **current**, **voltage**. Those measurements are however rough compared to the many micro phenomena that take place inside the battery, and whose entanglement causes a strongly non-linear and hardly mouldable degradation [14]. Those phenomena are impossible to measure in real-life applications, and interact with each other, resulting in various and different aging and capacity degradation shapes [1], thus we need solutions that can model degradation by using BMS measurements.

The objective of this thesis is to employ data-driven and machine learning approaches as possible solutions to RUL and SoH estimation problems, in particular, **Support Vector Machine** algorithm is used to perform SoH estimation, since it is lighter and faster in computation, and could be potentially used on every portable device, increasing the precision of parameters like the SoC, that is the % of battery remaining, leading to a more comfortable and optimized usage of the device, while **Recurrent Neural Network** is used for RUL prediction, which unavoidably needs for the sequential structure of the past and the future conditions of the battery to be taken into account.

The results of SoH estimation show the validity of the proposed approach, and in particular, of the proposed features. Performances reach a good accuracy along with a fast fitting and prediction time, highlighting the potential applicability in real-time scenarios. RUL prediction solution shows the suitability of Recurrent Neural Networks to battery degradation time series applications. In fact, we obtained good prognostic measurements from the very first moments of the time series, especially for those data sequences that lie near the mode of the dataset. For longer-lasting batteries, on the other hand, which are the great minority of the data, predictions are a bit noisier, yet fairly good, due to the data distribution. This shows the importance of finding new datasets, more balanced and possibly richer, on which to further validate our solutions.

This thesis is organized as follows. Chapter 2 introduces the concepts of State of Health and Remaining Useful Life and delves into why we need good estimators and predictors for them. Chapter 3 describes different methods SoH estimation and RUL prediction are approached, discussing, in particular, the advantages of machine learning. Chapter 4 explains more in detail the algorithms and models we are using in this thesis. In chapter 5 we explain and discuss the datasets that we are using. Chapter 6 is about the hybrid approach presented in [3], how we replicated it, and the results we obtained. In chapter 7 we present our solutions for SoH estimation and for RUL prediction. Finally, in chapter 8 we draw conclusions and discuss possible future work and developments.

2 | SoH and RUL for batteries

If we take into consideration the most general case of a system, the study of its health or that of some of its components revolves around the computation of some figure of merit of the condition of the given system or component.
Our approach towards the study of the health of batteries is aimed at computing two quantities, characterizing the condition of the batteries, which are *State of Health* (**SoH**) and *Remaining Useful Life* (**RUL**). In particular, in this thesis we investigate and apply different *Machine Learning* techniques for making **Estimation** of State of Health, and **Prediction** of Remaining Useful Life of Lithium Ion Batteries.

2.1. State of Health

The State of Health of a battery is a percentage value, defining the actual health condition and performance, in relation to the initial ones, when the battery was brand new and at its maximum.
There's not a unique mathematical definition of SoH, it may change according to the application field, but primarily it can be defined in two ways: in terms of the increasing internal resistance, in this case, a loss in health condition is described by a loss in potential current output. Or, it can be defined in terms of decreasing capacity: in this case the health condition is described by the total charge that the battery can store [30].

SoH changes as a function of the *Number of Cycle* k, which is the number of charging + discharging cycles the battery has gone through, since the beginning of its life.

In our case, the SoH at time (intended as cycle) k, is defined in terms of the maximum capacity, as follows:

$$\boldsymbol{SoH_k = \frac{C_k}{C} \cdot 100} \tag{2.1}$$

$$\tag{2.2}$$

Where C_k is the capacity of the battery at cycle k, and C is the *Nominal Capacity*, namely

the capacity that the battery had at the beginning. From now on we will refer to SoH as in this formulation.
SoH is then a quantity ranging from 0 to 100, indicating in fact the percentage of residual health. Figure 2.1 shows the capacity fade over cycles of one sample battery, along with the relative SoH degradation.

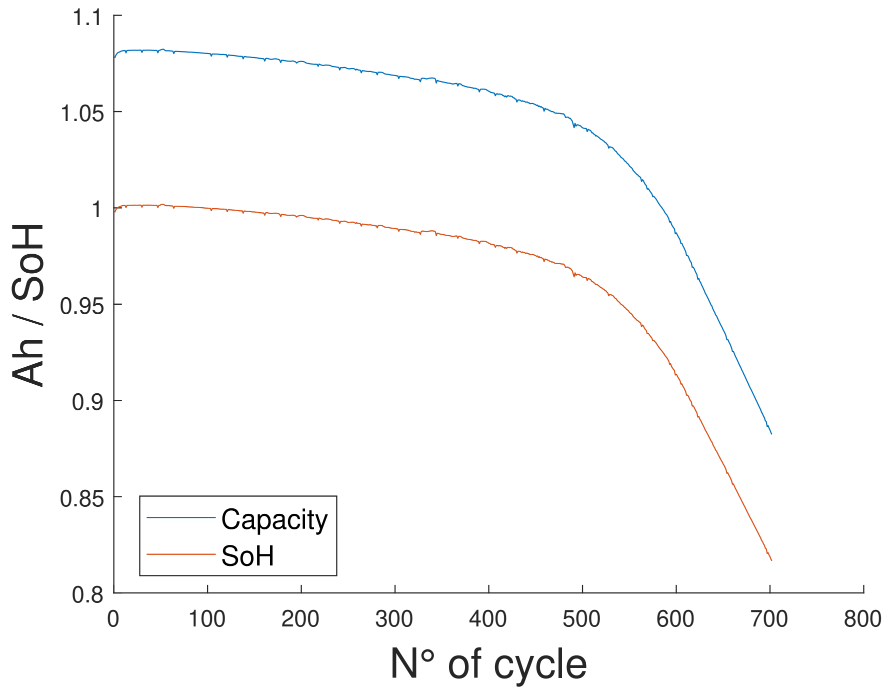


Figure 2.1: Capacity and State of Health of 1 sample battery.

SoH is an important parameter in deciding how reliable a battery is, and also influences other important parameters like State of Charge (**SoC**), whose measurement may become erroneous as the health of a battery decreases. An accurate SoH estimation translates in an accurate SoC, which allows the user of the device to follow the proper discharge-charge pattern: for instance to prevent the real SoC of the device from dropping too low, or for suddenly shut down before even reaching 0%.

2.1.1. SoH Measuring and Estimation

There are many methods for obtaining SoH, like *Coulomb Counting Integration*, which under the appropriate conditions are quite effective; but doing it in an accurate way is not an easy task.
SoH is an internal property of the battery which cannot be directly measured, instead we "compute" it starting from other sensor measurements like voltage, current, and in

our case charge. SoH has a strong variability, and it's strongly affected by environmental conditions like temperature, but also by smaller internal parameters combinations given by electrochemical reactions (like electrolyte decomposition and SEI film generation, acting on electrodes) in the same way we described for the battery degradation.

In **Coulomb Counting Method** which may be also known as the *Current Integration Method*, the discharge (or charge) current I is integrated in time over the whole cycle, to obtain a measure of the energy which was contained inside the battery (or that is being introduced into).

To apply the existing methods, the battery has to go through a complete **charge + discharge cycle** in order to measure the total current that it is capable to store and then provide, and in addition, for those measurements to be accurate, the temperature of the surrounding environment and many other conditions have to be controlled and fixed.
Such conditions are **not always reproducible**, depending on the context. As an example, consider measuring the SoH of the battery pack of an electric vehicle: temperature is not controllable, and performing a full cycle of the battery is money and time-consuming.

What has just been said is so important in understanding the reasons for the need of a SoH estimation system, and why research puts effort in the study of methods and techniques capable to *estimate* the SoH of batteries.

2.2. Remaining Useful Life

Remaining Useful Life is a parameter that applies to any kind of system and indicates the time instants that remain, from a given prediction time k, up to the End of Life (**EoL**) of the system. The correct evaluation of the system's RUL has a strong impact on its maintenance, since a precise RUL allows to schedule a proper substitution/maintenance of the system or of its components, and therefore to mitigate or prevent critical failures. This reflects directly into major safety, optimized resource-saving, and improved availability of the system.

RUL prediction is a complicated task since it is not only a function of the past and present condition, as it is SoH, but also on the future operational profile that the system is subject to. This means that information on the future usage of the system has to be known or at least inferred from the patterns in the history of the system. Of course, there will always be uncertainty in the future usage of the system, thus there will always be an essential uncertainty in the RUL prediction, which transcend our capacity to forecast.

For all these reasons, RUL forecasting accuracy has the intrinsic behavior of improving the more the prediction time k is moved forward, and the more information on its past we have. Since for maintenance purposes we are mostly interested in having good predictions close to the failure point, lower accuracy in the first-time instants is accepted.

RUL prediction research is active in various fields and applied to several systems, from aircraft engines to MOSFETs, any kind of industrial assets, and batteries.

2.2.1. RUL for batteries

Battery RUL from a cycle k is defined as the number of cycles in which the battery reaches the EoL threshold, from cycle k.
The EoL of a battery is conventionally fixed at 80% of its SoH, but depending on the application we're considering, this threshold may vary: more critical applications might require a SoH not lower than 90%, while for more relaxed ones, 60% SoH is also accepted. This is because, for many applications, there is not an actual battery failure threshold, but a point after which the usage of a battery is no longer profitable.

$$\boldsymbol{RUL_k = Cycle_{EOL} - k} \tag{2.3}$$

$$\tag{2.4}$$

Where k is the cycle at which we make the prediction, and $Cycle_{eol}$ is the cycle at which EoL is reached.

2.2.2. RUL and SoH connection

Battery RUL and SoH are tied together, indeed many approaches to compute the battery RUL, as first step compute the whole battery SoH curve (also called degradation curve), in order to find out at which cycle SoH reaches 80%;
So we can state that SoH and RUL are described by very similar physical properties, but when it comes to computing them, there are different difficulties to face, which will be explained more in detail later on.

3 | Approaches for estimating SoH and RUL

In literature, we find lots of different studies and approaches to deal with SoH and RUL, and those can be divided into three macro-categories: **Model-Based**, **Data-Driven**, and **Hybrid** [30].

3.1. Model-based

Model-based solutions are approaches that build mathematical models from the physical knowledge of the batteries (sometimes are referred to as Physic-Based approaches). Those models try to shape the internal phenomena of the battery in the most precise way possible, in order to approximate its degradation and decay. These techniques have been deeply studied and developed, but the big number of internal actions and reactions that happen and interact inside batteries makes these approaches really hard to develop and refine. In addition, such solutions are very battery specific, and also condition-specific, since a change in any physical factor would destabilize the model; therefore model-based solutions are not able to generalize.
Even though this approach has strong theoretical foundations and results, it is not applicable in real situations for the reasons listed above, and because some needed measurements would require to literally open up the battery, making it unusable.
Two subcategories can be defined:

- **Electrochemical Models**, which rely on the knowledge of the internal structure and behaviour of battery.
- **Equivalent Circuit Method**, applying electrotechnics, and considering the battery as a blackbox.

Some examples of electrochemical models are [25], where the authors studied the dissolution of the lithium salt close to the SEI of the negative electrode, obtaining a connection with the fade of capacity, and [6], where a solvent oxidation reaction was considered; here in [27], they used an equivalent circuit approach to select the first-order Thevenin model.

3.2. Data-driven

Data-driven methods have taken hold over the model-based approach in the last years, and a lot of research has been made and still there's to do about it. The data-driven approach allows to partially transcend from the knowledge of the battery domain, do not model internal reactions thus are generally easier to develop. They rely instead on a huge amount of data from which to model the degradation behaviour.
These methods are more flexible and adaptable, so more able to generalize, and more adapt to real usage, since less modeling and less tough measurements are required.
Nevertheless, the amount of data that is required is quite a drawback, which makes the approach heavily dependent on the availability of good datasets. Big data era, data mining, and data analysis fast evolution are pushing data-driven methods ahead, but still, battery-related data are slow and costly to collect, so there are still few datasets publicly available, that are suitable to apply for battery degradation studies.
Using a data-driven approach also implies the handling of the data, cleaning and manipulating them, understanding and rendering, which are delicate phases that condition the results.

According to various papers, [30], [26] data-driven methods can be divided into various subcategories, which are mainly resumed in:

- **Filtering methods**: Are generally used to elaborate signal containing noise, computing the probability distributions of data. Examples are **Kalman Filter** (**KF**), and its variations **Unscented Kalman Filter** (**UKF**) and **Extended Kalman Filter** (**EKF**), used respectively by [32] for RUL prognosis, and [29], that proposed an EKF combined with genetic algorithm, and **Particle Filtering** (**PF** or **UPF**) [10].
  Filtering methods have shown good results in the prognostic field, but have some difficulties in handling non-linear data; for this the *extended* versions of the algorithms were developed, slightly improving on this front.
  Furthermore, filtering methods computational complexity does not scale well with the quantity of data.
  Since battery degradation pattern is strongly non linear, KF and PF alone are not enough, and are often used in hybrid configurations.

- **Statistical methods**: Like **Wiener Process** (**WP**) and **Gaussian Process** (**GP**), applied to the regression problem of battery degradation, estimate a new step of the degradation pattern using statistical data [12, 15].
- **Machine Learning** Supervised machine learning techniques are able to learn from the data the aging patterns of batteries, and can be used for prediction and estimation. Many ML algorithms have the advantage of handling non-linear patterns, which is key in the case of battery aging, and also perform easier calculations with respect to other approaches, leading to more scalable models, and being applicable in more situations.

  In literature, many ML algorithms have been applied and tested on degradation estimation problems, on various kinds of datasets coming from different scenarios, different batteries, conditions and so on. For instance [13] used **Support Vector Machine** (**SVM**) to estimate values for capacity and resistance for electric vehicles batteries, [4] applied **Random Forest** (**RF**) with a feature-based approach, to estimate the SoH from measurements of a small portion of the charging curve of batteries. [24] used simple and multiple **Linear Regression** along with a feature-based approach, on a dataset of over 100 aged batteries. [7] instead used Artificial Neural Networks to predict the SoH, using past values of internal resistance and capacity.

  Overall, we can group the most used ML algorithms so far:

  - **Support Vector Machine**: It is one of the most classical and used ML algorithm, successfully applied in countless fields. *Support Vector Regression* (SVR), which is the version of SVM adapted to perform regression, has the advantage of working pretty well also with few data with respect to other models, in particular, the choice of the kernel allows to handle the dimensionality of your data, i.e. if you have few or many features, with respect to the data points. In addition it is quiet robust to noise and doesn't suffer from local minima issue, which instead affects Artificial Neural Networks. In [5], a LS-SVM model that is based on arbitrary entropy is created to estimate the SoH.
  - **Artificial Neural Networks**: The main advantages are that ANNs are good in catching non-linear patterns, and they generally don't need human knowledge about the context, to work. ANNs can be adapted to the complexity of che context just by adding neurons, or layers of neurons (in this case we talk about **Deep Neural Networks**). They have the capability of finding relations among the variables, without the need of knowledge of the context.

One variation of the ANNs, is the **Recurrent Neural Network**, which, thanks to recurrent connections, acquires the capability of memorizing relations with previous inputs, and is able to deal with the concept of *sequences* or *time series*. An application is [22], where maximum available capacity is used to show the SoH based on a back propagation (BP) neural network. In [20] the battery SoH is estimated given a time series of capacity over cycles. Qu proposed in [19] a prediction model for SoH through LSTM, and predicts SoH based on a moving window.

- **Random Forest**: It is an ensembling technique, namely, a technique that puts together more weak learners (decision trees) to "build" a stronger learner. The main advantage is that is quite easy to build and set up, and gives in output also a ranking of features importance. Last but not least, it is easily adjustable to face overfitting by changing the number of decision trees used.

3.3. Hybrid Models

Is a very general category of techniques, which fuses together different techniques, usually model-based and data-driven, to exploit the strengths of both. In [3], a hybrid solution of a physics-based degradation model, along with an Echo State Network (**ESN**) has been applied to RUL prediction. Zhao et al. [31] proposed a hybrid method based on an empirical degradation method and a data-driven method. They decoupled the degradation process into two types: capacity regeneration and normal degradation. Then, they predicted the battery State of Health by using the relevance vector machine (RVM), which is a technique similar to the support vector machine but uses Bayesian inference to obtain solutions for regression or probabilistic classification.

4 | Adopted models

4.1. Support Vector Machine

Support Vector Machine is one of the most classic and used ML supervised learning models, which performs binary classification. Given a set of samples, SVM finds the separating hyperplane with the **biggest distance** from each training point, so not only tries to classify the points but also to find the best and most robust possible hyperplane to do so. The problem is that in many cases, the points are not linearly separable in the input space, and a higher (even infinite) dimensional feature space is needed to find a linear hyperplane which can divide them. The **Kernel Trick** allows to move the data from the input space to a higher-dimensional space, keeping feasible the computational burden of the dot products between arrays of points: the dot product between 2 points in a N dimensional space where N is arbitrarily big, can become computationally unfeasible. Thanks to the kernel function, the dot products are no longer computed but are represented in terms of a **Kernel Function** $K(x, y)$. In other words, the kernel method represents the data only through a set of pairwise similarity comparisons between the original data observations, which are lower in dimension, instead of explicitly applying the transformations to a higher dimension.

The separating hyperplane will be defined in terms of the feature vector of data points, as a linear combination with parameters α_i. Those parameters are different from 0 only for those data points (called support vectors) which lie nearest to the hyperplane, and in fact define it. So the separating boundary is defined only as a function of the support vectors. [8] discuss the theory and applications of SVM.

4.1.1. Support Vector Regression

SVR is a version of SVM adapted to perform regression tasks. Compared to normal linear regression which finds the minimum of the squared error for the target values, SVR just fits the error of its predictions within a limit ϵ, while minimizing the L2 loss:

$$\begin{cases} min\,\dfrac{1}{2}\beta'\beta \\ |Y_n - (X'_n\beta + b)| \leq \epsilon \ \ \forall n \end{cases} \tag{4.1}$$

where β and β' are the weights array respectively normal, and transposed, Y_n are the target values, X'_n is the transposed descriptor array, and b is the bias, and ϵ is the maximum allowed divergence between the target value and predicted value. Given that is not always possible to satisfy the ϵ constraint, it is relaxed by adding what are called the **Slack Variables**. Any error greater then ϵ is now accepted but is considered in the loss function with a weight. This approach is called **Soft Margin**. The equations stated before become:

$$\begin{cases} min\,\dfrac{1}{2}\beta'\beta + C\displaystyle\sum_{n=1}^{N}(\xi_n + \xi_n^*) \\ Y_n - (X'_n\beta + b) \leq \epsilon + \xi_n \ \ \forall n \\ (X'_n\beta + b) - Y_n \leq \epsilon + \xi_n^* \ \ \forall n \end{cases} \tag{4.2}$$

where ξ_n and ξ_n^* are the slack variables for each point, and C represents the weight associated with them: $C = 0$ means that slack variables are not considered, and no error is allowed, while a high value of C leads to a more relaxed problem. Figure 4.1 from [18] gives a better visual understanding of the problem and the parameters.

