A project report submitted in partial fulfilment of the requirement

for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

ADUPALA ANKITHA (160720733016)

P. ADITYA VISHWA TEJA (160720733048)

P. KHAJA MUNEER (160720733005)

Under the Guidance of

Dr. M. Sharada Varalakshmi,

Professor, Department of CSE

Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

2023-2024

PREDICTION OF REMAINING USEFUL LIFE OF LITHIUM-ION BATTERY USING MACHINE LEARNING

A project report submitted in partial fulfillment of the requirement for the Award of the Degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

by

ADUPALA ANKITHA (160720733016)

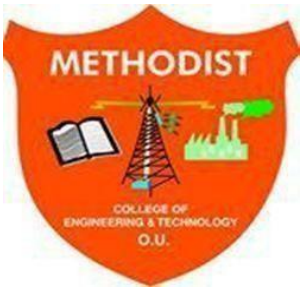
P. ADITYA VISHWA TEJA (160720733048)

P. KHAJA MUNEER (160720733005)

Under the Guidance of

Dr. M. Sharada Varalakshmi,

Professor, Department of CSE



Department of Computer Science and Engineering

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

2023-2024

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001,

Department of Computer Science and Engineering

DECLARATION BY THE CANDIDATES

We, **ADUPALA ANKITHA (160720733016), P. ADITYA VISHWA TEJA (160720733048)** and **P. KHAJA MUNEER (160720733005)** students of Methodist College of Engineering and Technology, pursuing Bachelor's degree in Computer Science and Engineering, hereby declare that this Project report entitled "**PREDICTION OF REMAINING USEFUL LIFE OF LITHIUM-ION BATTERY USING MACHINE LEARNING",** carried out under the guidance of **DR. M. SHARADA VARALAKSHMI** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science. This is a record work carried out by us and the results embodied in this report have not been reproduced/copied from any source.

Adupala Ankitha (160720733016)

P. Aditya Vishwa Teja (160720733048)

P. Khaja Muneer (160720733005)

Date:

Methodist College of Engineering and Technology,

King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

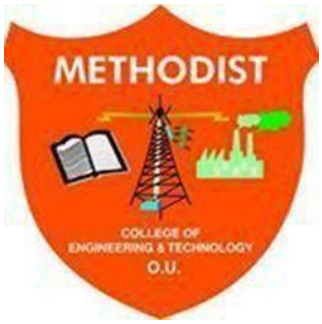
This is to certify that this Dissertation on project work entitled "**PREDICTION OF REMAINING USEFUL LIFE OF LITHIUM-ION BATTERY USING MACHINE LEARNING**" *by* **ADUPALA ANKITHA (160720733016), P. ADITYA VISHWA TEJA (160720733048) AND P. KHAJA MUNEER (160720733005)** submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2023-2024, is a bonafide record of work carried out by them.

Dr. M. Sharada Varalakshmi
Professor,
Department of CSE

Date:

Methodist College of Engineering and Technology,
King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering



CERTIFICATE BY HEAD OF THE DEPARTMENT

This is to certify that this Dissertation on project work entitled "**PREDICTION OF REMAINING USEFUL LIFE OF LITHIUM-ION BATTERIES USING MACHINE LEARNING**" *by* **ADUPALA ANKITHA (160720733016), P. ADITYA VISHWA TEJA (160720733048) AND P. KHAJA MUNEER (160720733005)** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2023-2024, is a bonafide record of work carried out by them.

Dr. P. LAVANYA,
Professor &
Head, Department of CSE.

Date:

Methodist College of Engineering and Technology,
King Koti, Abids, Hyderabad-500001.

Department of Computer Science and Engineering

PROJECT APPROVAL CERTIFICATE

This is to certify that this Dissertation on project work entitled "**PREDICTION OF REMAINING USEFUL LIFE OF LITHIUM-ION BATTERIES USING MACHINE LEARNING**" *by* **ADUPALA ANKITHA (160720733016), P. ADITYA VISHWA TEJA (160720733048) AND P. KHAJA MUNEER (160720733005),** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2023-2024, is a bonafide record of work carried out by them.

INTERNAL **EXTERNAL** **HOD**

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to my Project guide **Dr. M. Sharada Varalakshmi,**

Professor, Department of CSE, for giving us the opportunity to work on this topic. It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

We would like to thank our project coordinator **Dr. T. Praveen Kumar, Associate Professor, Department of CSE,** who helped us by being an example of high vision and pushing towards greater limits of achievement.

Our sincere thanks to **Dr. P. Lavanya, Professor** and **Head of the Department of Computer Science and Engineering,** for her valuable guidance and encouragement which has played a major role in the completion of the project and for helping us by being an example of high vision and pushing towards greater limits of achievement.

We would like to express a deep sense of gratitude towards the **Dr. Prabhu G. Benakop, Principal, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We would like to express a deep sense of gratitude towards the **Dr. Lakshmipathi Rao, Director, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We are indebted to the Department of Computer Science & Engineering and Methodist College of Engineering and Technology for providing us with all the required facility to carry our work in a congenial environment. We extend our gratitude to the CSE Department staff for providing us to the needful time to time whenever requested.

We would like to thank our parents for allowing us to realize our potential, all the support they have provided us over the years was the greatest gift anyone has ever given us and also for teaching us the value of hard work and education. Our parents have offered us with tremendous support and encouragement, thanks to our parents for all the moral support and the amazing opportunities they have given us over the years.

METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Approved by AICTE, Affiliated to Osmania University, - College Code – 1607

Department of Computer Science & Engineering
Vision & Mission

VISION

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

MISSION

M1: To offer flexible programs of study with collaborations to suit industry needs

M2: To provide quality education and training through novel pedagogical practices

M3: To Expedite high performance of excellence in teaching, research and innovations.