The loss function at this point considers only the errors which are greater then ϵ:

$$Loss_\epsilon = \begin{cases} 0 & \text{if } |Y - f(x)| < \epsilon \\ |Y - f(x)| - \epsilon & \text{otherwise} \end{cases} \tag{4.3}$$

Now, if we write the optimization problem, the loss function and the constraints, in its dual Lagrange form, the problem is more easily solvable and the parameters β of the model, as well as the predicted value for new observations, become a function of the training samples, according to this formulation:

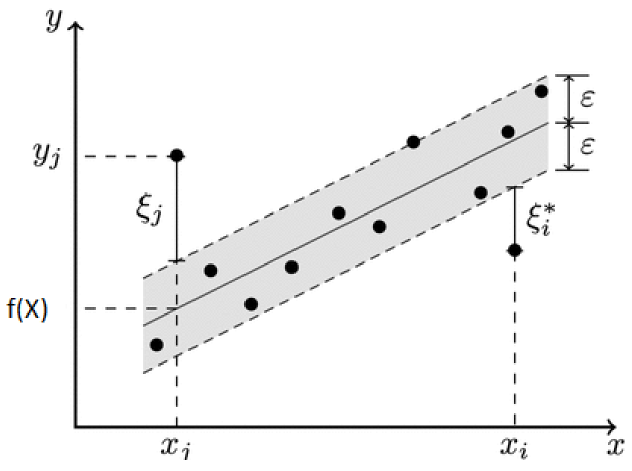


Figure 4.1: SVR epsilon-tube around f(x), and slack variables.

$$\begin{cases} \beta = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) X_n \\ f(X) = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*)(X_n' X) + b \end{cases} \tag{4.4}$$

where α_n and α_n^* are the Lagrangian multipliers, and if a sample has either α_n or α_n^* different from 0, then it is a support vector, and are those points which lie outside the ϵ-tube. This particular behaviour is granted by the **KKT Conditions**, which are optimization constraints required to obtain optimal solutions. In particular they indicate that all observations that is strictly inside the epsilon tube must have Lagrange multipliers $\alpha_n = 0$ and $\alpha_n^* = 0$. It follows that any point having either α_n or α_n^* equal to 0, lays outside the boundaries, and becomes a support vector.

As for SVM, in case the problem is not linearly solvable, Kernel functions are used to map the data in higher feature space where they are more likely linear. In this case, the dot product $(X_n' X)$ in the prediction formula is changed into $G(X_n, X)$, where G is the **Gram Matrix** containing the kernel values computed for all the couples of points.

4.2. Artificial Neural Networks

Artificial neural networks (ANN), a sub-branch of the Machine Learning field, are computing models which are inspired by the internal structure of the brain and neuron connections.

4.2.1. ANNs Structure

Artificial Neural Networks are composed by units called *neurons*, distributed in structures called *layers*, which are interconnected among each other, through links (or edges), which simulates the brain mechanisms of propagating signals through neurons.
Each edge is associated with a **weight W**, which is multiplied by the signal passing through it and is tuned during the training phase. Each neuron has in input one edge from every neuron of the previous layer, plus another single value called **bias**, and its job is to take the linear combination of all input values and apply the **activation function** to it.
The activation function generates the output of the neuron which is propagated to the next layer of neurons; using a non-linear activation function allows the network approximate non-linear patterns in the input data. Typical activation functions are *sigmoid, hyperbolic tangent, rectified linear unit.*

$$\begin{cases} g(a) = a \\ g'(a) = 1 \end{cases} \tag{4.5}$$

Equation (4.5) is linear activation, codomain is $\mathbb{R}$.

$$\begin{cases} g(a) = \dfrac{1}{1 + \exp -a} \\ g'(a) = g(a)(1 - g(a) \end{cases} \tag{4.6}$$

Equation (4.6) is sigmoid activation function, codomain is $[0, 1]$

$$\begin{cases} g(a) = \dfrac{\exp a - \exp -a}{\exp a + \exp -a} \\ g'(a) = 1 - g(a)^2 \end{cases} \tag{4.7}$$

Equation (4.7) is hyperbolic tangent activation function, codomain is $[-1, 1]$

The input of the network travels across the layers, converging in the final layer, which contains a number of neurons equal to the number of desired outputs. Figure 4.2 from [17], shows the structure of a (deep feedforward) Artificial Neural Network, and the structure of a neuron.

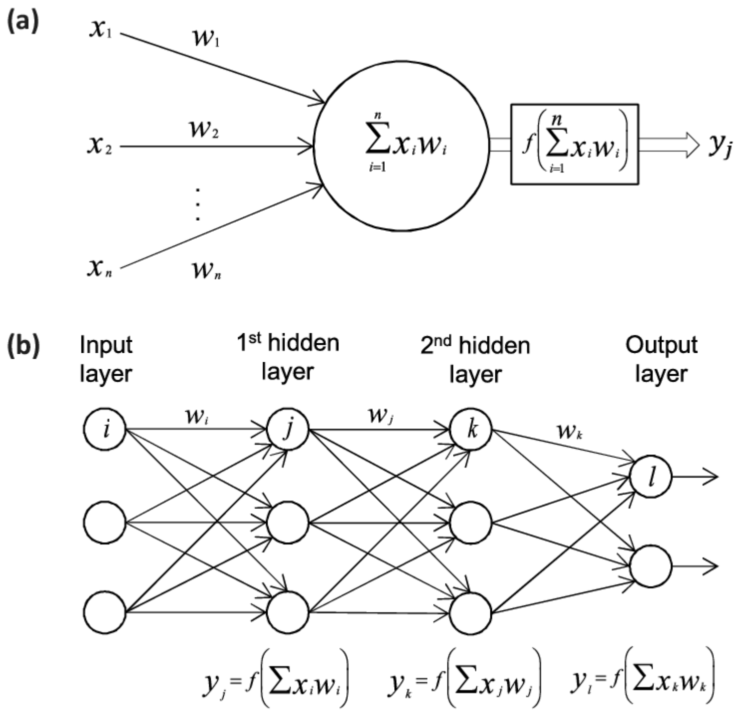


Figure 4.2: a. Structure of a neuron, input arcs, signals combinations, and activation function - b. Layers structure of an ANN.

4.2.2. ANNs training

According to the *supervised learning* paradigm, training samples are provided to the network in the form of *input* and known *result*, so that the model can learn to reproduce patterns, make decisions, take an action, without being specifically programmed for that. Other learning paradigms exists, but we will focus on supervised.

In neural networks the learning mechanism is implemented by minimizing a *loss function* that measures the error between the predicted output and the real output of the training samples. Examples of loss functions are *cross entropy* (binary or categorical), or *Mean Squared Error (MSE)*.
Loss functions are functions of the weights of the model, and for minimizing them in

ANNs usually **(Stochastic) Gradient Descent** is used: this method takes small steps following the opposite direction of the gradient of the loss function with respect to the weights, in order to converge toward the minimum. It follows that the loss function is required to be fully continuous and differentiable. The gradient is computed in an efficient way with **backpropagation**: it exploits the *chain rule* by expressing the gradient of the loss function as different gradients for each weight, computing them step by step, starting from the last layer, back up to the first.

Backpropagation together with gradient descent allow to efficiently train high dimensional models, with a drawback: the method doesn't find the global minimum, like a direct computation of the entire gradient would do, but moves toward local minima and may even get stuck in them.

The training phase involves multiple iterations of error computation and weight adjustment, over the training set. Every time the entire training set is fed to the network, we call it an *epoch*.
The number of epochs determines how long the training lasts. Generally we want it to stop when there's not significant improvements in the validation error, to avoid *overfitting*.

4.2.3. Overfitting

It is a phenomena which easily afflicts Neural Networks due to their complexity, and happens either if the model has been trained too much, or it is too complex (namely: with too many weights to train). It literally means that the model is learning too well the pattern of the training data, even their noise and fluctuations, and approximates them with a very low error. As a consequence, the model loses in *generalization* capacity, and will perform poorly on new data. Figure 4.3 shows the difference between a model that overfits, underfits, or fits well the data.

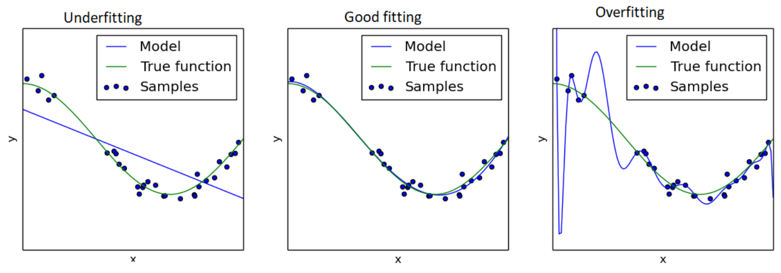


Figure 4.3: a. Structure of a neuron, input arcs, signals combinations, and activation function - b. Layers structure of an ANN.

A more wide understanding of overfitting is given by the study of the **Bias-Variance Tradeoff**: models parameters have a measure of *bias*, which states the difference between predicted and real values, and *variance*, that is how inconsistent is the model in predicting over different datasets. We can understand that, ideally, we want our model to have low bias and low variance at the same time, but ML theory says that bias and variance are tied by an inverse proportionality: you can reduce bias by increasing variance and vice-versa. Back to our topic, we can relate an high variance to overfitting, and an high bias to underfitting. Finding a good model for our context means to find the good trade-off between variance/overfitting and bias/underfitting.

4.2.4. ANNs hyperparameters and cross-validation

In order to develop and optimize a neural network some parameters called **hyperparameters** have to be tuned in order to best fit the data to the working context.
Unlike weights, hyperparameters do not train automatically with epochs iteration, instead the developer does the selection. Examples are:

- **Neurons:** The number of initial and final neurons has to be decided in function of the input and output dimension to match the shape of the problem. The number of neurons in the hidden layers can be used to introduce complexity in the model, if it is required from data.
- **Layers:** As well as neurons, adding more hidden layers (middle layers between input and output layers) increases the complexity of the network, adding weights to train, and thus dimensions in the function.
- **Learning rate:** Gradient descent algorithm takes steps towards the minimum error, proportional to the gradient of the function. Learning rate is an adjustable parameter which is multiplied to the step size and allows to control the speed of convergence to the local minimum. A high learning rate reduces the training time at the price of some accuracy.

Since the training loss only includes the error given by the bias of the model, we need to compute the results on another independent set called *validation set*, to evaluate also the error given by the model variance.
Running trainings with different values for the hyperparameters, finding those with the optimal validation error is the way to tune the model.

Depending on the dataset variance and how the validation set is selected (usually randomly), it may not reflect the real distribution of the dataset, and the validation error

can be biased and misleading.
To mitigate this problem, is good habit to use **Cross Validation** methods to have an unbiased error estimation: run multiple training and validation error evaluation sessions, changing the validation set every time. After that, the global validation error is computed by the mean of the obtained errors. Most used strategies are:

- **KFold**: K equal-sized sets are selected. For each, one model is trained on the remaining $K - 1$ sets, and evaluated on the Kth. The final error is computed on the K measures obtained.
- **Leave One Out**: N models are trained, where N is the number of samples in the training sets. Each model is trained on $N - 1$ elements, and evaluated on the remaining one.

4.2.5. Recurrent Neural Networks

RNNs are a type of neural networks in which recurrent connections are built, such that a neuron at timestep $t + 1$, gets in input the new data, and in addition the previous output of that same neuron at time t.
This mechanism empowers the network to produce outputs based on outputs at previous steps, so it has some sort of virtual memory, and the capacity of handling and forecasting *temporal sequences* and not only unrelated data points. This is a strong property which is perfectly suitable for predicting the degradation pattern of batteries and for this reason in our approach we used an **LSTM** network, a particular form of a Recurrent Neural Network.

4.2.6. LSTM

Long Short-Term Memory was initially developed to deal with the *vanishing gradient* problem, which affects RNNs, by vanishing or exploding the weight upgrade steps if the gradient was initially lower or grater then zero.
LSTM structure is composed of modules which can be stacked both vertically or horizontally, as shown in Figure 4.4. Each module controls 3 gates: **input**, **output** and **forget** gates. Modules have an element called **cell** which contains the information and makes it flow through the network. This flow is controlled by the gates, which modify the content of the cell and the information flow by the weights that the gates have associated.
Information contained in the network is also elaborated through activation functions, and moved to following layers up to the last, like in normal feed forward networks.

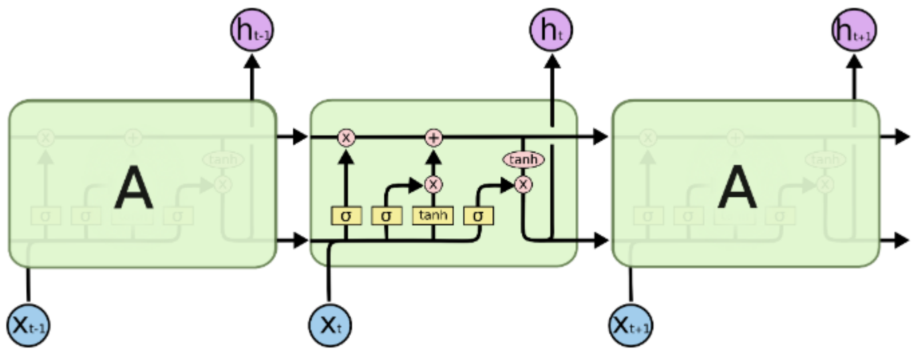


Figure 4.4: Horizontally stacked LSTM layers. X_t is the input and H_t is the output at timestep t.
Source: Image from Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The cell makes the information flow from the previous to the next module without applying any activation functions. It just keeps the value going from one module to the next one, as showed in Figure 4.5.

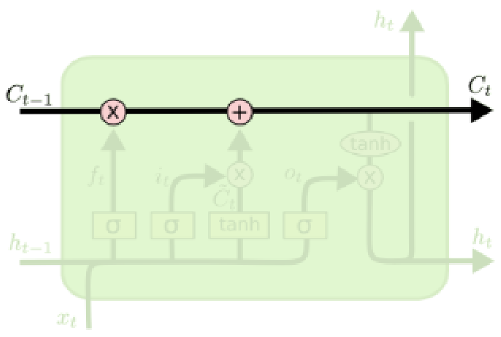


Figure 4.5: Cell of an LSTM module.
Source: Image from Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The **input** gate uses an hyperbolic tangent layer to generate a new value $\overline{C_t}$ to be added to the cell value. $\overline{C_t}$ is weighted by i_t generated by a sigmoid layer; Figure 4.6.

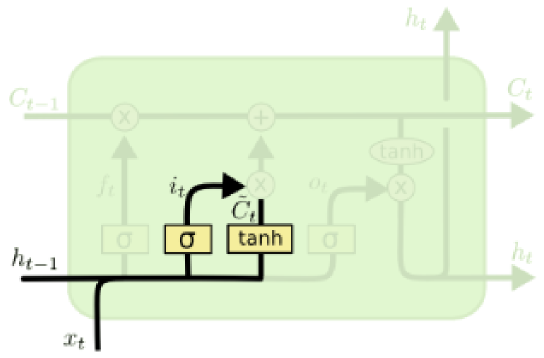


Figure 4.6: Input gate of an LSTM module, composed by a sigmoid and a tanh layer
Source: Image from Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The **forget** gate in Figure 4.7 decides how much of the current cell value we want to keep and how much to forget. This is done with a sigmoid layer whose output is multiplied to the cell state.

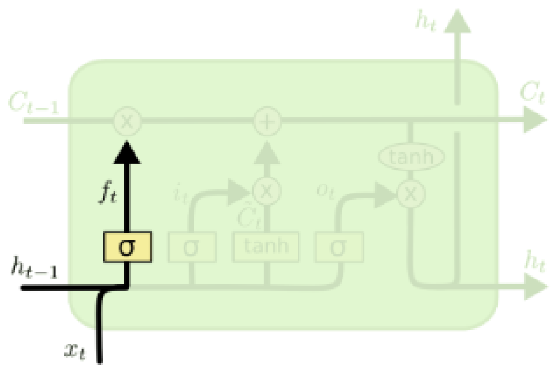


Figure 4.7: Forget gate of an LSTM module.
Source: Image from Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The **output** gate filter the cell state with tanh layers, and with sigmoid weights how much of that is going to be the output of the module; Figure 4.8.

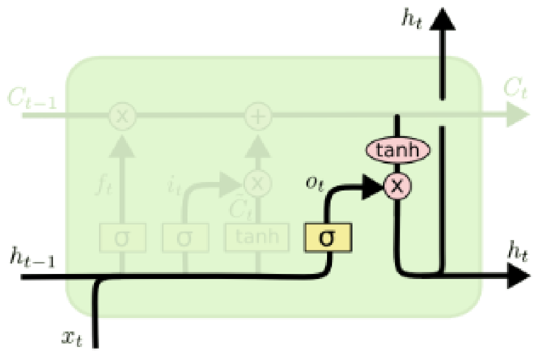


Figure 4.8: Output gate of an LSTM module.
Source: Image from Christopher Olah, Understanding LSTM Networks, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

5 | Datasets

Data-driven techniques strongly rely on data to operate. For successfully applying Machine Learning techniques to prognostic modeling, having quality dataset is essential.
Building a dataset of battery measurements which is suitable for prognostic applications is not an easy task: a *Data Acquisition System* is required, which runs charge and discharge tests over multiple batteries, and uses sensors to measure diagnostic values like voltage or temperature, at a fixed time interval. This process could take months to be completed, and in order to have the best possible quality of data, the environment conditions have to be kept stable, especially the chamber temperature.
Different patterns can be used to test the battery. Typically the test cycles are divided in two types: *aging cycles* and *capacity measurement cycles*, since fixed conditions are required for measuring the capacity, and aging pattern is not guaranteed to be so. The aging phase can be composed of charging and discharging cycles, or built following more particular patterns like *electric vehicle* driving simulations. Still, in some cases aging cycles and capacity cycles can coincide.
In a dataset specifically built for degradation prognostic purposes, batteries have to be tested to failure, so to their End of Life.

Due to the complicated setup that a Data Acquisition System requires, and the time required to test the batteries, not so many datasets are available and suitable for prognostic tasks, even though, in the last years the interest in battery prognostic is increasing, and so does the number of companies that research on it and provide quality data.

5.1. NASA dataset

NASA Ames Prognostics Center of Excellence (PCoE) published a data repository composed of 6 datasets of aged Li-Ion batteries [21]. According to their guidelines, only the first of these datasets is suitable for prognostic degradation prediction.

5.1.1. Properties

The tests have been carried out in a climatic chamber to keep the environment temperature under control. The dataset is composed of 4 batteries run to failure, which is fixed at 70% of the remaining capacity (from 2 Ah to 1.4 Ah). The charging phase is divided into a constant current mode, with 1.5 A until the voltage gets to 4.2 V, and after that with a constant voltage phase, until charging current lowers to 20 mA. While the charge phase is equal for each battery, the discharge phase for each of them is:

- **B0005**: 2 A current, down to 2.7 V.
- **B0006**: 2 A current, down to 2.5 V.
- **B0007**: 2 A current, down to 2.2 V.
- **B00018**: 2 A current, down to 2.5 V.

In this case, an addition type of cycle is performed and used to measure the *impedance*, through an electrochemical impedance spectroscopy (EIS) frequency sweep from 0.1 Hz to 5 kHz. This kind of procedure discharges a tiny part of the battery charge, therefore doesn't impact on its aging. Figure 5.1 shows the degratation process over time of battery B0005, and Figure 5.2 shows the voltage profile describing and entire charge + discharge cycle.

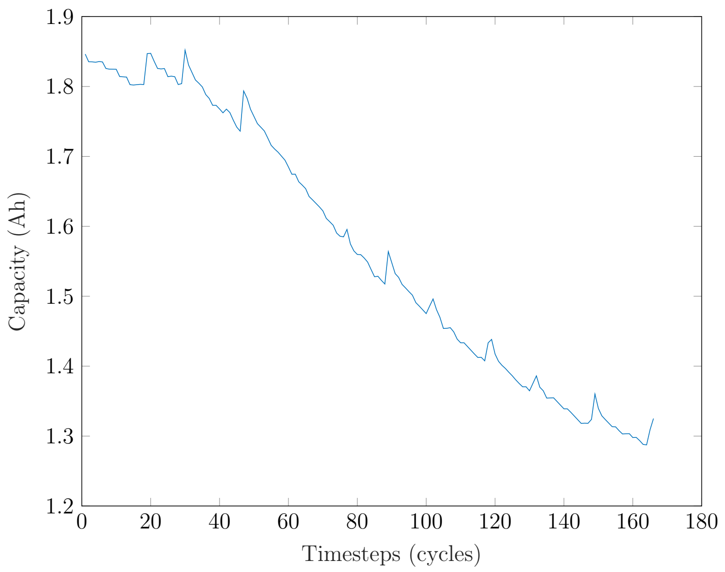


Figure 5.1: Capacity degradation over cycles. Battery sample 5

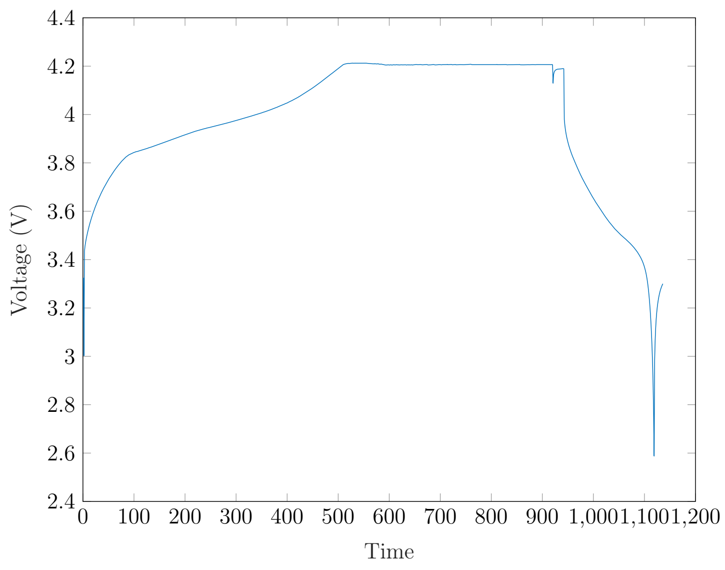


Figure 5.2: Voltage profile over time, of charge + discharge phases of 1 cycle

5.1.2. Structure

The top level of the structure contains the set of records of the cycles tested. Each record contains general information about the cycle, resumed in the following fields:

- **Type**: indicates the operation that is performed in this cycle: **charge**, **discharge**, or **impedance**.
- **Ambient temperature**: is the chamber temperature during this cycle.
- **Time**: the date and time of the start of the cycle.
- **Data**: a nested structure containing the sensors measurements within that cycle, for various parameters. For charge and discharge cycle this field contains:
  - **Time**: vector of time stamps (sec) which beats the time at which the sensors read the measurements.
  - **Voltage**: voltage level of the battery.
  - **Current**: current (Ampere) flowing into the battery. Positive is charging current, negative is discharging.
  - **Temperature**: battery temperature (degree C°).
  - **Voltage charge**: Voltage measured at charger (Volt)
  - **Current charge**: Current measured at charger (Volt)

– **Capacity (only disch)**: Battery capacity (Ah) for discharge till 2.7 V.

5.1.3. Pros and Cons

It is one of the first datasets made available for battery prognostic by a trustful entity, and it is well structured. This made it become one of the most used data repositories in research, and for this reason it is easy to find information about it and and make comparison with other solutions.
NASA adopted, for all the batteries in this dataset, the same charging policy, which also is constant in current over all the charging phase. This detail is not obvious, and allows us to extract valid features from the charging curves.
Unfortunately, being only 4 batteries available for prognostic purposes, and a number of tested cycles lower then 200 for each of them, this dataset is not particularly suitable to exploit the full potential of data-driven solutions, besides the battery number 18 quite diverges from the other 3, making it an outlier and hard to use.

5.2. Toyota-MIT dataset

Toyota and Massachusetts Institute of Technology (MIT) developed together a repository of tested Li-Ion batteries which is one of the most recent and rich datasets publicly available. It contains data of 124 commercial lithium-ion batteries, specifically lithium-ion phosphate (LFP)/graphite cells, manufactured by A123 Systems (APR18650M1A), cycled to failure, under fast charging conditions.

5.2.1. Properties

The batteries have been tested on a 48-channel Arbin LBT potentiostat, in an environment with temperature fixed at 30 °C, and the voltage cut-offs are 2.0 V and 3.6 V. Since the purpose of this dataset was to optimize the fast charging for Li-Ion batteries, different charging policies have been applied to them, leading to multiple different patterns of aging and charging phase curves. Also because of this reason, and because of the strong non linearity of the degradation, and its dependence on many factors all the batteries have drastically different life duration, from around 150 to 2200 cycles, even though all of them have an initial capacity of 1.1 Ah.
The charging phase is carried out with 2 steps of constant current, which are switched at a given **State of Charge** (**SoC**), and a 1 A CC-CV step which starts at 80% SOC. We will see that in some cases the switching between the first and second steps matches 80%

SOC, meaning that in these cases, there's only one step for constant current instead of two, after that CC-CV phase starts. In Figures 5.7 and 5.8 are reported the most used charging policies.
For completing the parallelism with NASA dataset, it is important to specify that in this case, the discharging phase is constant and equal for all the batteries, which wasn't in case of NASA dataset. Discharge current is fixed at 4 A.

5.2.2. Structure

The 124 batteries couldn't fit in the 48 channels available, so the dataset has been split in 3 smaller batches. Each batch contains 1 record for each of the 48 channels. Each record contains these fields:

- **Policy**: 2 fields containing the charging policy applied to the battery, in 2 slightly different formats.
- **Barcode**: a unique identifier of the battery.
- **Channel-id**: the channel to which the battery was attached in cycling phase.
- **Cycle-life**: number of cycles that the battery run over before failure.
- **Cycles**: nested structure, containing all the measurements done for each cycle of the battery.
  - **t**: vector of time stamps (sec) which beats the time at which the sensors read the measurements.
  - **Q charge**: cumulative moved charge in the charge phase (C).
  - **I**: current (Ampere).
  - **V**: voltage (Volt).
  - **T**: temperature (°C).
  - **Q discharge**: cumulative moved charge in the discharge phase (C).
  - **Discharge dQdV**: derivative of Q(V) in the discharge phase.
- **Summary**: nested structure containing summary overall information about each cycle (1 value per cycle):
  - **Q discharge**: charge moved in the discharge phase of each cycle.

- **Q charge**: charge moved in the charge phase of each cycle.
- **Tmax**: maximum temperature reached in the cycle (battery surface).
- **Tmin**: minimum temperature reached in the cycle (battery surface).
- **Tavg**: average of the cycle temperatures (battery surface).
- **Chargetime**: time needed for a full charge.
- **Cycle**: number of cycle.
- **IR**: measured internal resistance, obtained during charging at 80% SOC by averaging 10 pulses of ±3.6 C with a pulse width of 30 or 33 ms.

In Figures 5.3 and 5.4 is shown respectively the degradation over time (less noisy than degradation of NASA dataset batteries) and the voltage profile of a sample battery from the MIT dataset. In the voltage profile, we can see a step between cycles 300 and 400, which represents the voltage switching of the considered charging policy.

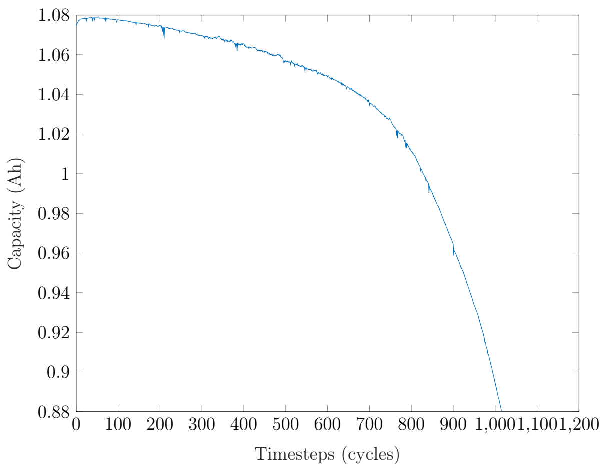


Figure 5.3: Capacity degradation over cycles. Battery sample 20

5.2.3. Pros and Cons

As mentioned before, this dataset is one of the largest available and is way bigger than the NASA dataset we discussed previously. Furthermore thanks to the different fast-charging policies which have been applied to the batteries, the dataset contains a big variety of data and measurements. This makes this repository perfectly suitable for applying Machine Learning and data-driven solutions to find and understand patterns and relationships. Its structure makes it easy to handle and many group measurements and calculations, like *average temperature*, or *discharge derivative* $dQdV$ have already been carried out,

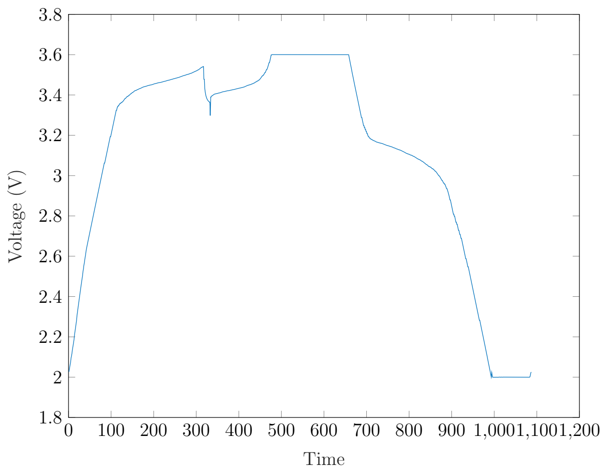


Figure 5.4: Voltage profile over time, of charge + discharge phases of 1 cycle

increasing its completeness.

One of the most remarkable difference we can notice among the different batteries is the length of their life in terms of cycles. In Figure 5.5 it is clearly shown that there's a wide discrepancy between the longest and shortest battery lives, with an average value settled around 700-800 cycles. We'll see that some of the most extreme cases will cause problems to the algorithms in the prediction because they have to be considered outliers, especially the very short ones. Figure 5.6 shows that the majority of the batteries are distributed between 400 and 1000 cycles life length, and very few are distributed outside these boundaries.

We analyzed the batteries life lengths by grouping them according to their charging policy. The charging policy is specified in the format:

Initial Current (% of SoC for current switch) Current after switching

Since all the tested batteries are equal in the model and initial condition, we would expect a general trend with batteries under the same policy having around the same life length. Figures 5.7 and 5.8 show instead a different situation with no general recognizable patterns, except for some isolated cases.
For example policy 4.8C(80%)-4.8C which is standard and pretty linear, has batteries going from 450 to 1800+ cycles in length.
This is probably caused by very intrinsic physical differences in the chemical components of the batteries, their conditions, and their manufacturing. It's well known indeed that battery degradation is also due to very complex combinations of little internal factors, chemical and physical which can't be measured, estimated, or known in any suitable way but doing an internal inspection of the battery.

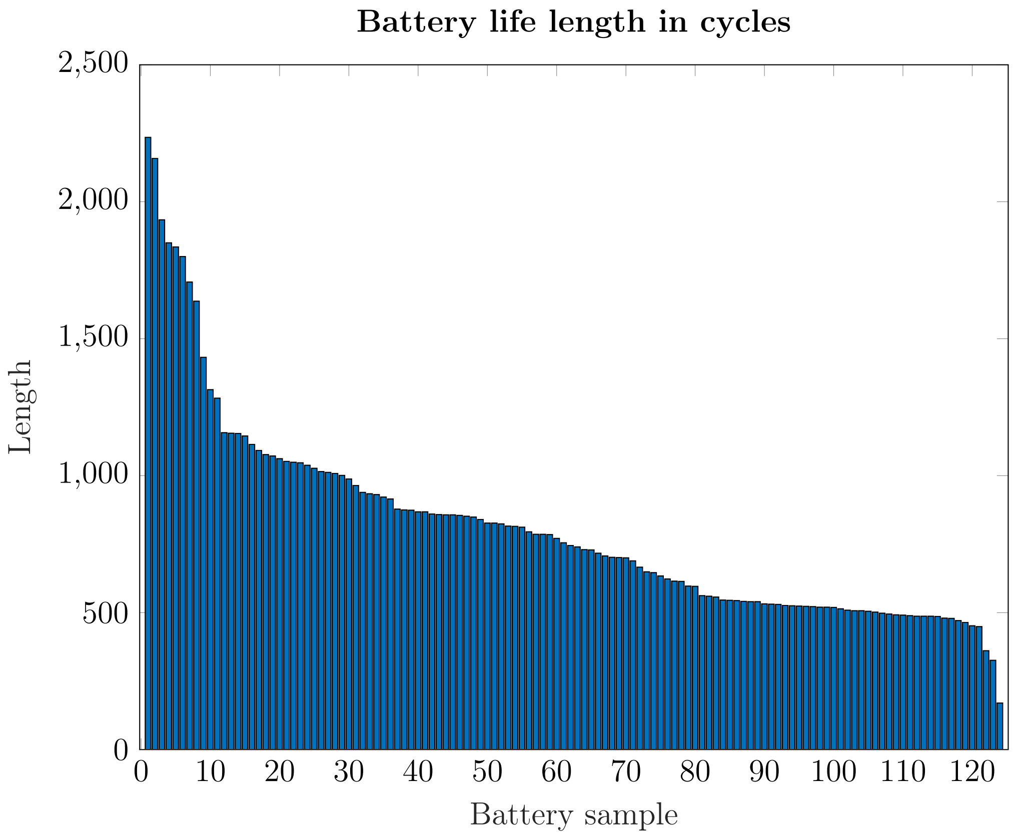


Figure 5.5: Life length of the batteries, in cyles. Descending order

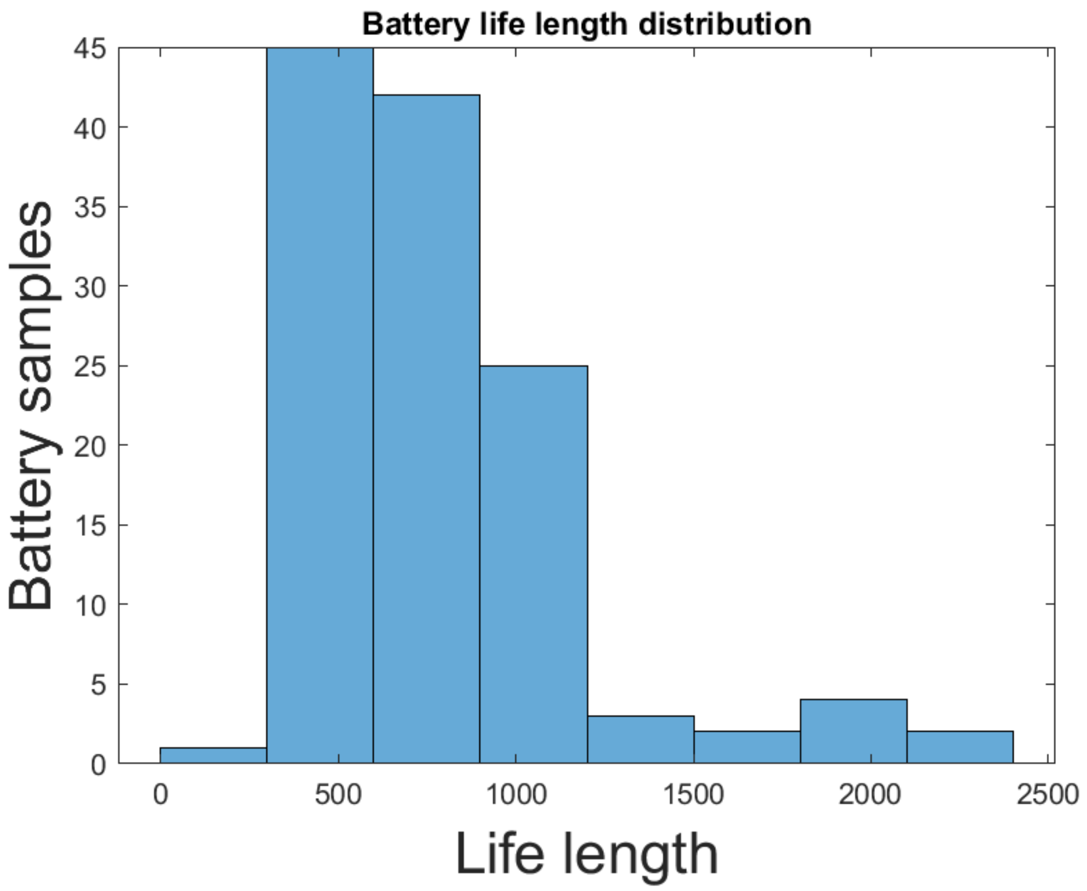


Figure 5.6: Distribution of the batteries according to the length of their life in cycles.

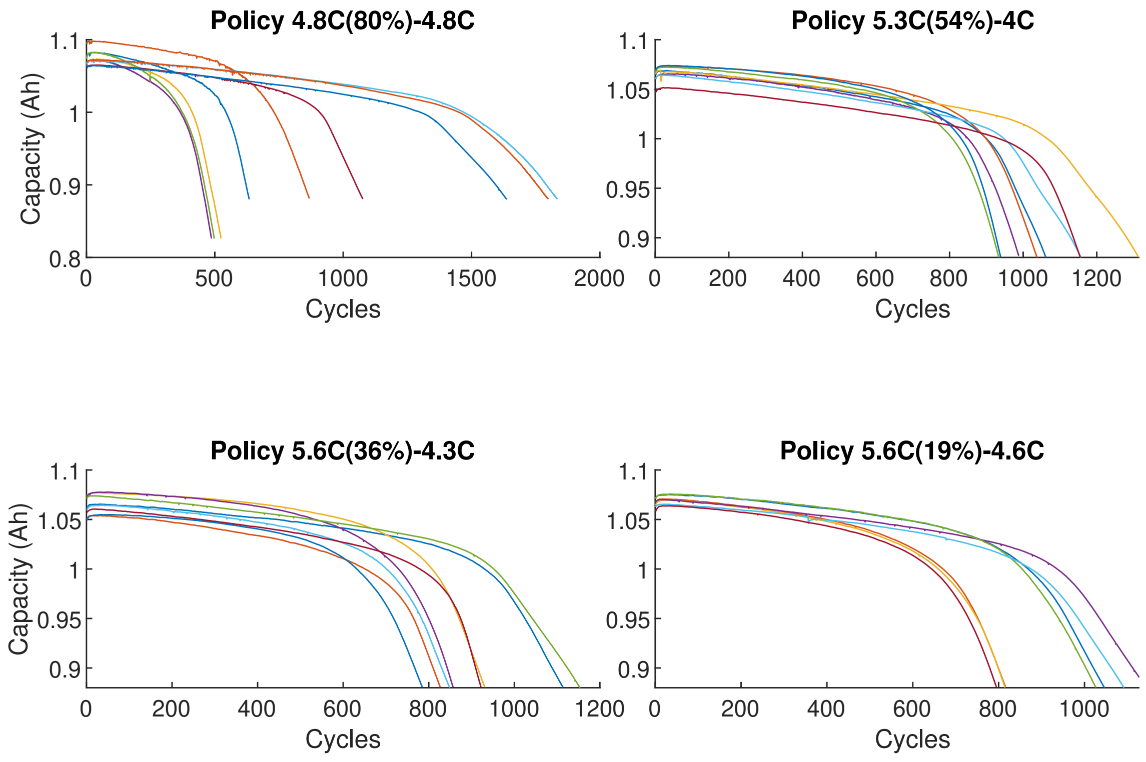


Figure 5.7: Capacity degradation of batteries with same charging policy. First 4 most populated policies

In this dataset, those intrinsic differences are strongly present and noticeable.