M4: To impart moral, ethical valued education with social responsibility.

Program Educational Objectives

Graduates of Compute Science and Engineering at Methodist College of Engineering and Technology will be able to:

PEO1:	Apply technical concepts, Analyse, synthesize data to Design and create novel products and solutions for the real-life problems.
PEO2:	Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.
PEO3:	Promote collaborative learning and spirit of team work through multidisciplinary projects
PEO4:	Engage in life-long learning and develop entrepreneurial skills.

Program Specific Outcomes

At the end of 4 years, Compute Science and Engineering graduates at MCET will be able to:

PSO1:	Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.
PSO2:	Develop software applications with open-ended programming environments**.**
PSO3:	Design and develop solutions by following standard software engineering principles and implement by using suitable programming languages and platforms

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Abstract

Automobile vehicles, essential for commuting, goods transportation, travel services, and more, predominantly use hydrocarbon fuels like petrol and diesel. These traditional vehicles, while widely used, contribute significantly to air pollution, global warming, and fossil fuel depletion due to their emission of greenhouse gases such as carbon dioxide, nitrous oxide, and methane. In response, the automotive industry is shifting towards environmentally friendly alternatives, primarily electric vehicles powered by lithium-ion batteries (LIBs). However, LIBs have a limited lifespan, and their end-of-life usage can lead to severe consequences, including catastrophic explosions. Additionally, range anxiety remains a significant barrier to electric vehicle adoption. Thus, accurately predicting the remaining useful life (RUL) of LIBs is crucial. Advanced algorithms, such as machine learning models, are employed to predict the RUL of these batteries, ensuring safety and enhancing consumer confidence in electric vehicles.

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

- **AED-SDA** - Average Euclidean Distance Stacked Denoising Autoencoder
- **ANN** - Artificial Neural Network
- **ARIMA** - Auto Regressive Integrated Moving Average
- **BATPFNN** - Bat based Particle Filter Neural Network
- **BOL** - Beginning of Life
- **BMS** - Battery Management System
- **BPSO-SVM** - Binary Particle Swarm Optimization - Support Vector Machine
- **CALCE** - Center for Advanced Life Cycle Engineering
- **CC** - Coulomb Counting
- **CCCV** - Constant Current Constant Voltage
- **CNN** - Convolutional Neural Network
- **CTGAN** - Conditional Generative Adversarial Network
- **DCNN** - Deep Convolutional Neural Network
- **DCNN-EL** - Deep Convolutional Neural Network with Ensemble Learning
- **DCNN-TL** - Deep Convolutional Neural Network with Transfer Learning
- **DOD** - Depth of Discharge
- **DNN** - Deep Neural Network
- **DTW** - Dynamic Time Warping
- **EDA** - Exploratory Data Analysis
- **EIS** - Electrochemical Impedance Spectroscopy
- **ELM** - Extreme Learning Machine
- **EOL** - End of Life
- **EV** - Electric Vehicle
- **GA** - Genetic Algorithm
- **GPR** - Gaussian Process Regression
- **GRU-GPR** - Gated Recurrent Unit kernel - Gaussian Process Regression
- **IBL** - Instance Based Learning
- **KNN** - k Nearest Neighbors
- **LCO** - Lithium Cobalt
- **LIB** - Lithium-ion Battery
- **LSSVM** - Least Squared Support Vector Machine
- **LSTM** - Long Short-Term Memory
- **MLE** - Maximum Likelihood Estimation
- **NASA** - National Aeronautics and Space Administration
- **OCV** - Open Circuit voltage

- **OEM** - Original Equipment Manufacturer
- **PCoE** - Prognostics Center of Excellence
- **PDF** - Probability Density Function
- **PF** - Particle Filter
- **PHM** - Prognostics and Health Management
- **RDR** - Remaining Driving Range
- **RNN** - Recurrent Neural Network
- **RTF** - Run to Failure
- **RUL** - Remaining Useful Life
- **RVM** - Relevance Vector Machine
- **SEI** - Solid Electrolyte Interphase
- **SBM** - Similarity based Model
- **SOC** - State of Charge
- **SOH** - State of Health
- **SVM** - Support Vector Machine

Chapter 1

Introduction

The main area of focus of this research is to predict the remaining useful life (RUL) of a lithium- ion battery (LIB) onboards an electric vehicle (EV) in real-time. The remaining useful life of a lithium-ion battery is affected by various factors such as voltage, current, temperature, state of charge (SOC), state of health (SOH), resting duration, etc. In this research, we implement machine learning models that predict the RUL of a lithium-ion battery. Additionally, we also show that duration of rest of a battery has a significant effect on its RUL.

In Section 1.1, we motivate the importance of this study.
In Section 1.2, we define the scope of this study.
In Section 1.3, we briefly discuss the research approach and methodology used in this research.
In Section 1.4, we present the outline of this thesis.

1.1 Motivation

Conventional automobile vehicles, also known as internal combustion engine vehicles (ICEV), use hydrocarbon fuels as their source of energy. These vehicles produce poisonous gases such as carbon dioxide, nitrous oxide and methane, which are the byproducts of hydrocarbon fuel combustion. The emission of these gases has raised concerns about air pollution and global warming. Extensive use of these vehicles has also resulted in fossil fuel depletion. Thus, there is huge demand for automobile vehicles that run on alternative green energy such as electricity.

Electric Vehicles (EVs) are becoming increasingly popular over the years. These vehicles are environmentally friendly as they run on green energy. Many automobile manufacturers have started manufacturing electric vehicles and over the years there is a growing demand for electric vehicles. EVs generally use lithium-ion batteries as their energy source.