The repository, in the form which is provided, requires some cleaning before being used: the failure threshold of the batteries is fixed at 80% SoH (0.88 Ah). Even if the batteries are said to be aged until failure, actually some of them don't reach the failure point, making them useless for most purposes like RUL prediction. Those samples have to be discarded.
In addition, measures of batteries 8, 9, 10, 16, and 17 of the 2nd batch are actually the continuation of the batteries from 1 to 5 of the first batch.

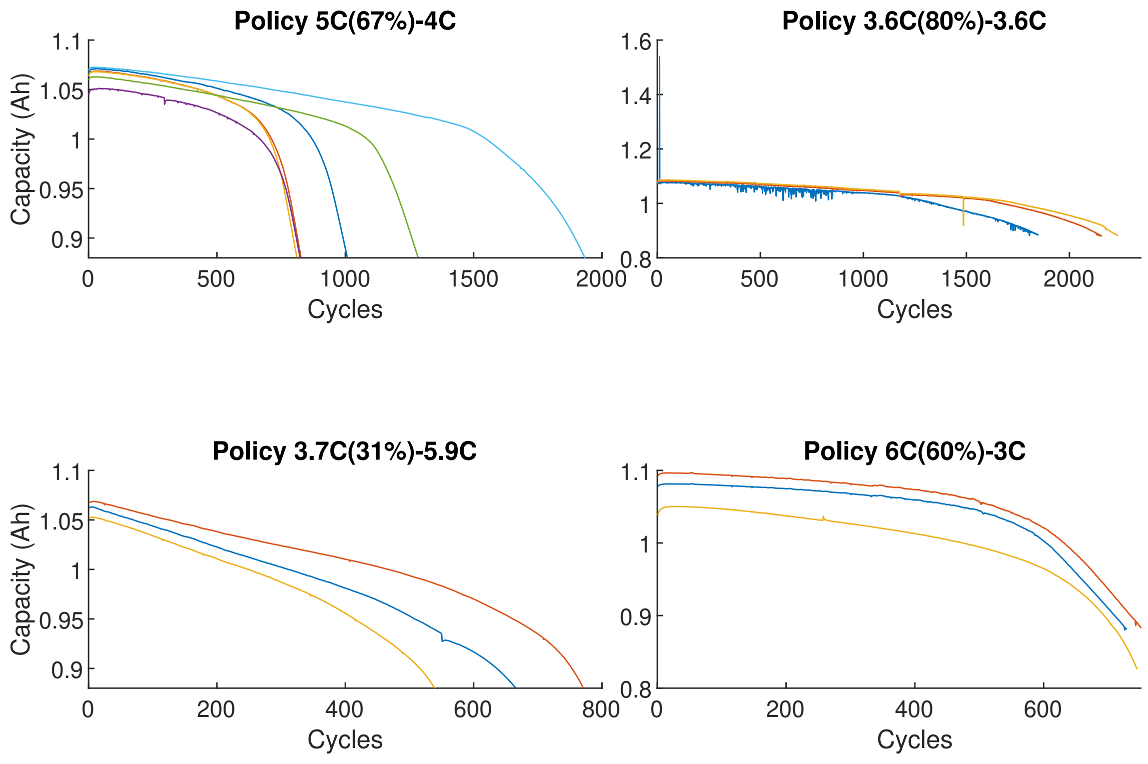


Figure 5.8: Capacity degradation of batteries with same charging policy. Second 4 most populated policies

6 | Hybrid approach: Neural Networks and physic-based models

The first approach we had to RUL prediction comes from the solution proposed in [3]. They proposed a hybrid approach that mixes a physic model component along with an **Echo State Network**, which is one of the first created NNs before the back propagation was developed to train actual NNs. In this work, they used the NASA dataset we described before.

6.1. Approach description

They start by taking into consideration a capacity degradation model called **Double Exponential**, which is well known and widely used in literature to describe the degradation of systems performance. The model is:

$$Q = a\exp(bk) + c\exp(dk)$$

where a, b, c and d are the parameters of the model and have to be estimated, while k is the number of cycles. From here, given a prediction cycle k, from the capacity measurements available up to k, the state space X_z is estimated

$$X_z = [a_z; b_z; c_z; d_z]$$

which is the set of parameters a, b, c, d which best fits the double exponential function over the capacity degradation function from the initial time to cycle k.

$$Q_z = a_z \exp(b_z k) + c_z \exp(d_z k)$$

They proposed then an alternative degradation model called **Single Exponential**, which is similar but simpler compared to double.

$$Q = C_0 + a_z \exp(\frac{b_z}{k}) + c \exp(dk)$$

Using these 2 models, they estimated the state space of batteries B0005, B0006, B0007 and B0018 of the NASA dataset, at prediction cycles respectively [80,100,120], [80,100], [80,100,120], and [80] for both models. The curves generated by the state spaces are then used to create a new dataset of fixed size, by sampling them, which follows the shape of the model used (single or double exp).
This new dataset is fed to the Neural Network, which predicts the remaining part of the capacity degradation curve, that is from the prediction cycle to the last cycle.
At this point, they find the cycle for which the capacity falls below the failure treshold, according to the formulas:

$$RUL_{meas} = EOL_{meas} - K_{pred}$$

$$EOL_{meas} = min\boldsymbol{K_r} = min\{kr|C_{kr} < FTH\}$$

$$\boldsymbol{K_r} = \{k_r|C_{kr} < FTH\} \forall k_r \in [k_{pred}, n]$$

$$FTH = 0.7C_{rated} = 1.4Ah$$

$RUL_{meas]}$ is the measured value of the RUL, EOL_{meas} is the cycle in which the end of life is reached. K_{pred} is the prediction cycle. C_{k_r} is the capacity at cycle k_r, and K_r is the set of cycles indices, from prediction cycle to the last cycle, for which the capacity is less then the failure threshold FTH. C_{rated} is the nominal capacity of the battery.

The descriptor of the data they give as input to the network is the sequence of numbers of cycle of the degradation curve until prediction cycle k.

We tried to replicate their approach adopting an LSTM network instead of an ESN, composed by 1 **Input layer**, 1 **LSTM layer**, 1 **Fully connected layer** and 1 **Regression layer**, and applying their method on battery B0007.

6.2. Results and discussion

The results we found prove that their approach is not very solid and sound.

We were able to replicate the results they had using a dataset of 100 points from the one generated by the single exponential state space (like the procedure they described), and giving it as a sequence to train the network. We then used the network to predict the remaining part of the degradation curve. We did this for prediction cycles 80, 100, and 120, thus we trained 3 different networks, with 3 different state spaces, each estimated using the Matlab fitting tool.

Figure 7.6 and Table 6.1 show good results very similar to the original ones, but we've been able to obtain them only out of a lucky training configuration. Since one model per prediction cycle is built and only one battery is used to train the network, only one sequence is used by the network to learn the pattern and predict another sequence, which is in a completely different degradation phase compared to the training one. So one data point has been used to train a model which then predicts another data point which in the input space is very far from the training sample, since the only descriptor is the number of cycle, which is always different and constantly increasing from the first known phase of the curve, to the last unknown phase.
This approach is not to be considered data-driven, and doesn't exploit the potentiality of the dataset. In addition, a neural network is likely a too complex model to use with such approach.

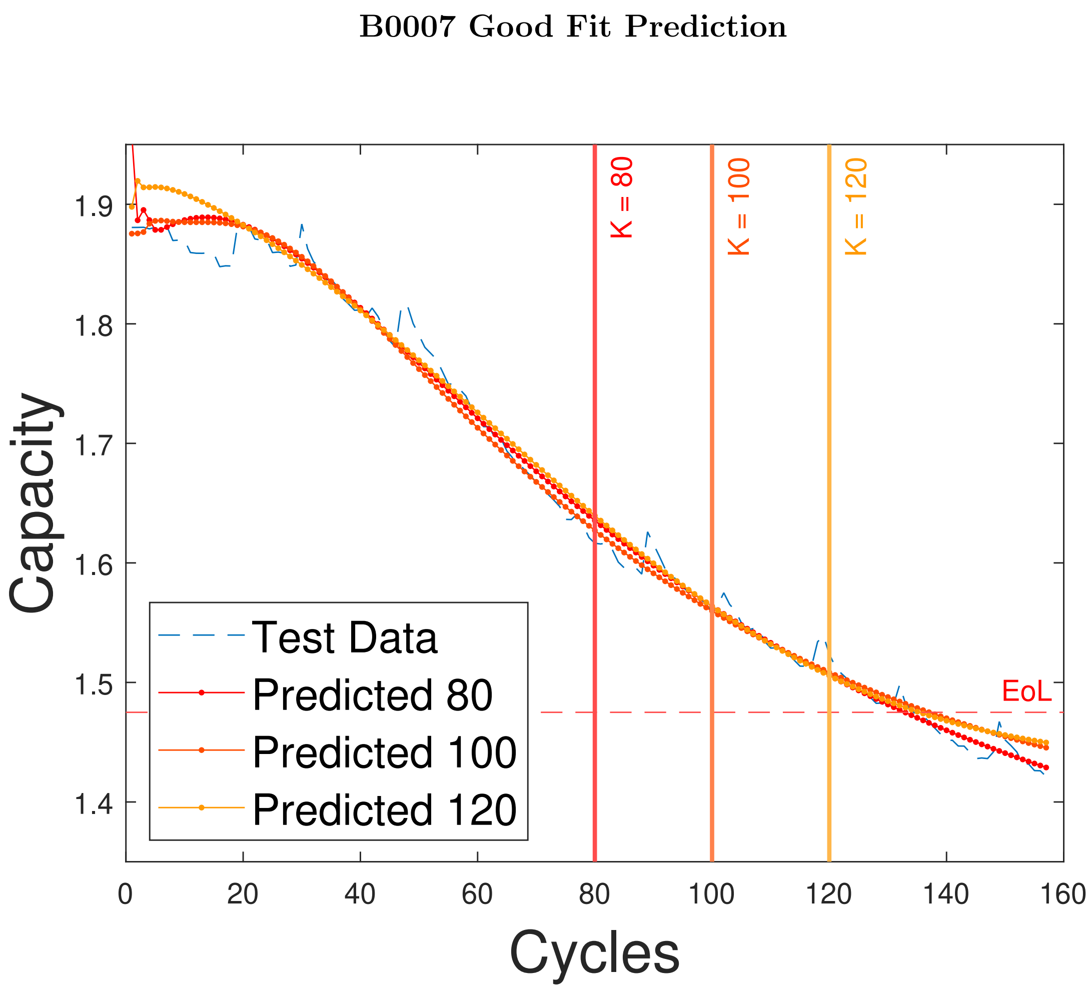


Figure 6.1: B0007 Capacity degradation predicted using LSTM network, trained with Single Exponential Model, for prediction cycles 80, 100, 120.

B0007 Good fit RUL error

Model	K_{pred}	RUL prediction error
Single Exponential	80	1
	100	2
	120	0

Table 6.1: Error in RUL prediction from cycles 80, 100 and 120. Error is expressed in cycles.

B0007 Standard Fit Prediction

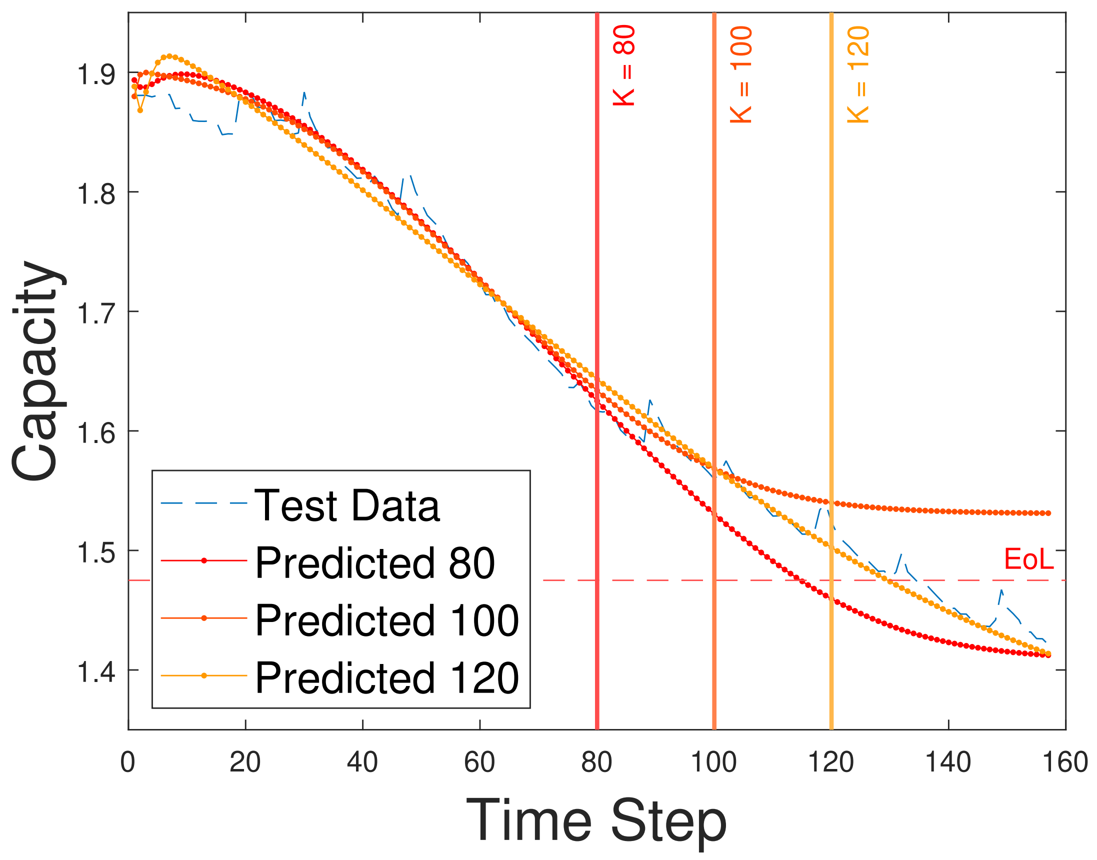


Figure 6.2: B0007 Capacity degradation predicted as before, but with model retrained. Time step is measured in cycles.

B0007 Standard Fit RUL error

Model	K_{pred}	RUL prediction error
Single Exponential	80	20
	100	-
	120	5

Table 6.2: Error in RUL prediction from cycles 80, 100 and 120. Error is expressed in cycles.

The results we get if we retrain the network multiple times are similar to the ones reported in Figure 6.2 and Table 6.2. In fact, the only part of the curve that the network can learn

is the part before the prediction cycle. Not having any data available about how a battery behaves in the unknown part, the prediction becomes all the times a random continuation of the pattern of the first half of the curve.
For this reason, the prediction is very sensitive to any small change in the configuration of the network, like training epochs, learning rate, and so on. For example, we get a different random prediction for the second part just by adding or removing one training epoch. Looking at the curves predicted in Figure 6.2 we can notice how each predicted curve just tends to flatten sooner or later, following the inertia of the curve of the first part.
The particular case of the prediction from $k = 100$ in Figure 6.2 doesn't even get to predict the value of capacity low enough to compute the RUL.

The only way in which we can obtain solid results over different runs of the algorithm is by using the data to predict also as a validation set, and by telling the network to output the model with the best validation error, at the end of the training.
This is of course a case of overfitting over the test set, as well as an impossible solution to apply in real cases.

7 | Our solution

The solution we propose here is articulated through the first phase of SoH estimation, performed on both NASA and MIT/Toyota datasets, and the second phase of RUL prediction applied on the MIT dataset only. We made this decision in light of the results we obtained on RUL prediction on the MIT dataset: the three sequences of battery degradation measures in the NASA dataset would not be enough to properly train and validate the neural network we used in our approach, and obtain meaningful results.
SoH and RUL are closely tied together since they both represent the degradation of the system, and in the particular case of the battery, a frequently used approach for computing the RUL, is predicting the SoH at future cycles, locating that point when SoH reaches failure ($<$EoL threshold), and then measuring the cycles remaining until that point.
Despite this, SoH estimation and RUL prediction are still very different tasks to carry out, and this difference lies between the concepts of estimation and prediction, which is important to understand in order to properly understand the steps of this work.

Estimation process requires to use data collected up to time t, to infer the value of something at the same time t, so we have to guess some state and we can use all the possible information available which led to that state, to make our guess.

Prediction instead requires to use data collected up to time t (called prediction time) to guess the value of something at some future time t_f, which, however, depends on everything that happens from the beginning to the end. In the case of batteries, the RUL at cycle k depends on the aging cycles before and after k, to the last cycle (failure). This means that we need a way to evaluate also how the battery behaves in the time window for which we have no measures. For doing this we adopted a model able to represent the sequential structure of the system life, and not only consider one cycle and the relative measures as one point in the dataset, but as one point within a sequence with a precise beginning and end. This allows the model to catch how the battery did behave from the beginning to cycle k, and use this information to approximate also its future behavior.

Is well known that battery degradation has an intrinsic time-series structure, indeed models like Recurrent Neural Networks, which handle sequential structures, are by nature the best suited to deal with battery aging.

7.1. SoH estimation: NASA dataset

Support Vector Regression technique has been proved to be effective in battery degradation estimation, and used in many researches even with this very same NASA dataset, because of its ability to work with data scarcity. We applied it to our SoH estimation problem along with a **feature-based** approach oriented to the extraction of features that are both meaningful in explaining the data, and easily exploitable in most contexts of real-scenarios usage.
For instance, some industrial applications of batteries may make constant and reliable usage of them during the time, performing full charge and discharge cycles of the cells, with regularity over time, and these cases allow to extract measurements and feature from the entire cycle. These features are more likely to properly describe the data and the state of the battery because they explore the full range of the battery state of charge.
There are other applications instead, like electric vehicles, in which the usage of the cells is not regular: the user could make use of the battery for a small portion of its capacity, then charge it for a while, then discharge it again, and so on, and this would make impossible to take measurements and feature from a continue charge + discharge cycle of the cell.

Our solution proposes to extract features from the cycling of the cells that could be easily extracted in almost any real case of use, at the price of a small portion of the descriptive power of the features themselves.