When electric batteries are used in real-time, they are subjected to various factors such as temperature changes, load changes, etc., which directly affect their degradation. Lithium-ion batteries have a limited lifetime and cannot be used after they have reached their end of life. Using a lithium-ion battery after it has reached its end of life can be dangerous. It can lead to a battery fire incident or the battery may explode (P. Sun et al., 2020). While most of these fire incidents are reported to have occurred during the

charging of the battery, recently it was reported that an electric vehicle caught fire while the vehicle was running (Herh, 2021). These battery fires are lethal to the drivers and the passengers. Additionally, the harmful gases released during these fire incidents are also poisonous (P. Sun et al., 2020).

Thus, to prevent catastrophic battery failure, accurate and timely prediction of remaining useful life of a battery in an EV can update the driver and then the driver can take necessary action.

1.1 Scope

In this research, the research problem is to predict the RUL of the LIB in an EV. In this study, we will implement the following RUL prediction models - Support Vector Regression (SVR), Long Short-Term Memory (LSTM) and Similarity based model (SBM). In this research, we consider the effects of battery resting (calendar aging) on the RUL of LIBs. The data set used for this research is having the data of eight LIBs. In this research, we are interested in implementing RUL prediction models that return the point estimate of the RUL. The models developed in this research work only on the data set provided by TNO and it may not work for a different data set.

1.2 Research Approach and Methodology

To develop machine learning models to predict the RUL of an LIB onboard an EV in real-time, relevant battery health indicators and/or stress factors have to be selected from the data set. In prognostics and health management (PHM), the health indicators of a system, in general, are the properties of the system through which one can infer the present health of the system. The RUL of a battery is directly impacted by the health of the battery. In general, a battery with higher health would generally have a higher RUL as compared to a battery with lower health.

Also, appropriate evaluation metrics have to be selected to evaluate the performance of the machine learning models. The task of predicting the RUL of LIB on-board an EV in real-time can be categorized as a regression problem, as the predicted RUL is a continuous value. Hence, evaluation metrics such as mean absolute error (MAE), mean absolute percentage error (MAPE), root mean squared error (RMSE) and coefficient of determination (R^2) are used to evaluate the different RUL prediction models that will be implemented as part of this research. Also, the running time of a model is an extremely important factor as the RUL prediction model will be used in real time. An ideal RUL prediction model will typically take as less time as possible to generate the prediction.

1.3 Thesis Outline

This thesis report is organized as follows. Chapter 2 covers preliminaries where we discuss about the basic working principle of a lithium-ion battery. We also discuss briefly about battery aging, lithium-ion battery form factors and battery management systems. We also introduce the battery related terminology that are used throughout this report.

1.4 Preliminaries

A good understanding of the structure and working of a lithium-ion battery is crucial in under- standing the various factors that affect the RUL of the battery. In this chapter, several definitions and terminologies related to LIBs are introduced which will be used throughout this thesis.
In Section 1.1.1, we describe the basic working principle of a lithium-ion battery. In Section 1.1.2, we describe briefly the various LIB form factors. We define various terms and definitions related to battery In Section 1.1.3.
In Section 1.2, we describe regarding battery management systems (BMS).
In Section 1.3, we briefly describe about the aging in lithium-ion batteries.

1.5 Lithium-ion batteries

For an electric vehicle, the battery is the main source of energy. When EVs were first introduced, they used lead-acid batteries as their energy source. The energy density of a lead-acid battery is around 30 Wh/kg and in general, a lead-acid battery has a cycle life of around 300 cycles. The energy density of lead-acid battery is very less when compared to the energy density of gasoline used in internal combustion engine vehicles (ICEV) which is more than 10000 Wh/kg (Xiong, 2019). Hence, to match the performance of an ICEV, an EV with a lead-acid battery occupies more space and it also increases the mass of the vehicle. Also, lead-acid batteries result in lead emissions which are harmful to the environment.
The above shortcomings of a lead acid battery are overcome by a lithium-ion battery. The ad- vantages of using lithium-ion batteries in EVs are as follows (Xiong, 2019).

1. Lithium-ion batteries (LIB) have very high specific energy (about 200 Wh/kg). It is much higher compared to lead-acid batteries.
2. Lithium-ion batteries have a longer cycle life (about 2000) than lead acid batteries. LIBs, when operated under a low depth of discharge, can be cycled more than 2000 cycles.
3. LIBs have a very low self-discharge rate (about 6% - 8%).

4. They are environmentally friendly. They are non-polluting batteries.
5. They have a high working voltage of (about 3.6 V).
6. LIBs can be manufactured in various form factors and thus, they easily meet the battery layout requirements in EVs. Further details about these form factors are discussed in Section 1.1.2.

They can be charged or discharged at any time and this does not affect their performance. Thus, lithium-ion batteries are memoryless.
The smallest packaged unit of lithium-ion battery that is manufactured is known as a *cell*. A single cell cannot power an EV. Hence, many cells are connected either in series or in parallel to form a *module*. Several modules are then connected either in parallel or in series to form a *battery pack*. A battery pack can supply enough power to power an EV (Team, 2008).

1.5.1 Functioning of a lithium-ion battery

The basic structure and working of a lithium-ion battery is illustrated in Figure 1.1. Both the positive electrode and the negative electrode of the battery are immersed in the electrolyte. The charging and discharging of the lithium-ion battery is achieved by inserting and releasing the lithium ions between the positive and negative electrodes.

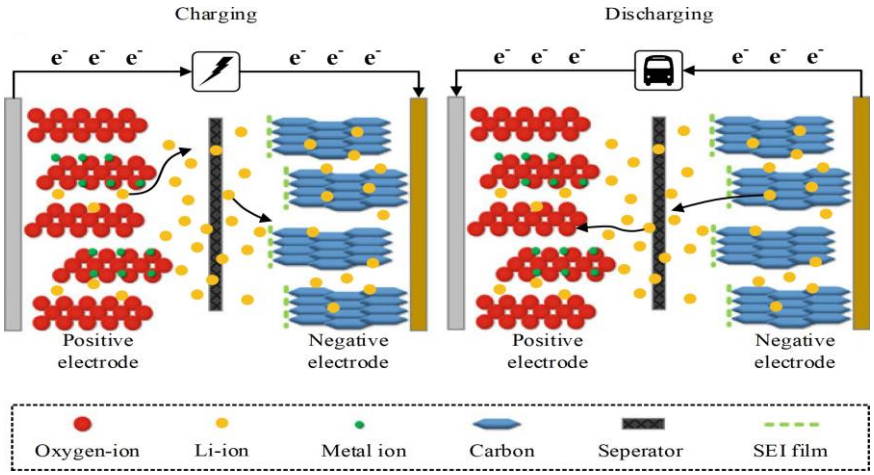


Figure 1.1: Illustration of the working principle of a Lithium-ion battery

During charging, the lithium ions move from the positive electrode into the electrolyte. The lithium ions present in the electrolyte are driven to move toward the negative electrode as the lithium ions in the electrolyte have a large concentration difference near the positive and negative electrodes. Finally, the lithium ions are embedded in the negative electrode through the separator. Meanwhile, the electrons in the external charging circuit move from the positive electrode to the negative electrode to form a current. Reduction occurs at the negative electrode as it receives electrons and oxidation occurs at the positive electrode as it loses electrons.

When a LIB is charged for the very first time, a solid interface film is formed between in the electrolyte near the negative electrode. This film is called solid electrolyte interphase (SEI) film. It prevents the negative electrode from corrosion and it also prevents the reduction reaction between the electrolyte and the negative electrode. The SEI film constantly thickens or dissolves over time. This results in a decrease in the number of lithium-ions and active materials available for recycling. This is the primary reason for battery capacity decline.

1.5.2 Lithium-ion battery form factors

Lithium-ion batteries that are used in EVs are manufactured mainly in three form factors - *cylindrical cell, prismatic cell and pouch cell*. Each form factor has its own advantages and disadvantages. Different EV original equipment manufacturers (OEM) use different LIB form factors and no form factor satisfies all the requirements of the OEM. In this section, we briefly discuss about the different LIB form factors.

1.5.2.1 Cylindrical cell

A cylindrical cell is one of the commonly used lithium-ion battery form factors. It comprises sheet- shaped anodes, separators and cathodes. All these components are sandwiched, rolled and are packed within a cylinder-shaped container (Arar, 2020). This form factor of lithium-ion battery is one of the earliest form factors that was produced in bulk. The structure of a cylindrical cell is illustrated in Figure 1.2.

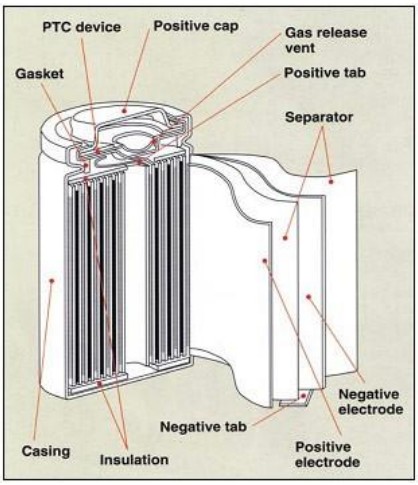


Figure 1.2: Structure of a cylindrical cell.

A Cylindrical cell has several advantages. This form factor is well suited for automated manufacturing (Arar, 2020). The cylindrical shape of the cell evenly distributes the internal pressure generated from the side reactions over the cell surface. This feature

makes cylindrical cell form factor withstand a much higher internal pressure while still being intact (Arar, 2020).

The circular cross-section of the cylindrical form factor does not allow to completely utilize the battery space. Hence, the packing density of this form factor is low. However, because of the presence of cavities between the cells, this form factor allows for better thermal management.

1.5.2.2 Prismatic cell

A prismatic cell consists of large sheets of anodes, separators and cathodes sandwiched and rolled up similar to a cylindrical cell. However, the components of the prismatic cell are fit within a metallic or a plastic container in cubic form (Arar, 2020). The structure of a prismatic cell is illustrated in Figure 1.3.
The parts of the separator sheet and electrode that are close to the corners of the cubic container experience more stress than others. This can lead to damage of the electrode coating and may lead to uneven electrolyte distribution.
Since the prismatic cells are enclosed within a cubic container, cells of this form factor can be packed together compactly. Unlike cylindrical cells, prismatic cells completely utilize the battery space. However, this compact packing requires a complex thermal management system as there are no cavities between the cells in the pack.