7.1.1. Partial charging curve features

In most applications, the charging phase is guaranteed to be carried out under constant current, since it is given by the charger device. Also it is likely to be carried out completely since it is good habit and convenient to wait for the charge to be at 100%, before unplugging, but this is not guaranteed. We decided then to use a small portion of the charging curve to extract the features.
Other studies have been done on degradation estimation using partial curves, for instance, in [9] a portion of the charging curve of batteries is taken and the degradation is estimated by similarity, finding the 2 batteries in the training set with the same partial curve being the most similar. In this work specifically, the features we extract are the time required

by the battery to recharge some little voltage steps, defined within a voltage interval. For instance, we can define a voltage interval from 3.5 to 4 V, of 0.5 V, and a voltage step of 0.1 V. We will obtain 5 features, each being the time taken by the battery to cover that recharge step.
The interval and the steps considered become hyperparameters to tune.

$$\begin{cases} V_1 = V(t_1) \rightarrow (V_1 + V_s) = V(t_{s_1}) \\ feature_1 = t_{s_1} - t_1 \\ (V_1 + V_s) = V(t_{s_1}) \rightarrow (V_1 + 2V_s) = V(t_{s_2}) \\ feature_2 = t_{s_2} - t_{s_1} \\ \vdots \\ (V_1 + (n-1)V_s) = V(t_{s_{n-1}}) \rightarrow (V_1 + nV_s) = V(t_{s_n}) \\ feature_n = t_{s_n} - t_{s_{n-1}} \end{cases} \tag{7.1}$$

We can see from **Figure 7.1** that, as we expected, the time of recharge decreases while the battery ages, and this is because of the global capacity reduction due to the degradation. We can actually see how the charging time is halved near to final cycles, with respect to the first.
We see also that the first part of the charging phase is almost vertical, which means that voltage raises rapidly in time, and it is hard to appreciate the time differences between different cycles. The middle phase instead is the more suitable to extract time features since it is the most extended over time, and we expect it to contain more variance over the cycles set of the battery.

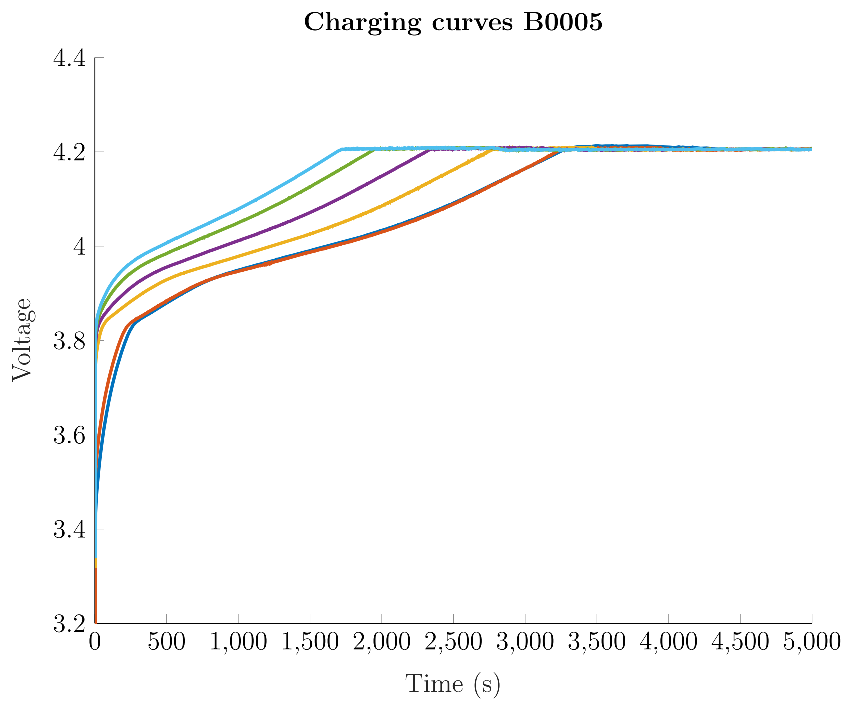


Figure 7.1: Charging curves B0005 at different cycles

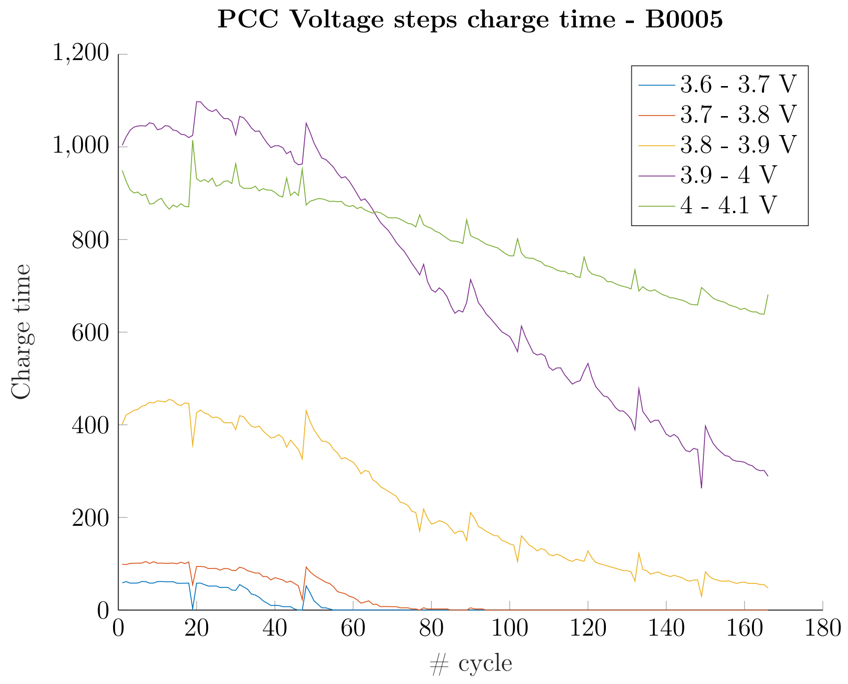


Figure 7.2: Time needed to charge voltage steps of 0.1 V from 3.6 to 4.1 V, for all cycles of battery B0005

In **Figure 7.2** we can see the result of sampling the charge times for voltage steps of 0.1 V, in the interval 3.6 - 4.1 V. Each line in the plot is a voltage step and a potential

feature. As expected, the plots from the lower part of the spectrum of the voltage curve result in almost flat features, showing no variance over the life cycles of the battery.
Similar behavior occurs in the higher voltage steps, where the green line again tends to flatten.
The same results hold for batteries B0006 in **Figure 7.3** and B0007 in **Figure 7.4** so we decided to consider the middle range from 3.8 to 4 V for feature extraction.

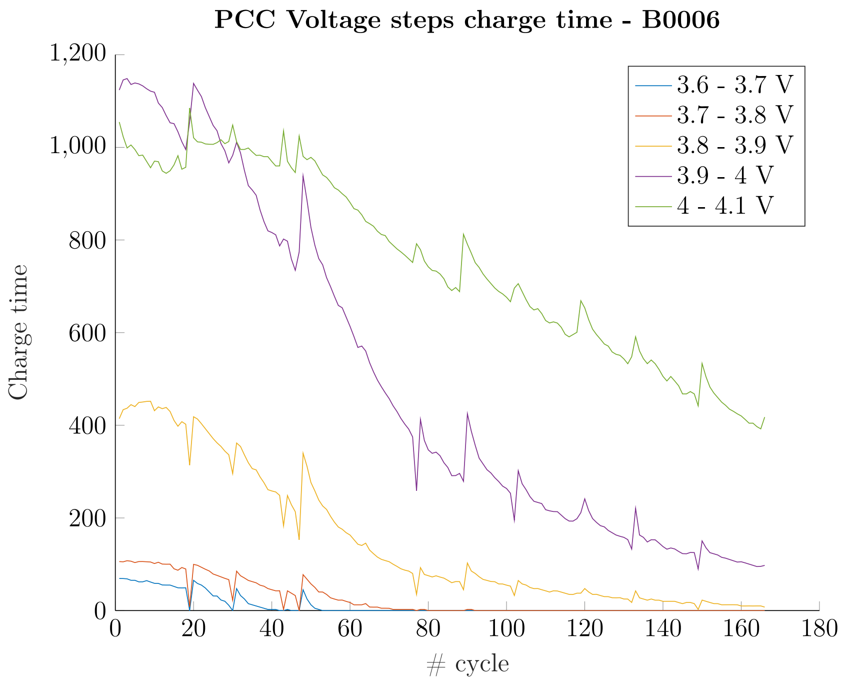


Figure 7.3: Voltage steps for all cycles of battery B0006

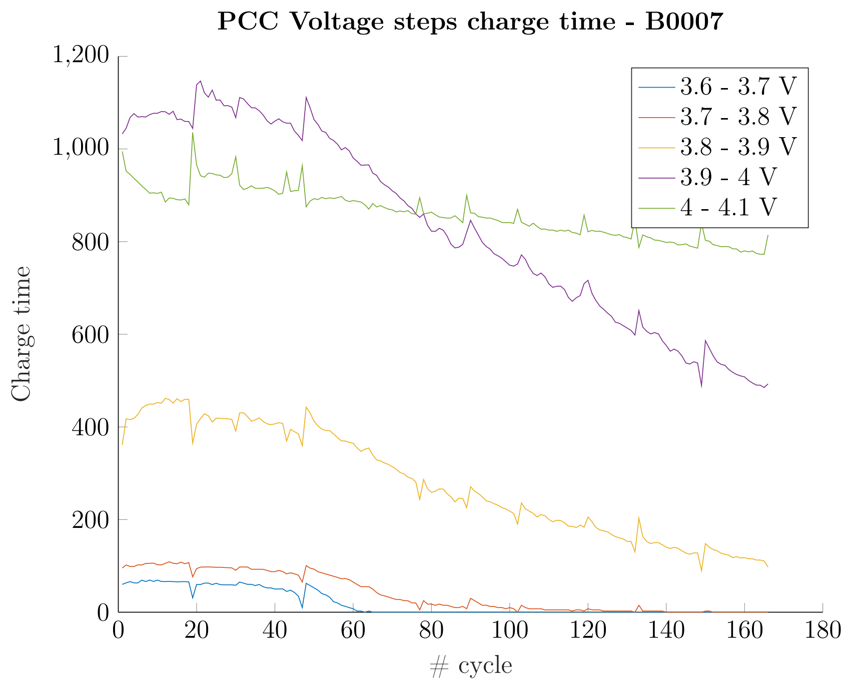


Figure 7.4: Voltage steps for all cycles of battery B0007

7.1.2. Models and results

We developed 6 models with different subsets of features, gathered from the 3.8 - 4 V range. We measured the fitting capacity of the model through R^2 value: also called the coefficient of determination, it is a measure used in statistics, indicating how much a hypothesis describes the variance of the thesis. In other words, it is used to describe how well a model can fit observed data. R^2 formula is:

$$\begin{cases} R^2 = 1 - \dfrac{SS_r}{SS_t} \\ SS_r = \sum_i (y_i - f_i)^2 \\ SS_t = \sum_i (y_i - \overline{y})^2 \\ \overline{y} = \dfrac{1}{n} \sum_{i=1}^{n} y_i \end{cases} \tag{7.2}$$

where SS_r and SS_t are the residual sum of squares and total sum of squares. y_i is the target value, f_i the estimated value. $\overline{y}$ is the mean of the target values. The descriptor values of the data have been standardized feature-wise before fitting the model. Standardization is a normalization procedure that is a good habit to apply in machine learning applications. It sets the variance of the data to 1, and the mean to 0, without changing its distribution. Standardization improves the numerical stability of the model and may reduce training time.

Hyperparameters of the SVR model have been tuned initially with the Matlab built-in function using Bayesian optimization run for 100 iterations, to define a starting good range for the hyperparameters:

- **Box Constraint**: SVR solves the optimization problem of finding weights beta s.t.:

$$y_n - (x'_n\beta + b) \leq \epsilon \tag{7.3}$$
$$\tag{7.4}$$

  while minimizing the norm value:

$$j(\beta) = \frac{1}{2}\beta'\beta \tag{7.5}$$

  as explained in Equations 4.1 and 4.2.
  When slack variables are introduced to deal with unfeasible constraints, they are weighted by a factor called Box Constraint **BC**.

$$j(\beta) = \frac{1}{2}\beta'\beta + BC\sum_{n=1}^{N}(\xi_n + \xi_n^*) \tag{7.6}$$

  Box Constraint helps controlling overfitting, balancing how flat the fit will be, and how much to weight those points who lie outside the boundary defined by ϵ.

- **Kernel Scale**: rescales the predictors: each value in the predictors are divided by the **KS** value.

- **Epsilon**: it is the value ϵ defining the radius of the **Epsilon Tube** where the algorithms tries to contains the points, or in other words according to Equation (7.3), the maximum error allowed.

After this procedure, further tuning of the hyperparameters has been done during the validation procedure.
The kernel function has been chosen to be **linear** since the features represent in a pretty much linear way the target value to estimate, and other kinds of kernel functions led to lower validation accuracy.

SVR Hyperparameters

	Hyperparameter	**Value**
1	Box Constraint	0.1989
2	Kernel Scale	11.55
3	Epsilon	0.030
4	Kernel Function	Linear

Table 7.1: contains the tuned hyperparameters values for the considered SVR model

We evaluated the performance of the models, and tuned the hyperparameters by Leave One Out Cross Validation among the 3 batteries B0005, 6 and 7. The Table 7.2 reports the models used and the relative validation errors.

The best accuracy is provided by the 3.9 - 4 V range since either model with one feature or two performs better in that range (Model 2 and Model 4). Since two features also gave better results then 1, we tried to extract charge time from 0.025 V steps in the same range (Model 6), to understand if more complexity would benefit the performance, but it didn't. We suppose that if more data were available, and our dataset had more variance in it, we could have exploited successfully more complex models.

Leave one out cross-validation results

	N° of features	Features	Validation R^2
Model 1	1	Charge time of 0.1V step from 3.8 to 3.9 V	0.918
Model 2	1	0.1V step from 3.9 to 4	0.932
Model 3	2	0.05V steps from 3.8 to 3.9 V	0.939
Model 4	2	0.05V steps from 3.9 to 4 V	**0.950**
Model 5	4	0.05V steps from 3.8 to 4 V	0.945
Model 6	4	0.025V steps from 3.9 to 4 V	0.935

Table 7.2: Results of leave one out cross-validation for different subsets of features

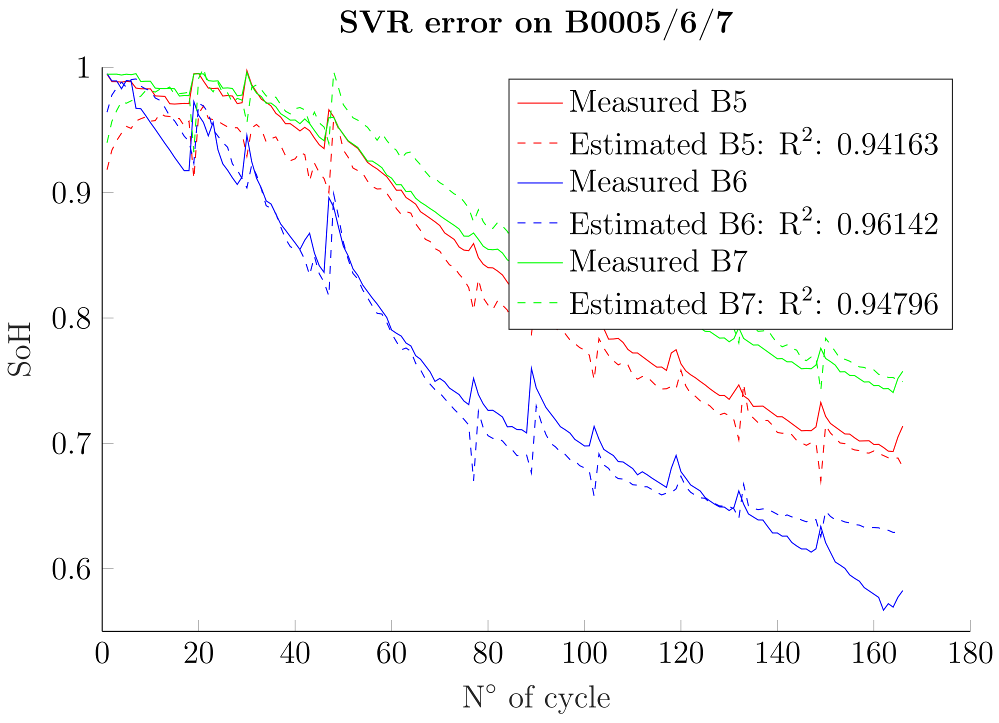


Figure 7.5: Results of the proposed model. Each estimation has been made by training on the other 2 batteries.

7.2. RUL Prediction Toyota-MIT dataset

The second step of this thesis work is to perform RUL prediction, and to do so we moved on the Toyota-MIT dataset: it contains many more different samples and more variance, which allows us to dig deeper into the subject and obtain more sound results.
RUL prediction as we said before is a more complex task to accomplish, we approached it using artificial neural networks, in particular a bidirectional **LSTM** Neural Network, which we introduced in Subsection 4.2.5, and has the ability to handle the battery aging as time series.
We use this property to make the network learn the past aging pattern that the battery has been subject to, so that it can estimate also the future aging, and guess the value for RUL.

We again used a feature-based approach, slightly different from the previous one, because, as we mentioned, batteries now have been charged with many different policies, thus any features collected from the charging measurements is not suitable to compare the degradation state of different batteries and thus not suitable to train a model and make predictions.

According to [23] there's much more information about degradation contained in the Q discharge curves of each cycle rather than in the degradation curve itself. They indeed analyzed the Toyota-MIT dataset and extracted different features from the discharge curve of the Q as a function of voltage, and used them to produce a linear regression model for predicting RUL at a fixed prediction cycle 100.
Starting from their studies, we used their approach a re-adapted some of the features they computed, so that we can predict RUL at *any* prediction time, and not only for the beginning as they did. In this way we expect to have a result for predictions at cycle 100 very similar to their result, and to increase the accuracy step by step while we move toward the EoL of the battery.

7.2.1. RUL Features

Neural Networks are known for their capacity to abstract from the domain of the data they are operating on, by computing their internal representation of the data, which we could call features, that would not be human interpretable. Despite this, we still keep a feature-based approach, since we are able to exploit our knowledge of the battery domain and we gain in interpretability.

The first 2 features derive from the fact that the analyzing the Q provided during the discharge phase, and how it changes over cycles, we can describe the progressive degradation of the battery performance.
We consider the Q(V) function of voltage for each cycle (discharge phase). Since the discharge voltage window is equivalent for all batteries and for all cycles, we can actually compare the curves for different cycles putting in relation different cycles and about the occurred performance loss.
Figures 7.6 and 7.7 show the discharge curve of Q as a function of V, for a sample battery, at two different cycles. In particular, they show that the gap in the curves between the initial reference cycle (we chose 10th cycle as a reference) and the prediction cycle constantly increases. The area between the 2 curves can be seen as the capacity lost due to aging from cycle 10 to cycle k.
To have a measure of the gap between the curves, we subtract them, but other methods like measuring directly the area between the curves, could be used.
We applied an interpolation with a common base of voltage values to both the arrays of Q values. This step is needed so that we can subtract the curves point by point on the same domain.