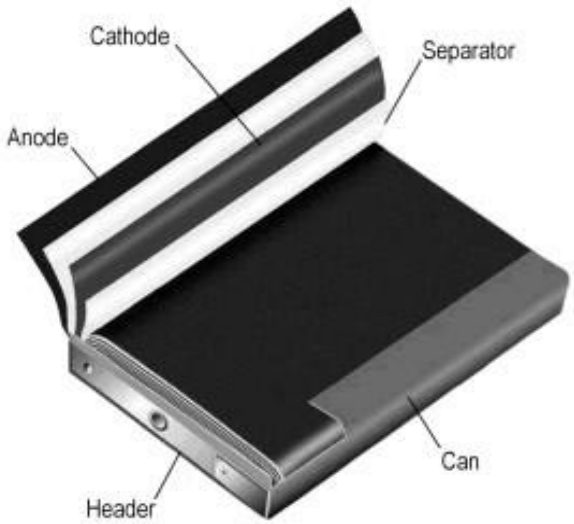


Figure 1.3: Structure of a prismat

1.5.2.2 Pouch cell

A pouch cell is another commonly used battery form factor. In a pouch cell, the separator layers and the electrode are stacked atop one another unlike cylindrical cells and prismatic cells where the anode, cathode and separator layers are sandwiched and rolled up. The contents of the pouch cell are enclosed within a container that is not rigid. The components of a pouch cell are packed and sealed within a flexible foil that acts as a cell container. The structure of a pouch cell is illustrated in Figure 1.4.

Figure 1.4: Structure of a pouch cell.

1.5.3 Battery related terminology

In this section, we define some of the commonly used battery related terminology.

Beginning of Life (BOL): The beginning of life (BOL) of a battery is the time point at which the battery has just started operating.

End of Life (EOL): The end of life (EOL) of a battery is the time point since the beginning of life (BOL) of the battery after which the battery does not operate as per its specifications.

State of Charge (SOC): SOC of a battery is the available capacity of the battery. SOC is generally expressed as a ratio of the available battery capacity to the maximum capacity. SOC is computed based on the values of current, temperature and voltage (Omariba, L. Zhang and D. Sun, 2018).

Depth of Discharge (DOD): The percentage of the battery capacity that is discharged during a load cycle is known as depth of discharge (DOD). Typically, DOD is expressed as percentage of the maximum capacity (Omariba, L. Zhang and D. Sun, 2018).

State of Health (SOH): SOH of a battery is the ratio of its present maximum charge capacity to the maximum charge capacity at its beginning of life (Omariba, L. Zhang and D. Sun, 2018). SOH is computed as shown in equation 1.1

$$SOH = \frac{Qact}{Q_R} * 100\% \quad (2.1)$$

where Q_{act} is the actual capacity of the battery and Q_R is its rated capacity.

State of Power (SoP): The state of power (SOP) of a battery is the ratio of peak power to nominal power (Xiang et al., 2018). The peak power of a battery is defined as the power that the battery can deliver persistently within the specified voltage, SoC, current and power constraints for T seconds (Plett, 2004).

Remaining useful life (RUL): The remaining time or the remaining number of load cycles the battery has during which it will be able to maintain its operating requirements is known as the remaining useful life (RUL) of a battery. Alternatively, RUL is also the length of time from the present time to the end of life (Omariba, L. Zhang and D. Sun, 2018). When the battery is used in an electric vehicle (EV), the remaining useful life of the battery can also be expressed in terms of the remaining distance that the battery can power the electric vehicle. RUL of a battery when expressed as a remaining distance for which it can power an EV before reaching the end of life (EOL) is known as remaining driving range (RDR). RUL is computed as follows.

$$\boldsymbol{RUL} = \boldsymbol{T}_{EOL} - \boldsymbol{T}_{present} \quad (2.2)$$

where $\boldsymbol{RUL}$ represents the current remaining useful life of the battery, $\boldsymbol{T}_{EOL}$ represents the time at end of life $(\mathbf{EOL})$ of the battery, $\boldsymbol{T}_{present}$ represents the current time instance at which the $\boldsymbol{RUL}$ is computed.

1.5 Battery management system

A battery management system (BMS) is a system present in an electric vehicle (EV) that manages and ensures proper functioning of the battery. The schematic diagram of a BMS is shown in Figure 1.5. BMS verifies and ensures the reliability and safety of the battery. It also takes necessary actions on the battery when it detects irregular conditions. A battery management system contains a sampling unit that is responsible for the collection of battery temperature, voltage, current and other required information from the battery pack as well as from the individual cells within the battery pack. BMS also contains several computation algorithms in the control

circuit which estimate the state of health (SOH), state of charge (SOC), state of power (SOP) and remaining useful life (RUL) of the battery (Xiong, 2019).

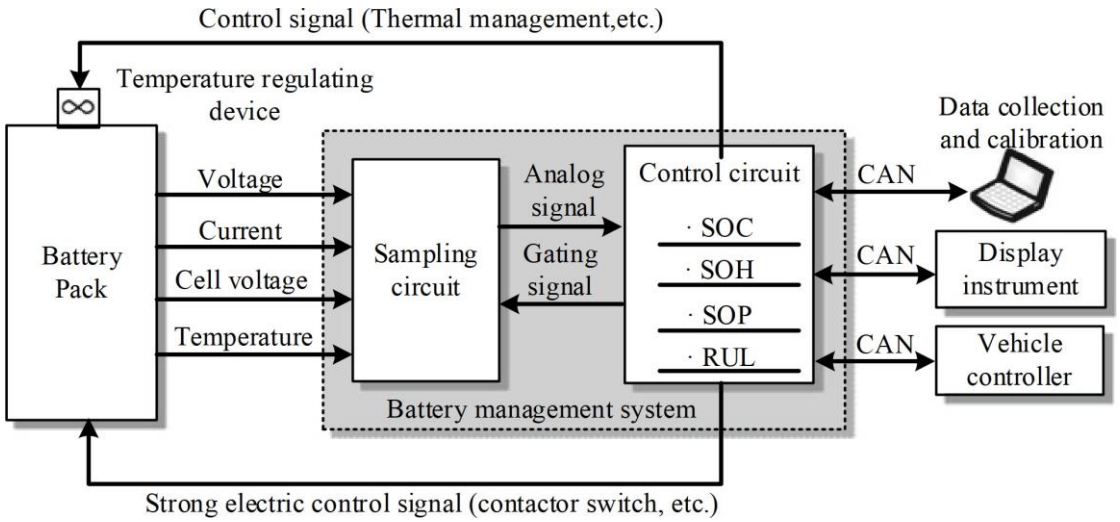


Figure 1.5: A schematic diagram of a Battery Management System

Some of the basic functionalities of a BMS are battery state monitoring, data acquisition, energy management, thermal management, battery state monitoring and battery data management (Xiong, 2019).