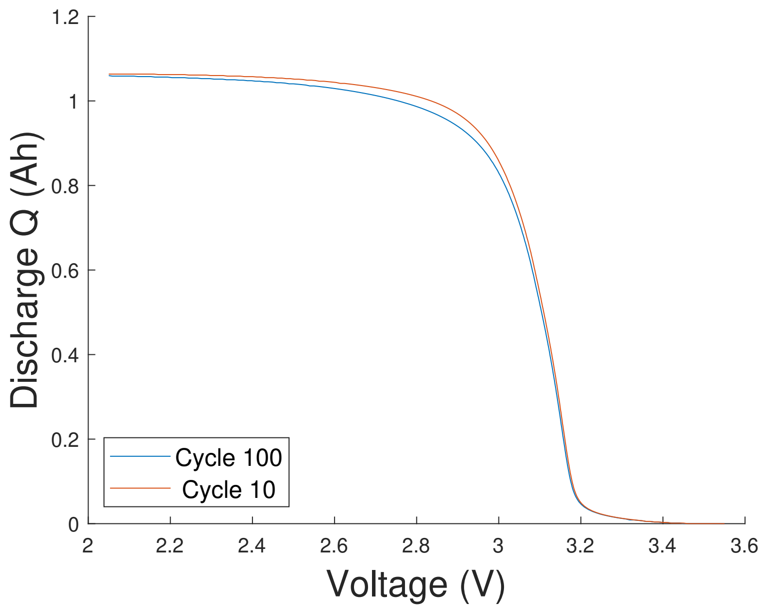


Figure 7.6: Q(v) discharge curves cycles 10 and 100, from a sample battery

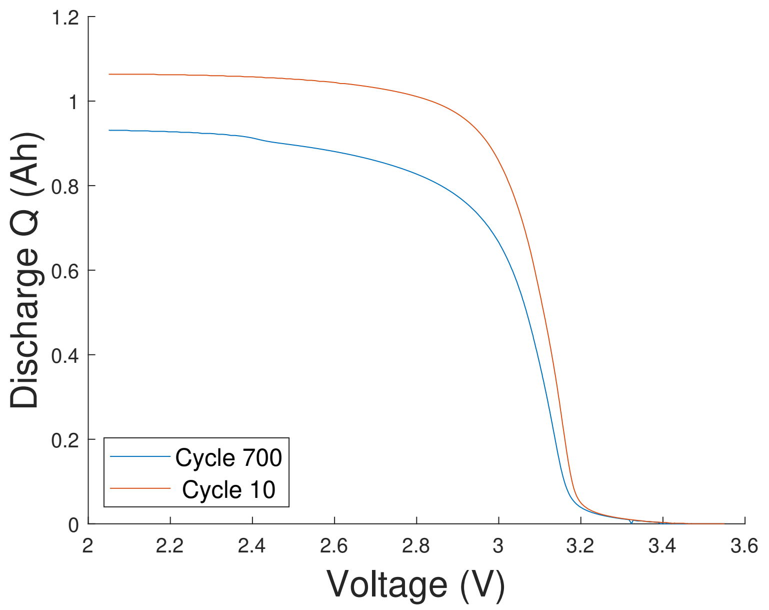


Figure 7.7: Q(v) discharge curves cycles 10 and 700, from a sample battery

For each cycle k of each battery, $Q_k(v)$ is subtracted to $Q_(10)(v)$, obtaining Figure 7.8, and from this we compute the variance of each difference curve. The result is a feature space on which the data are almost linearly distributed as a function of the length of the battery life, that is, after subtracting the prediction cycle, the RUL we want to predict.

From the same curves in Figure 7.8, we obtained a second feature by measuring the **min-**

imum value of each curve, for each cycle, for each battery. This is a similar description of the very same phenomena we computed with the first feature, indeed the data projected on the space of this second feature have a pretty similar shape of the previous.
PCA analysis will confirm that this second feature doesn't bring any new knowledge of the variance of the dataset.

In Figures 7.9 and 7.10 it is shown the value of the Log10 Variance feature and Log10 Min feature, for cycle 100 and cycle 700, and we see that for both fresh and aged batteries, they are linearly described according to their remaining useful cycles.

The third feature we computed is the sum of temperature: for every cycle k of each battery, the average temperature of each cycle from the 2nd is summed up to the $k-th$. Figure 7.11 shows the 3rd feature for all batteries at cycles 100 and 700.

$$\begin{cases} Ftr_k^1 = log\left(|\frac{1}{p-1}\sum_{i=1}^{p}(\Delta Q_k(V) - \overline{\Delta Q_k}(V))^2|\right) \\ Ftr_k^2 = log\left(|min(\Delta Q_k(V))|\right) \\ Ftr_k^3 = \sum_{cycle=2}^{k} \overline{T}_{cycle} \\ \Delta Q_k(V) = Q_k(V) - Q_{10}(V) \\ \overline{\Delta Q_k}(V) = \frac{1}{p}\sum_{i=1}^{p} \Delta Q_k(V) \\ \overline{T}_{cycle} = \frac{1}{Nmeasures} \sum_{i=1}^{Nmeasures} T_i \end{cases} \tag{7.7}$$

Equation 7.7 explains the 3 three features we used and how we computed them. Each of them as a value for each cycle of each battery. $Q_{10}(V), Q_k(V)$ are the discharge capacity as a function of voltage, at reference cycle 10, and cycle k (Figure 7.7), p is the number of values in the curve $\Delta Q_k(V)$, hence $\overline{\Delta Q_k}(V)$ is the mean of that curve, for cycle k. T_i is a single temperature measurement, and $\overline{T_{cycle}}$ is the mean temperature over one cycle.

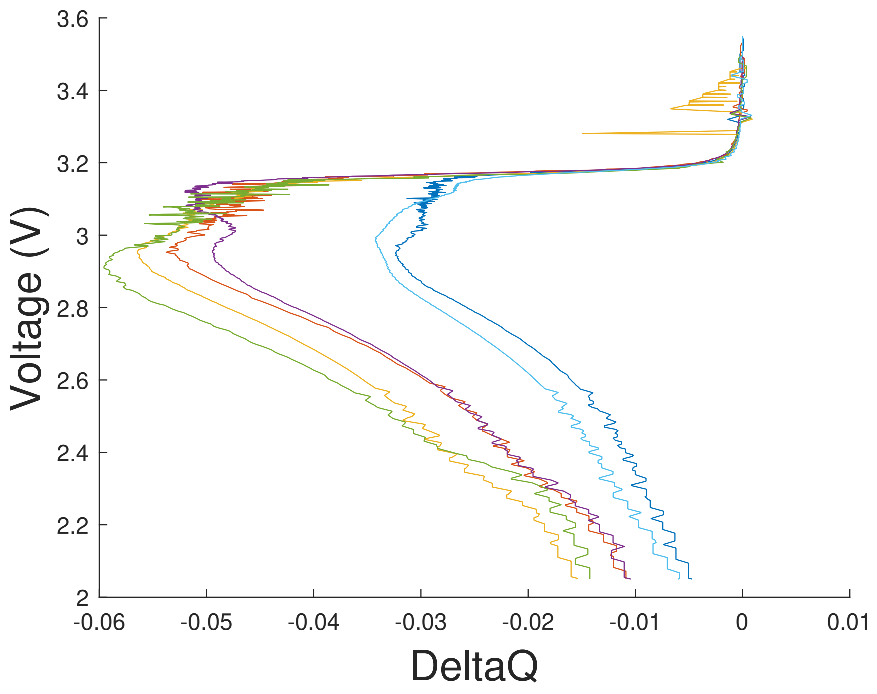
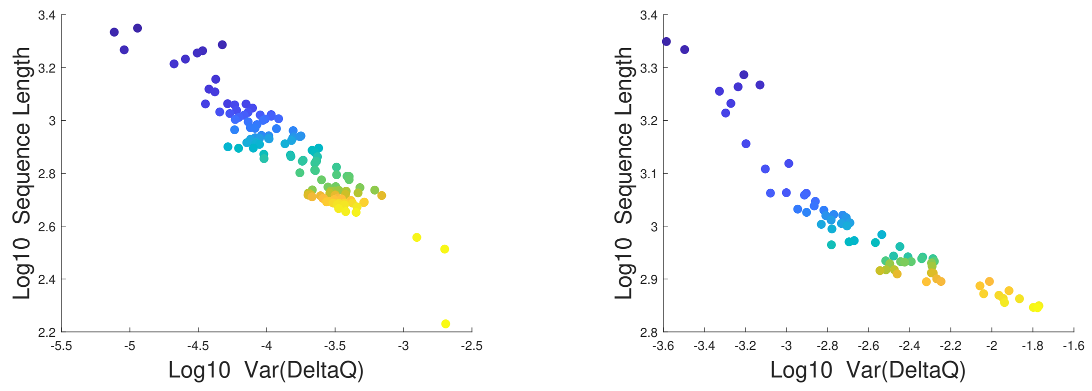


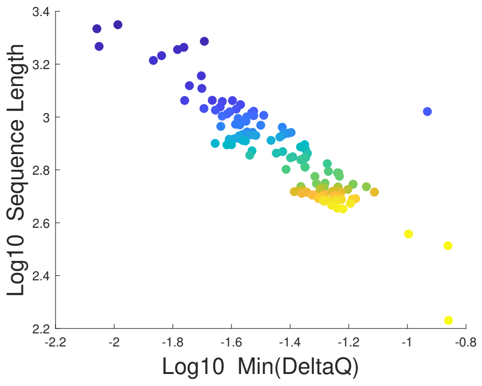
Figure 7.8: Difference between Q(v) curves at cycles 100, for different batteries



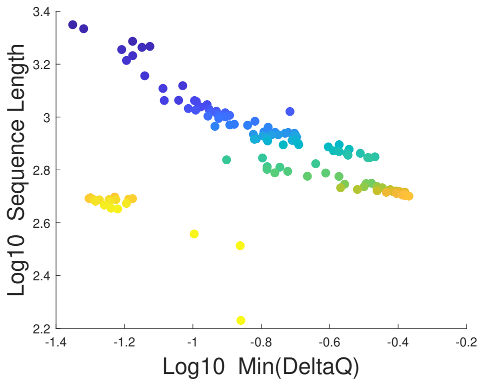
(a) Log10 of Variance DeltaQ feature, for all batteries at cycle 100

(b) Log10 of Variance DeltaQ feature, for all batteries at cycle 700

Figure 7.9: Feature 1: variance

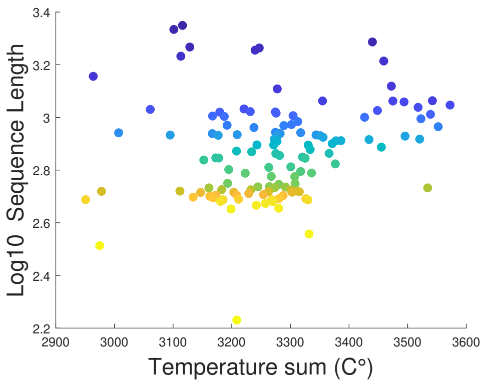


(a) Log10 of Min DeltaQ feature, for all batteries at cycle 100

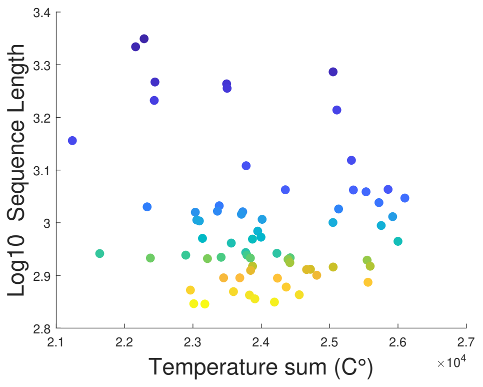


(b) Log10 of Min DeltaQ feature, for all batteries at cycle 700

Figure 7.10: Feature 2: minimum



(a) Sum of average temperature at cycle 100, for all batteries



(b) Sum of average temperature at cycle 700, for all batteries

Figure 7.11: Feature 3: temperature sum

7.2.2. LSTM Network and Training

We used an LSTM Recurrent Network composed like this:

- **Sequence Input Layer**: This let us to give in input an entire sequence to the network.
- **Bidirectional LSTM Layer**: Compared to the classic module, bidirectional puts together 2 modules that cross the network from left to right, and from right to left. Bidirectional LSTM is slightly more powerful than the classic LSTM module. This layer will output a sequence of 1 value for each input. Each value is the prediction associated with all the inputs and outputs prior to this timestep. Each value is also an input for the next step of prediction.
- **3 Fully connected layers**: respectively with 200, 50, and 50 neurons each. Those are called **hidden Layers**, and elaborate on the internal representation of the data which is produced by the LSTM module, to obtain the final output. Our LSTM layer is composed of 20 hidden units.
- **3 Dropout layers**: One right after each hidden layer, with a dropout factor of 0.4. This layer randomly shuts down a percentage of neurons, specified by the dropout factor, at each iteration. This is used to prevent overfitting, and acts like a sort of regularization. Since dropouts are only applied during training, as a drawback sometimes this leads to a validation error constantly lower then the training error, which is an unwanted behaviour because is hardly interpretable and misleading.
- **Fully Connected layer**: one last fully connected layer with only one neuron (the number of outputs we need).
- **Regression Layer**: computes the final output response and the corresponding loss value.

The memory capacity of Recurrent Neural Networks is limited in any case, and generally LSTM module hardly deals with sequences longer than around 500-600 time steps, (or slightly more depending on other factors) without incurring in the **Vanishing Gradient** problem, which reduces the weights update to 0 due to the subsequent multiplication of the gradients in the chains rule.
To deal with this, we decided to limit the target RUL value of all the samples to 800 cycles. The price for this is that we're not going to have a prediction for the first cycles of those batteries which are longer than 800 cycles, but this is not a problem from the point of view of RUL prediction, since it is intended to become useful and more precise

near to the end of the life of the battery, so to schedule appropriate maintenance.
For this reason, in the results plot, we show the full RUL prediction for those batteries whose life is shorter than 800 cycles, while for long-lived batteries prediction will be done only for the last 800 cycles.

Before training the network we ordered the sequences by length. We used a **Batch Size**=8, to give the sequences in input to the network 8 at a time and thus smooth the gradient direction by averaging it over 8 samples per time. The sequences are not all the same length though, and padding is added to each sequence to match the longest within the same batch.
We can see then that in order to add less padding possible it is important for sequences in the same batch to be as close as possible in length, so they have to be ordered.
We used a soft **L2 regularization** to further help reduce overfitting, and a dynamic **Learning Rate** schedule starting from $LR = 0.002$, and decreasing by a factor of 1/2 every 300 epochs.

We applied 5-fold **Cross Validation** to tune the hyper parameters: hidden layers, number of neurons and hidden units, batch size and dropout factor, and the dynamic of the learning rate.
Instead for training the final model, another validation set has been used to early stop the training and pick the best general model possible, avoiding overfitting.
10 batteries have been left out from the rest of the set, in order to test the model. It is important to keep the test set only for the final evaluation of the generalization capacities of the model, and do not produce any changes in the strategy, based on the test error, otherwise, we risk overfitting the test set, and we won't have any other unknown sample to perform a final unbiased error estimation.

Again it is important to specify that data have been standardized to have mean = 0 and variance = 1, before the training.

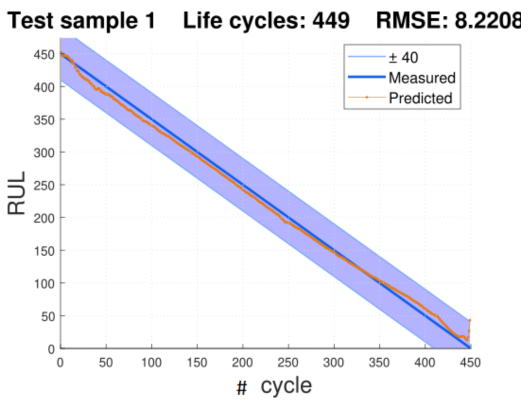
7.2.3. Results

To measure the goodness of the results we used the **Root Mean Squared Error** metric:

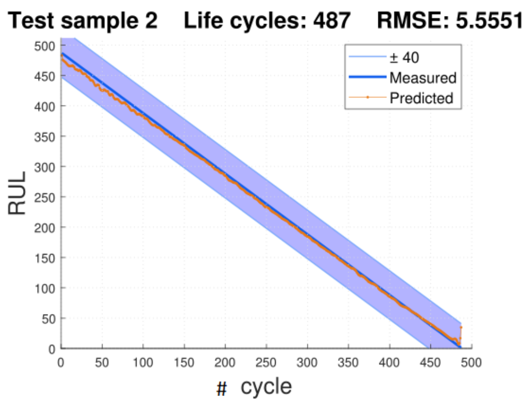
$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(\overline{y_n} - y_n)^2}{N}}$$

where $\overline{y_n}$ is the predicted value, and y_n the target. We computed RMSE over every test sequence.

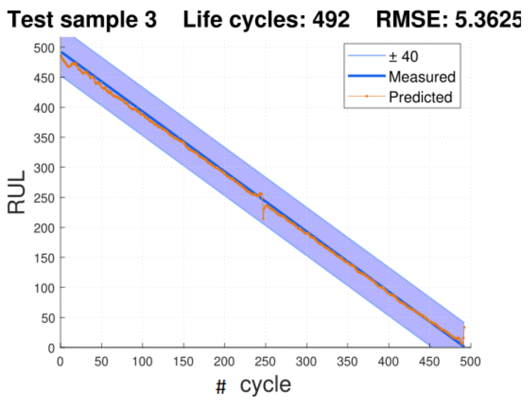
The training phase lasts around 15 minutes running on a GPU environment Nvidia GeForce 1070 with 2GB dedicated. Training time can be reduced by adjusting the dynamic schedule of the learning rate.
Prediction phase is about seconds.
Figures 7.12 and 7.13 show the prediction results for the test samples. For batteries whose life length is less then 800 cycles, the prediction is made for the entire battery life. Instead, if the battery has more then 800 cycles, the prediction is made only for the last 800 cycles.