1.6 Battery ageing

Lithium-ion batteries have a limited useful life and they undergo irreversible capacity loss when subjected to repeated charging and discharging and also when kept idle. The capacity loss of a lithium-ion battery due to repeated charging and discharging is called as cyclic aging. The capacity loss of a lithium-ion battery when it is at rest is called calendar aging.

1.7 Types of Aging in Batteries

1.7.1 Cyclic Aging

The irreversible capacity loss that a lithium-ion battery undergoes when it is discharged and charged repeatedly is called cyclic aging. The capacity loss is dependent on the battery's state of charge (SOC), voltage, current and temperature. As a result, several factors play a role in this type of ageing. High operating temperatures speeds up the degradation process (Barr´e et al., 2013).

1.7.2 Calendar Aging

Calendar ageing refers to any ageing processes that cause a battery cell to degrade without being subjected to charge-discharge cycle. It is an important concern in EVs as most of the times the EVs will be in rest. The growth of passivation layers at the electrode–electrolyte interfaces is the major cause of calendar ageing (Keil et al., 2016)

Chapter 2

Literature Review

2.1 LITERATURE SURVEY

S.no	Authors	Title	Methodology	Datasets	Scope	Evaluation Parameters
1	Wang et al.	Remaining Useful Life Prediction of Li-ion Batteries Based on Recurrent Neural Networks	Recurrent Neural Networks (RNN)	NASA Prognostics Center of Excellence (PCoE)	RUL prediction of Li-ion batteries	RMSE, MAE
2	Zhang et al.	A Review on RUL Prediction Techniques for Li-ion Batteries	Literature review	Various	Review of RUL prediction techniques	N/A
3	Liu et al.	Predicting Remaining Useful Life of Li-ion Batteries Using Artificial Neural Networks	Artificial Neural Networks (ANN)	Laboratory cycling data	RUL estimation for EV applications	RMSE, MAE

4	He et al.	A Data-Driven Method for Predicting the Remaining Useful Life of a Li-ion Battery	Machine learning algorithms	NASA PCoE, CALCE datasets	Data-driven RUL prediction	RMSE, MSE, R2
5	Severson et al.	Data-driven prediction of battery cycle life before capacity degradation	Statistical learning methods	Cycling data from commercial batteries	Early prediction of cycle life	R2, RMSE, MAE
6	Li et al.	Remaining Useful Life Prediction for Lithium-Ion Batteries Using Gaussian Process Regression	Gaussian Process Regression (GPR)	NASA PCoE datasets	RUL prediction with uncertainty	RMSE, MAE, Coverage probability
7	Xiong et al.	A systematic review on machine learning in RUL prediction of Li-ion batteries	Literature review	Various	Systematic review of ML methods	N/A
8	Dong et al.	Ensemble Learning for Li-ion Battery RUL Prediction	Ensemble learning techniques	CALCE, NASA PCoE datasets	Improved accuracy in RUL prediction	RMSE, MAE, R2

9	Wu et al.	Model-based and Data-driven Prognostics for Lithium-ion Batteries	Hybrid model-based and data-driven approaches	Laboratory and field data	Comprehensive prognostics methods	RMSE, MSE, MAE
10	Zhang et al.	Long Short-Term Memory Networks for Battery Remaining Useful Life Prediction	Long Short-Term Memory (LSTM) networks	NASA PCoE datasets	Deep learning for RUL prediction	RMSE, MAE, R2
11	Feng et al.	Online State-of-Health Estimation and Remaining Useful Life Prediction for Li-ion Batteries	Online estimation using Kalman filters	Real-time data from EV applications	SOH estimation, RUL prediction	RMSE, MAE
12	Liu et al.	A novel remaining useful life prediction method for lithium-ion batteries based on a weighted relevance vector machine	Relevance Vector Machine (RVM)	Laboratory cycling data	Novel machine learning approach	RMSE, MAE, R2

13	Sun et al.	Battery Remaining Useful Life Prediction Based on a Hybrid Model Combining Physical and Data-driven Methods	Hybrid physical-data-driven models	Laboratory data, NASA PCoE datasets	Hybrid modeling for enhanced accuracy	RMSE, MAE
14	Zheng et al.	Prognostics of Lithium-Ion Batteries Based on Statistical Feature Extraction	Statistical feature extraction and regression models	NASA PCoE datasets	Feature extraction for prognostics	RMSE, MSE, MAE
15	Li et al.	Remaining Useful Life Prediction of Lithium-ion Batteries Based on Particle Filter	Particle Filter (PF)	Simulated and experimental data	RUL prediction with filtering methods	RMSE, MAE, Coverage probability
16	Xie et al.	Bayesian Neural Network-Based Remaining Useful Life Prediction for Lithium-Ion Batteries	Bayesian Neural Network (BNN)	CALCE, NASA PCoE datasets	Bayesian approach for uncertainty	RMSE, MAE
17	Chen et al.	Remaining Useful Life Prediction of Lithium-ion Batteries	Multiscale Convolutional Neural	NASA PCoE datasets	Multi-scale deep learning model	RMSE, MAE, R2