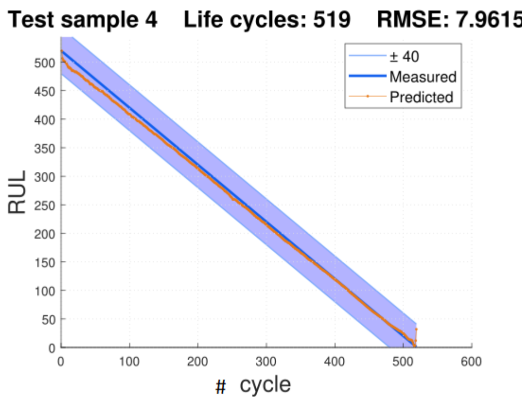
(a)



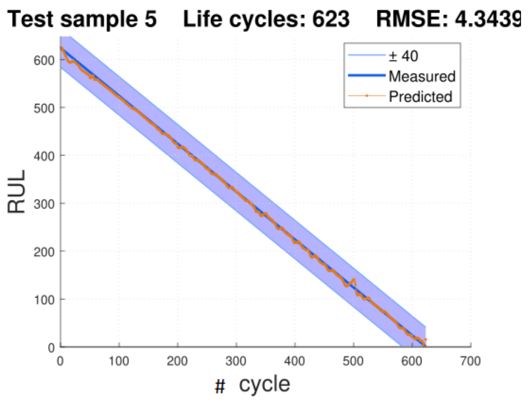
(b)



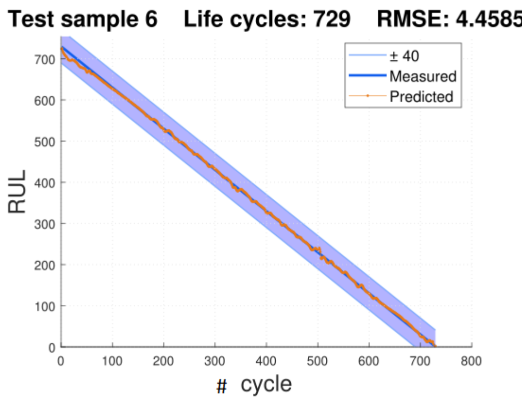
(c)



(d)

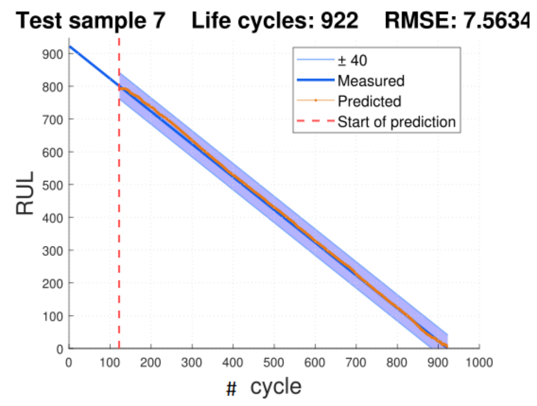


(e)

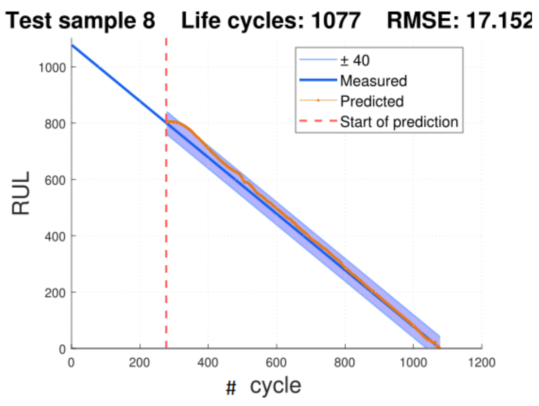


(f)

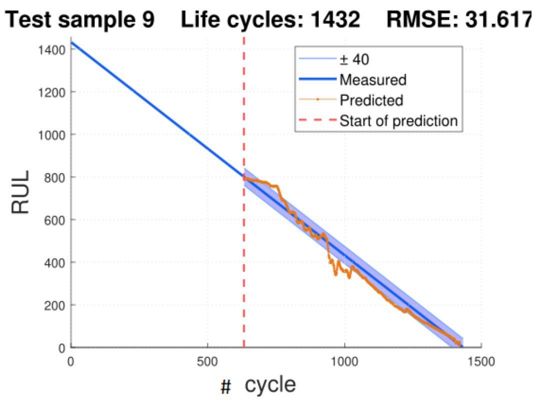
Figure 7.12: RUL prediction results, test samples 1-6



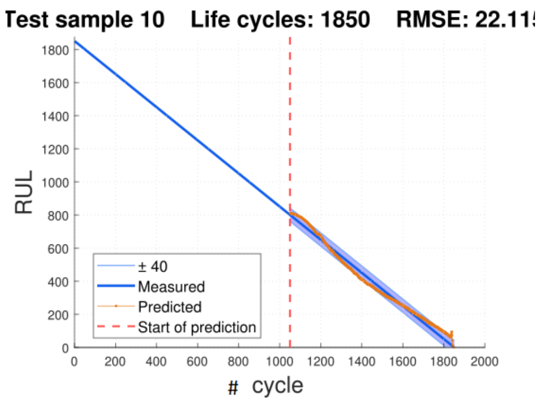
(a)



(b)



(c)



(d)

Figure 7.13: RUL prediction results, test samples 7-10

The blue line in the plot is the measured RUL, which is, for each time instant, the remaining amount of cycles until the aged batteries reach the failure cycle. Since RUL and x axis have the same unit of measure (cycles), the measured RUL is a perfect straight line.

The network successfully predicts RUL with high precision and a very low RMSE value, especially for those sequences long between 400 and 900 cycles.
Instead we see that for longer sequences the model loses a bit in accuracy, and predictions are slightly more noisy, but not so drastically, indeed RMSE values are still low and acceptable, especially in the final part, which is the most important since it is near to the failure point. We see indeed that test samples 10 and 1, which are longer then the average, increase the gap between prediction and measurements around RUL value of 400 (measured), but converge again to a fairly good prediction right around 200 RUL value (measured).
The noise in the very same predictions instead is likely due to the features we used:

the difference in the charge provided by batteries during discharge phases, in different cycles, is naturally more pronounced for those cells which age and degrade faster, instead a battery with a long life time is aged slowly, thus the feature becomes less sensitive and loses variance, and as a consequence generates more uncertain and noisy predictions.

7.3. SoH estimation: Toyota-MIT dataset

The third and last step of the thesis attempts to close the circle between RUL prediction and SoH estimation on the Toyota-MIT dataset, indeed we want to perform SoH estimation on this dataset, which is bigger and more sound than NASA dataset, and works as a litmus test for the SoH approach we presented.
Unfortunately, as we told, Toyota-MIT dataset has many different charging policies applied on its batteries, and this makes it impossible to adopt the same partial charging curve approach we had on NASA dataset because there won't be coherence among the same feature for different policy-charged batteries. Furthermore the charging policies are build to switch the charging current in the middle of the charging phase. This causes the interruptions of the charging voltage curves, creating steps in the curves themselves, as we have shown in Figure 5.4, that fake the measurements, making it hard also to find an interval of voltage without the presence of any steps for all the available batteries.

We then decided to measure **Partial Discharging Curves** instead, and from them computing the same features we used for RUL prediction.
Although this assumption is not as strong as the assumption on partial charging was, still being able to record a small enough partial discharge phase is a good compromise.

7.3.1. Partial Discharging Curve features

Similarly to what we did for the Partial Charging Curve, we analyzed the structure of the discharge curve $Q(V)$, and together with cross validation, we found an optimal interval for extracting the the 3 features described in Equation (7.7):

- **Variance**: Log10 of the variance of the curve obtained by subtracting discharge curve $Q(V)$ at cycle k to $Q(V)$ at cycle 10.
- **Minimum**: Log10 of the minimum of the curve obtained by subtracting discharge curve $Q(V)$ at cycle k to $Q(V)$ at cycle 10.
- **Temperature sum**: Summation of the average cycle temperature from cycle 2 to cycle k.

These give us 3 descriptors values for each cycle of each battery.

Instead of using the full voltage spectrum from 2 to 3.4 V, we selected the window 2.8 to 3.1 V.

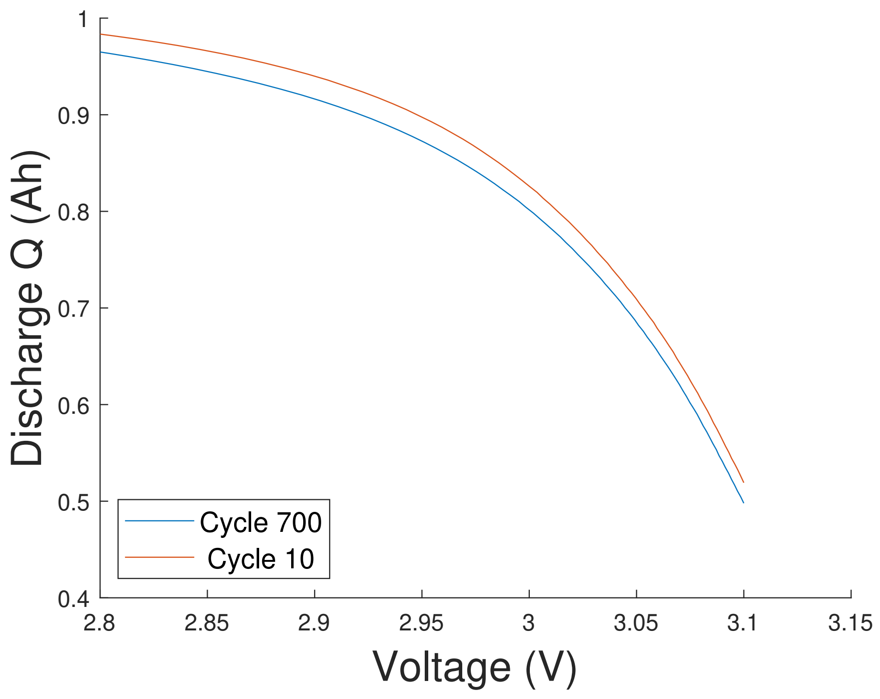


Figure 7.14: Partial discharge curve Q(V) at cycles 10 and 100

The first consideration is that by reducing the window for collecting measurements we lose a part of the information describing the variance of the data. We've seen in Figures 7.9 and 7.10, and also through PCA analysis, that the features that we called **variance** feature and **minimum** feature, if collected over the full discharge window, contain quite the same information. Instead, if **varianc** feature is collected over the considered partial discharge curve, it loses its property of linearity over the battery samples, as we can see by comparing Figure 7.15 and Figure 7.9a.
On the other hand, the **minimum** feature keeps its properties and its linear shape over the different samples, as shown in Figure 7.16.

7.3.2. Model and results

We used **Minimum** feature and **Temperature sum** feature, to train the same SVR model we used for SoH estimation on NASA dataset.
We 5-fold cross validated the model to tune the hyperparameters of the model. The procedure results are in Table 7.3.

In this case, it has been of primary importance to use a non-linear **Kernel Function** to handle the non linearity of the data and the features, in particular, **Gaussian kernel**

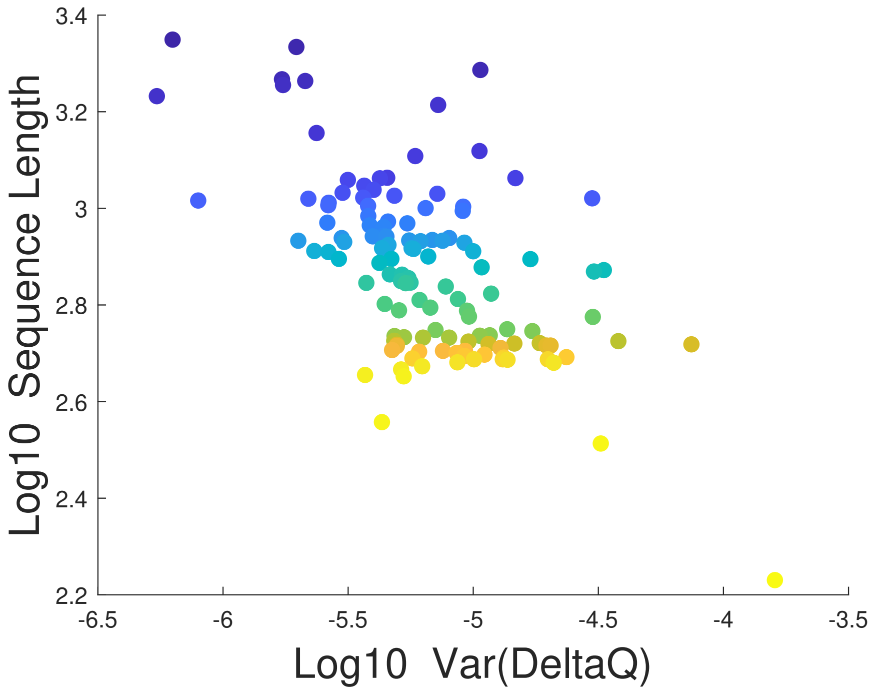


Figure 7.15: Partial (Log10) Variance of delta Q(v) at cycle 100, over all dataset samples.

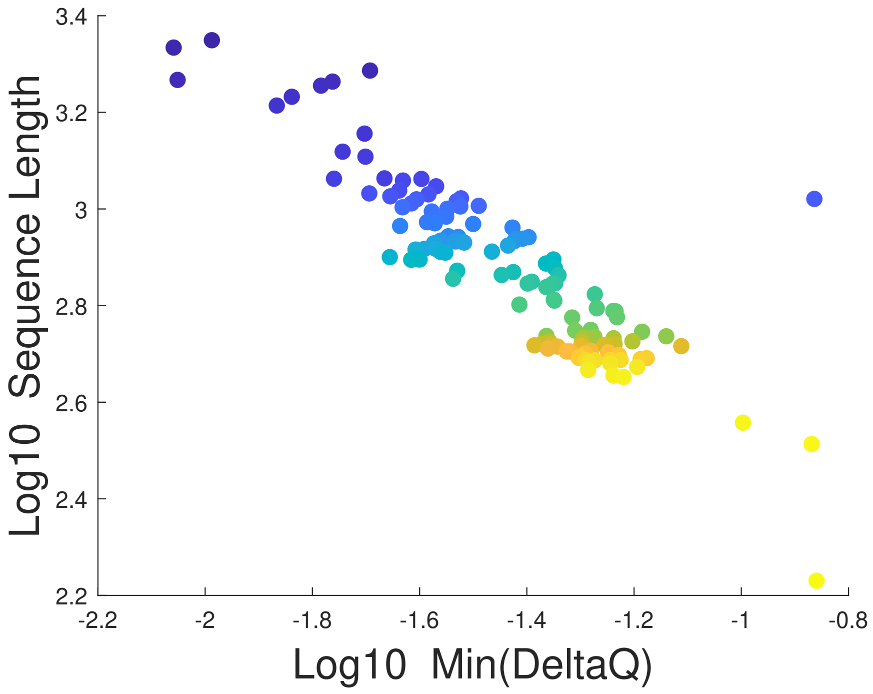


Figure 7.16: Partial (Log10) Minimum of delta Q(v) at cycle 100, over all dataset samples.

gave the best results in term of validation R^2.

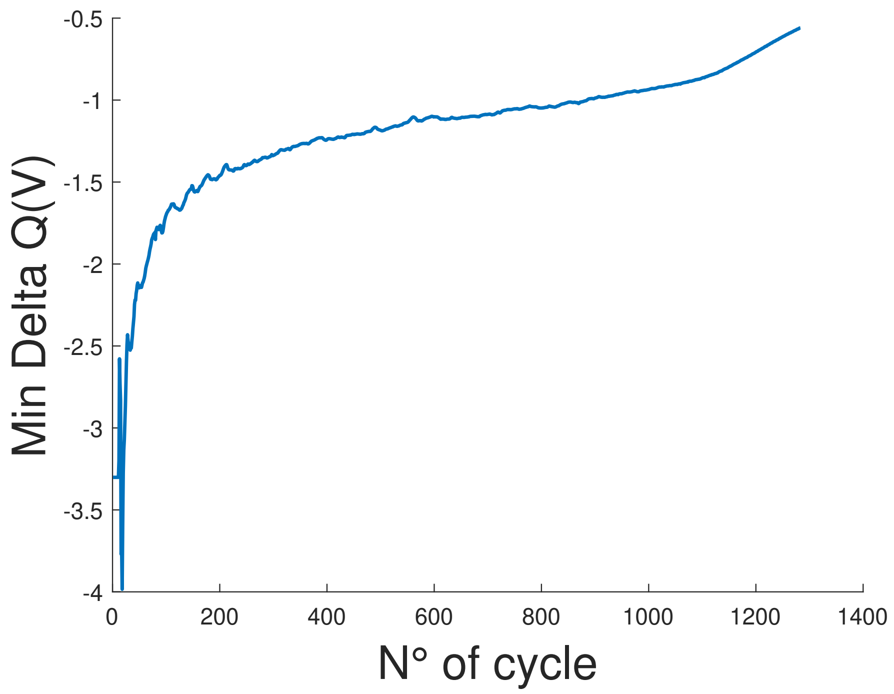


Figure 7.17: Partial (Log10) Minimum of delta Q(v) over cycles, for 1 sample battery.

SVR Hyperparameters

	Hyperparameter	**Value**
1	Box Constraint	0.1
2	Kernel Scale	1
3	Epsilon	0.006
4	Kernel Function	Gaussian

Table 7.3: contains the tuned hyperparameters values for the considered SVR model.

PCA analysis on all the 3 features shows that the third in order, that is the **Variance**, accounts for less then 10 of the total variance (Figure 7.18).
We could then remove it to reduce the complexity of the model and the training time, almost without losing any appreciable level of performance, according to the cross-validation

R^2.

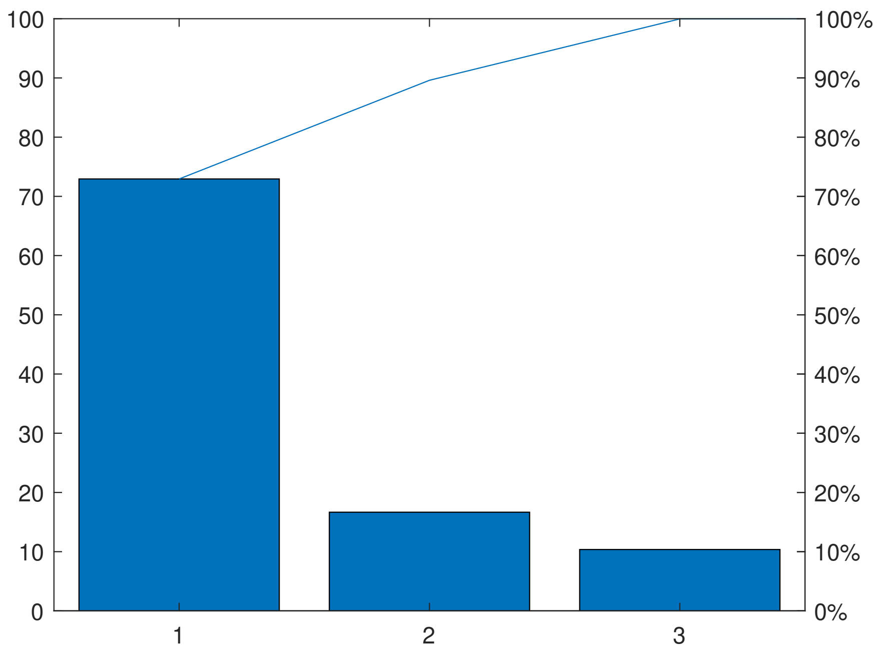
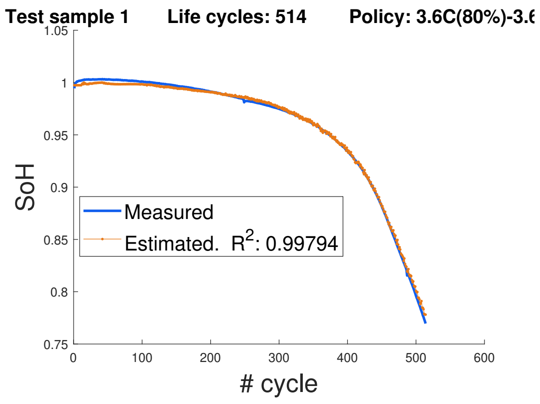


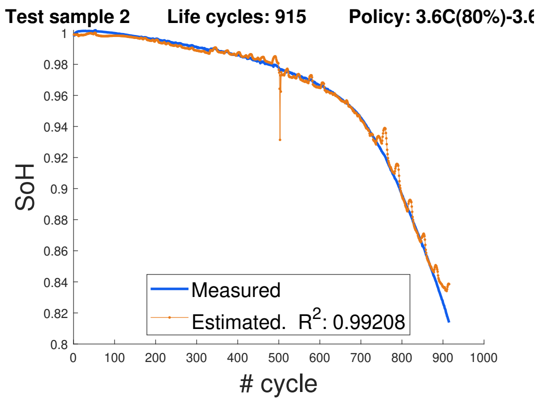
Figure 7.18: PCA analysis for all 3 features, in order: minimum, temperature sum, variance.

The overall model has been trained on 110 battery samples, and tested on 10. We achieved an average R^2 value over all the test samples of **0.97**.
Data have been standardize to have mean = 0 and variance = 1, before the training.

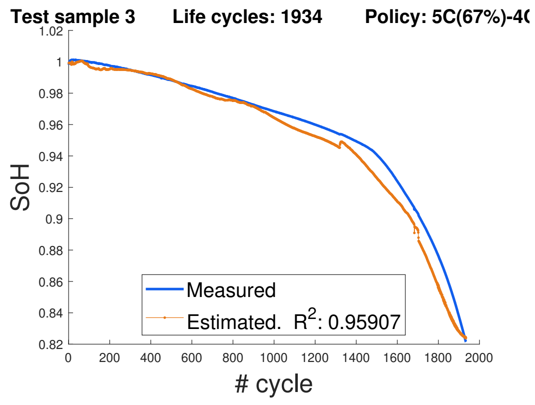
Results are shown in Figures 7.19, 7.20 and 7.21.



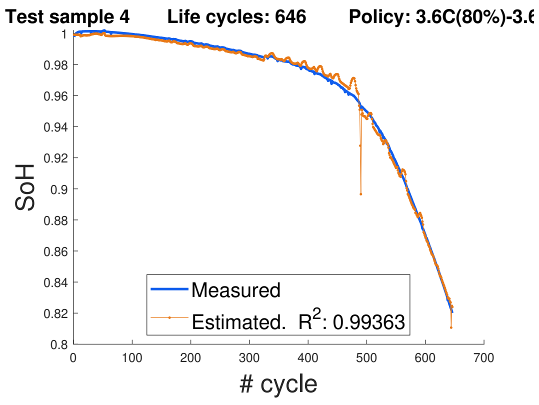
(a)



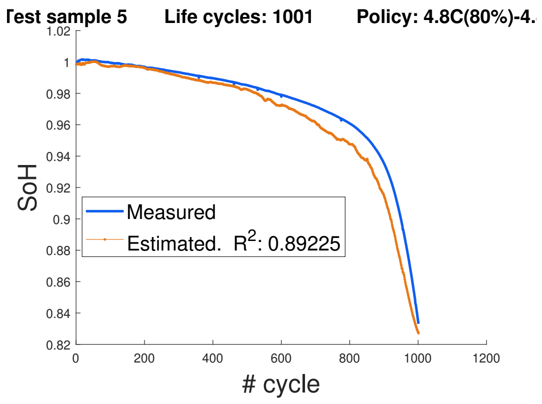
(b)



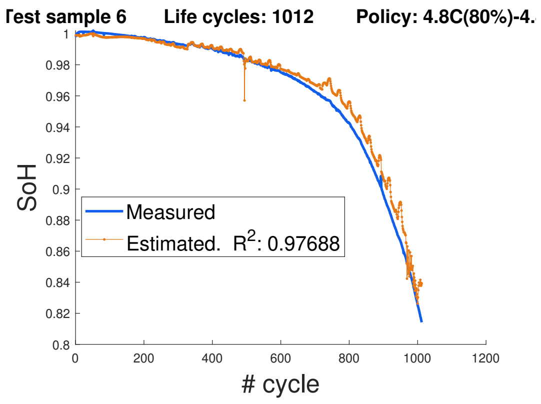
(c)



(d)

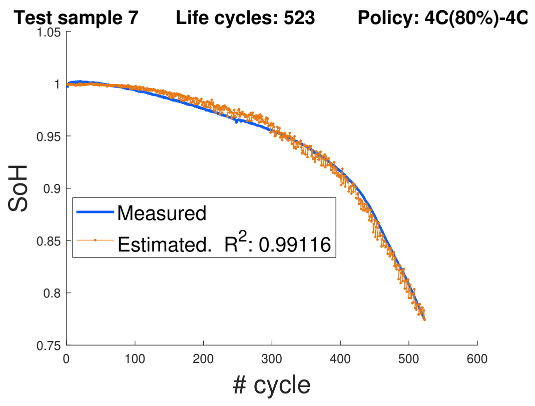


(e)

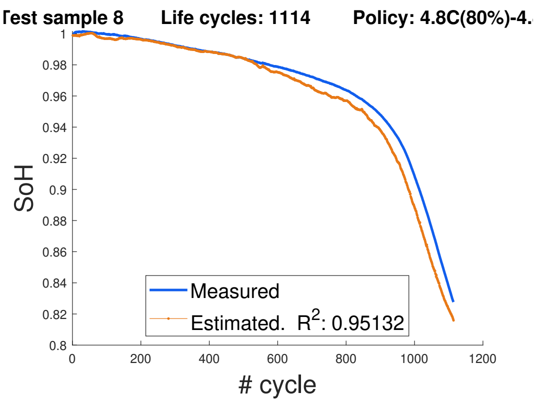


(f)

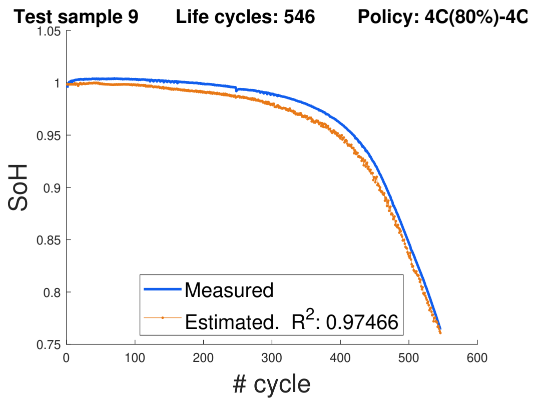
Figure 7.19: SoH estimation results, test samples 1-6



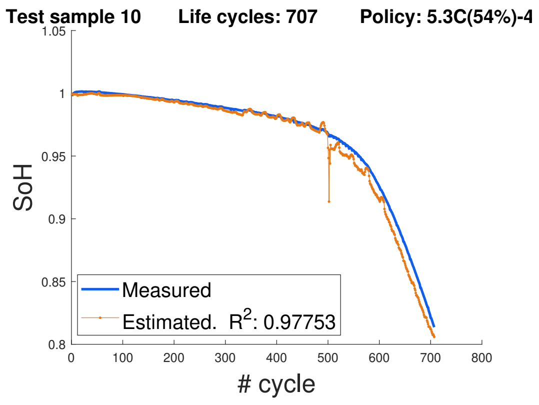
(a)



(b)



(c)



(d)

Figure 7.20: SoH estimation results, test samples 7-10

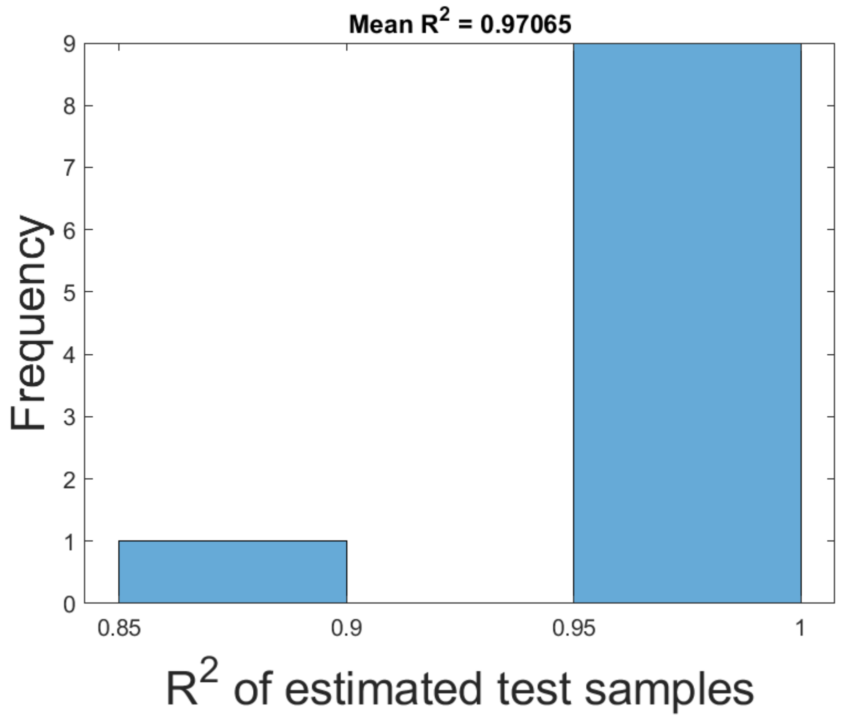


Figure 7.21: Distribution of the test samples over the estimations R2 value.

8 | Conclusions and future developments

We proposed a feature-based battery**SoH estimation** system whose results are very promising on both the datasets we considered, and the partial curve feature extraction method gave good results, for both long and short life batteries. The approach applied on the **MIT-Toyota** dataset has a bit stronger assumption though, compared to the **NASA** one: indeed is less likely that any device is discharged from 3.1 V to 2.8 V without interruptions, compared to how likely is the partial charging condition to be verified. For this reason, a future development of this work could be testing the solution with a reduction of the discharge voltage window explored, and more in general, try different variations of such voltage window, and compare results. Even better, apply the partial charging curve solution to a dataset with the same proportions of MIT-Toyota dataset, but with constant and equal charging policy for all the batteries, since NASA dataset results can only be considered validated over batteries with very similar aging among each other, due to the reduced dimension of such dataset.

RUL prediction model also produced excellent results for prediction all along the 800 cycles window considered. Indeed for the majority of the batteries, RUL prediction is accurate also at early cycles, thanks to the high population of the dataset, especially for those batteries whose life length is around the average. In fact many samples in this dataset are close to the average, leading to good predictions for such samples. Not the same can be said for batteries with longer life: even though results are still very accurate, the trend is that predictions are more noisy, and the error slightly bigger, in the earliest cycles of the predictions. Predictions anyway converge to very accurate RUL values while getting closer to the End of Life, which is the behaviour we expect RUL predictions to follow.
An idea for future development is to adopt a sliding window approach to crop the longest sequences to smallest ones, and overcome the memory limitations of LSTM Network.
Another approach to be carried out, in order to fully exploit the power of Neural Network, is to quit the feature-based approach and feed the network with sequences of raw mea-

surements instead, so that we can totally forget about any knowledge on the context. To do this, many issues have to be faced, first of all dealing with the fact that a cycle (which is the time unit that measures our time sequences) has a huge amount of measurements, that can be different from the number of measurements at all other cycles, and we need a way to **synchronize** them without having to cut out a big amount of information.

Bibliography

[1] S. Barcellona, L. Cristaldi, M. Faifer, E. Petkovski, L. Piegari, and S. Toscani. State of health prediction of lithium-ion batteries. In *2021 IEEE International Workshop on Metrology for Industry 4.0 IoT (MetroInd4.0IoT)*, pages 12–17, 2021. doi: 10.1109/MetroInd4.0IoT51437.2021.9488542.

[2] J. Brady and M. O'Mahony. Introduction of electric vehicles to ireland: Socioeconomic analysis. *Transportation Research Record*, 2242(1):64–71, 2011. doi: 10.3141/2242-08. URL `https://doi.org/10.3141/2242-08`.

[3] M. Catelani, L. Ciani, R. Fantacci, G. Patrizi, and B. Picano. Remaining useful life estimation for prognostics of lithium-ion batteries based on recurrent neural network. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021. doi: 10.1109/TIM.2021.3111009.

[4] Z. Chen, M. Sun, X. Shu, J. Shen, and R. Xiao. On-board state of health estimation for lithium-ion batteries based on random forest. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pages 1754–1759, 2018. doi: 10.1109/ICIT.2018.8352448.

[5] Z. Chen, X. Xia, M. Sun, J. Shen, and R. Xiao. State of health estimation of lithium-ion batteries based on fixed size ls-svm. In *2018 IEEE Vehicle Power and Propulsion Conference (VPPC)*, pages 1–6, 2018. doi: 10.1109/VPPC.2018.8605041.

[6] R. Darling and J. Newman. Modeling side reactions in composite li y mn2 o 4 electrodes. *Journal of The Electrochemical Society*, 145(3):990–998, mar 1998. doi: 10.1149/1.1838376. URL `https://doi.org/10.1149/1.1838376`.

[7] A. Eddahech, O. Briat, N. Bertrand, J.-Y. Delétage, and J.-M. Vinassa. Behavior and state-of-health monitoring of li-ion batteries using impedance spectroscopy and recurrent neural networks. *International Journal of Electrical Power and Energy Systems*, 42(1):487–494, 2012. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2012.04.050. URL `https://www.sciencedirect.com/science/article/pii/S0142061512001779`.

[8] T. Evgeniou and M. Pontil. Support vector machines: Theory and applications. volume 2049, pages 249–257, 01 2001. doi: 10.1007/3-540-44673-7_12.

[9] X. Feng, C. Weng, X. He, X. Han, L. Lu, D. Ren, and M. Ouyang. Online state-of-health estimation for li-ion battery using partial charging segment based on support vector machine. *IEEE Transactions on Vehicular Technology*, 68(9):8583–8592, 2019. doi: 10.1109/TVT.2019.2927120.

[10] A. Guha and A. Patra. Particle filtering based estimation of remaining useful life of lithium-ion batteries employing power fading data. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 193–198, 2017. doi: 10.1109/ICPHM.2017.7998327.

[11] Z. He, M. Gao, C. Wang, L. Wang, and Y. Liu. Adaptive state of charge estimation for li-ion batteries based on an unscented kalman filter with an enhanced battery model. *Energies*, 6:4134–4151, 08 2013. doi: 10.3390/en6084134.

[12] J. Jia, J. Liang, Y. Shi, J. Wen, X. Pang, and J. Zeng. Soh and rul prediction of lithium-ion batteries based on gaussian process regression with indirect health indicators. *Energies*, 13(2), 2020. ISSN 1996-1073. doi: 10.3390/en13020375. URL `https://www.mdpi.com/1996-1073/13/2/375`.

[13] V. Klass, M. Behm, and G. Lindbergh. A support vector machine-based state-of-health estimation method for lithium-ion batteries under electric vehicle operation. *Journal of Power Sources*, 270:262–272, 2014. ISSN 0378-7753. doi: https://doi.org/10.1016/j.jpowsour.2014.07.116. URL `https://www.sciencedirect.com/science/article/pii/S0378775314011707`.

[14] M. T. Lawder, B. Suthar, P. W. C. Northrop, S. De, C. M. Hoff, O. Leitermann, M. L. Crow, S. Santhanagopalan, and V. R. Subramanian. Battery energy storage system (bess) and battery management system (bms) for grid-scale applications. *Proceedings of the IEEE*, 102(6):1014–1030, 2014. doi: 10.1109/JPROC.2014.2317451.

[15] J. Liu and Z. Chen. Remaining useful life prediction of lithium-ion batteries based on health indicator and gaussian process regression model. *IEEE Access*, 7:39474–39484, 2019. doi: 10.1109/ACCESS.2019.2905740.

[16] K. C. F. Mailonline. Electric scooter explodes while charging in man's living room in china. [online]. 2018. URL `https://www.dailymail.co.uk/news/china/article-6014191/`.

[17] Z. Meng, Y. Hu, and C. Ancey. Using a data driven approach to predict waves generated by gravity driven mass flows. *Water*, 12, 02 2020. doi: 10.3390/w12020600.

[18] M. Moustapha, J.-M. Bourinet, B. Guillaume, and B. Sudret. Comparative Study of Kriging and Support Vector Regression for Structural Engineering Applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 4(2), June 2018. doi: 10.1061/ajrua6.0000950. URL `https://hal.archives-ouvertes.fr/hal-01893274`.

[19] J. Qu, F. Liu, Y. Ma, and J. Fan. re-network-based method for rul prediction and soh monitoring of lithium-ion battery. *IEEE Access*, 7:87178–87191, 2019. doi: 10.1109/ACCESS.2019.2925468.

[20] S. M. Rezvanizaniani, S. Lee, and J. Lee. A comparative analysis of techniques for electric vehicle battery prognostics and health management. *SAE Technical Papers*, 09 2011. doi: 10.4271/2011-01-2247.

[21] B. Saha and K. Goebel. Nasa ames prognostics data repository. *NASA Ames, Moffett Field, CA, USA*, 2007. URL `http://ti.arc.nasa.gov/project/prognosticdata-repository`.

[22] B. Saha, K. Goebel, and J. Christophersen. Comparison of prognostic algorithms for estimating remaining useful life of batteries. *Transactions of The Institute of Measurement and Control - TRANS INST MEASURE CONTROL*, 31, 06 2009. doi: 10.1177/0142331208092030.

[23] K. A. Severson, P. M. Attia, N. Jin, N. Perkins, B. Jiang, Z. Yang, M. H. Chen, M. Aykol, P. K. Herring, D. Fraggedakis, M. Z. Bazant, S. J. Harris, W. C. Chueh, and R. D. Braatz. Data-driven prediction of battery cycle life before capacity degradation. *Nature Energy*, 4(5):383–391, May 2019. ISSN 2058-7546. doi: 10.1038/s41560-019-0356-8. URL `https://doi.org/10.1038/s41560-019-0356-8`.

[24] K. A. Severson, P. M. Attia, N. Jin, N. Perkins, B. Jiang, Z. Yang, M. H. Chen, M. Aykol, P. K. Herring, D. Fraggedakis, M. Z. Bazant, S. J. Harris, W. C. Chueh, and R. D. Braatz. Data-driven prediction of battery cycle life before capacity degradation. *Nature Energy*, 4(5), 3 2019. doi: 10.1038/s41560-019-0356-8.

[25] K. Tasaki and S. J. Harris. Computational study on the solubility of lithium salts formed on lithium ion battery negative electrode in organic solvents. *The Journal of Physical Chemistry C*, 114(17):8076–8083, 2010. doi: 10.1021/jp100013h. URL `https://doi.org/10.1021/jp100013h`.

[26] S. Wang, S. Jin, D. Deng, and C. Fernandez.

[27] H. Wei, X. Chen, Z. Lü, Z. Wang, H. Pan, and L. Chen. Online estimation of lithium-ion battery state of health using grey neural network. *Dianwang Jishu/Power System Technology*, 41:4038–4044, 12 2017. doi: 10.13335/j.1000-3673.pst.2017.0522.

[28] N. Williard, W. He, C. Hendricks, and M. Pecht. Lessons learned from the 787 dreamliner issue on lithium-ion battery reliability. *Energies*, 6(9):4682–4695, 2013. ISSN 1996-1073. URL `https://www.mdpi.com/1996-1073/6/9/4682`.

[29] Z. Xue, Y. Zhang, C. Cheng, and G. Ma. Remaining useful life prediction of lithium-ion batteries with adaptive unscented kalman filter and optimized support vector regression. *Neurocomputing*, 376:95–102, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.09.074. URL `https://www.sciencedirect.com/science/article/pii/S0925231219313426`.

[30] L. Yao, S. Xu, A. Tang, F. Zhou, J. Hou, Y. Xiao, and Z. Fu. A review of lithium-ion battery state of health estimation and prediction methods. *World Electr. Veh. Journal*, pages 4–5, 2021.

[31] L. Zhao, Y. Wang, and J. Cheng. A hybrid method for remaining useful life estimation of lithium-ion battery with regeneration phenomena. *Applied Sciences*, 9(9), 2019. ISSN 2076-3417. URL `https://www.mdpi.com/2076-3417/9/9/1890`.

[32] X. Zheng and H. 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 and System Safety*, 144:74–82, 2015. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2015.07.013. URL `https://www.sciencedirect.com/science/article/pii/S0951832015002148`.

ACKNOWLEDGEMENTS

Acknowledgements

I Would like to thank Ing. Emil Petkovski, Prof. Loredana Cristaldi, Prof. Francesco Amigoni, and Ing. Davide Azzalini for their help in the development of this thesis.