		Using Multiscale Convolutional Neural Network	Network (MCNN)			
18	Yang et al.	Remaining Useful Life Prediction of Lithium-ion Batteries Based on Support Vector Regression	Support Vector Regression (SVR)	Laboratory cycling data	SVR for battery RUL prediction	RMSE, MAE, R2
19	Xu et al.	Remaining Useful Life Prediction of Lithium-ion Battery Based on Fusion of ARIMA Model and Elman Neural Network	ARIMA and Elman Neural Network	NASA PCoE datasets	Fusion model for improved prediction	RMSE, MAE
20	Hou et al.	Prognostics of Lithium-ion Batteries Using Echo State Network with Degradation Feature Extraction	Echo State Network (ESN)	Laboratory and field data	Novel RNN approach for prognostics	RMSE, MAE

Table 2.1: *Literature Survey*

In this chapter, the literature review regarding this research presented. A brief overview of various remaining useful life (RUL) prediction methodologies is presented. In section, we review some of the commonly used publicly available lithium-ion battery degradation data sets.

2.2 Types of RUL prediction data sets

The data used to predict the RUL of a system can be classified into three categories as illustrated in Figure 3.1 - lifetime data, run-to-failure data and threshold data.

In lifetime data, the probability distribution of the failure times of the system are acquired and it is used to predict the RUL. Figure 3.2 shows the probability of a battery reaching its end of life (EoL) given the number of cycles the battery is in operation (Baru, 2018).

Figure 2.1: Plot showing SOH vs Cycles for the LIBs in NASA data set

In run-to-failure data, the entire degradation history from the beginning of life (BoL) to end of life (EoL) of similar systems are acquired. The RUL of a new test system can be predicted based on the RUL values of the degradation histories of the systems that resemble closely. Figure 2.3 illustrates the RUL prediction of a test system (highlighted in red) given the entire degradation histories of similar systems (highlighted in light blue).

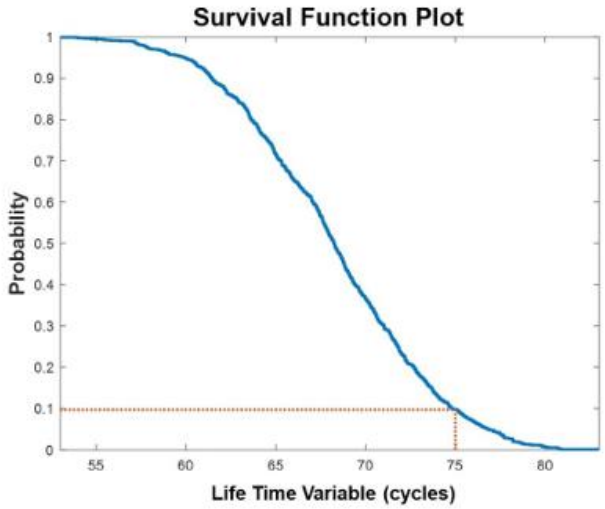


Figure 2.2: Illustration of RUL prediction using lifetime data.

2.3 A review of publicly available data sets

There are many lithium-ion battery (LIB) degradation data sets that are publicly available. A list of publicly available lithium-ion battery datasets are aggregated and reviewed in a review paper published by Reis et al. (2021). In this section, we review some of the most commonly used publicly available LIB degradation data sets.

2.2.1 NASA Data set

The lithium-ion battery degradation data set published by NASA Ames Prognostics Center of Excellence (PCoE) is one of the most commonly used data sets to develop machine learning models for State of Charge (SOC), State of Health (SOH) and Remaining Useful Life (RUL) prediction. This data set contains the run-to-failure data of 4 LIBs, which are labeled as *B0005*, *B0006*, *B0007* and *B0018*. These batteries were subjected to charging in a constant current (CC) mode at 1.5A. The charging was performed until the battery voltage reached 4.2 V. Then the batteries were charged in a constant voltage (CV) mode until the charge current dropped to 20 mA.

Frequent charge and discharge cycles of the batteries resulted in accelerated aging of the batteries. The experiments and measuring battery parameters such as voltage, current and temperature were stopped when the batteries reached their end of life (EOL), i.e., when the capacity of the batteries reached 70% of their initial capacity value (from 2Ahr to 1.4Ahr) (Saha and Goebel, 2007).

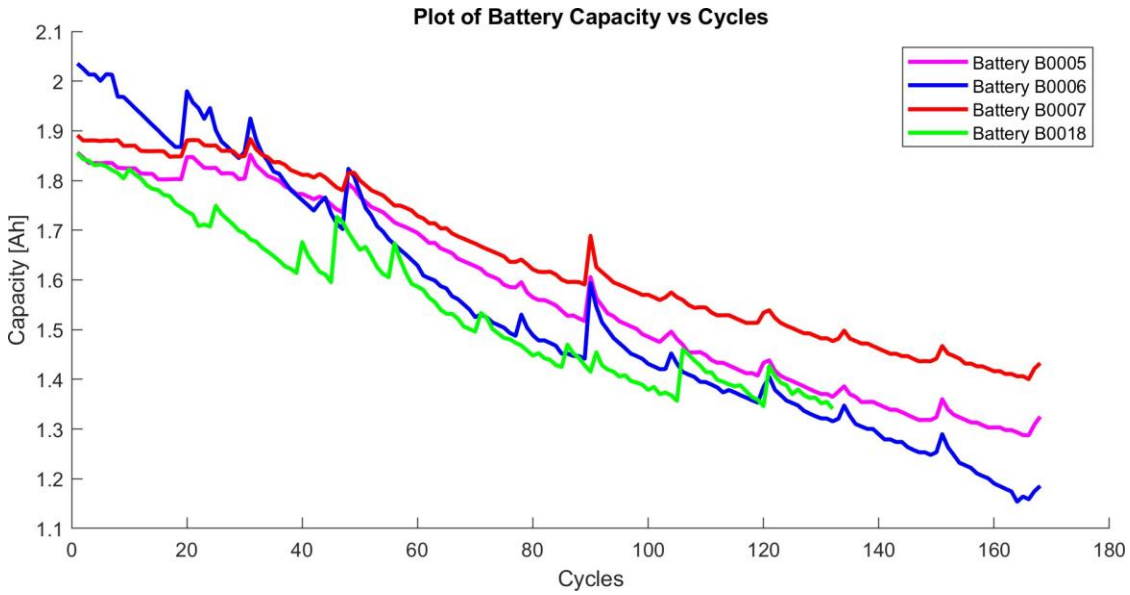


Figure 2.3: Plot showing SOH vs Cycles for the LIBs in NASA data set

2.2.1 CALCE Data set

The CALCE data set was published by the Center for Advanced Life Cycle Engineering (CALCE) at University of Maryland (Pecht, n.d.) and it contains data of several sets of batteries that were subjected to degradation by repeated charging and discharging.

The data for the first set of batteries were acquired by testing 15 Lithium Cobalt (LCO) prismatic cells. The cells are grouped into 6 groups based on the experimental conditions. The cells are at cycled at a room temperature of 23 ˚C. The cells are subjected to diverse range of partial charging and discharging. The data for this set of cells contains measurements of voltage, current, battery internal resistance, impedance, discharge capacity and charge capacity. The detailed measurement data for each battery cycle is stored in a separate file. The measurements from the batteries are acquired until the batteries end of life, i.e., until they reach 80% SOH.
The second set contains measurements of 12 LCO prismatic cells. The cells have a rated capacity of 1.35Ah. Similar to the first of cells used in this data set, the second set of cells are also grouped into 6 groups. The various parameters measured are the same as that of the first set of cells.
The third set contains measurements of 16 LCO pouch cells. The cells have a rated capacity of
1.5Ah. The cells are cycled at controlled temperature range of 23 ˚C to 27˚C. The data for this set of cells contains the measurements of voltage, current and battery capacity.

2.2.2 Oxford Data set

The Oxford Battery Degradation data set contains measurements of battery aging data from 8 small lithium-ion pouch cells (Birkl, 2017). All the cells used for testing were tested at 40^{o}C. The required temperature during testing was maintained with the help of a thermal chamber. The cells under test were exposed to a constant current constant voltage (CCCV) charging profile, followed by a drive cycle discharging profile. The discharging profile was obtained from the urban Artemis profile. The measurements were taken every 100 cycles (Birkl, 2017).

Figure 3.6 illustrates the variation of battery temperature w.r.t time for various battery cycles for a battery in the Oxford data set. From this figure, it can be observed that the battery temperature increases when the battery is subjected to repeated charge discharge cycles. This increase in temperature is an indicator of battery aging (Nuhic et al., 2013). Figure 3.7 shows the graph of voltage vs time for different battery cycles for a battery in the Oxford data set.

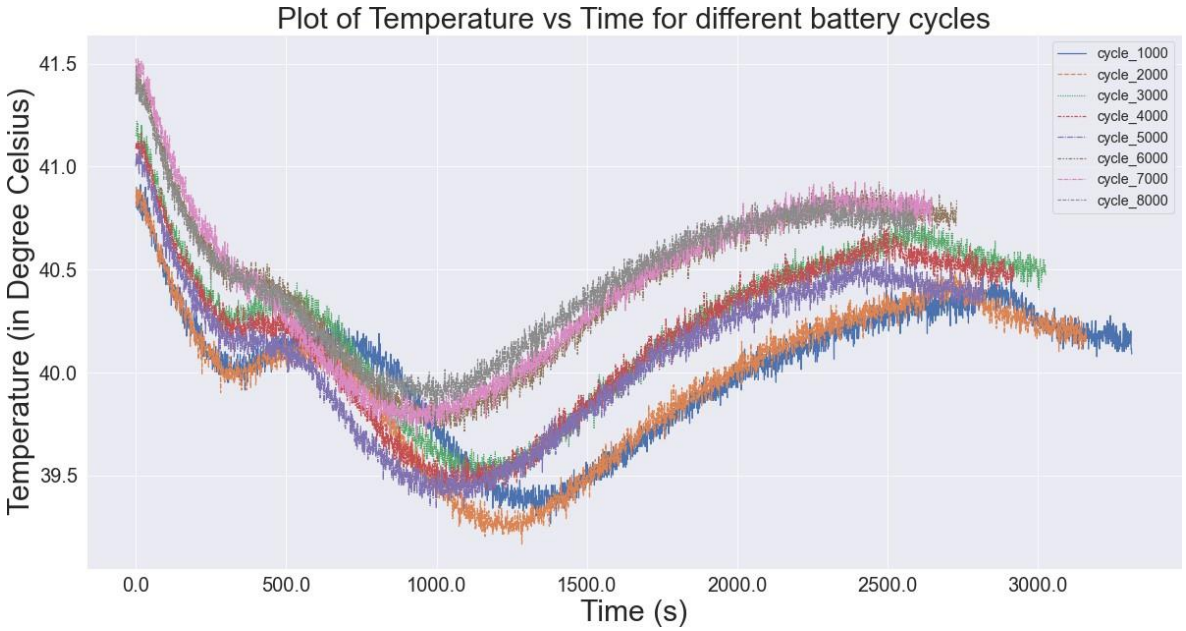


Figure 2.4: Plot of Battery Temperature vs Time of a battery in Oxford data set.

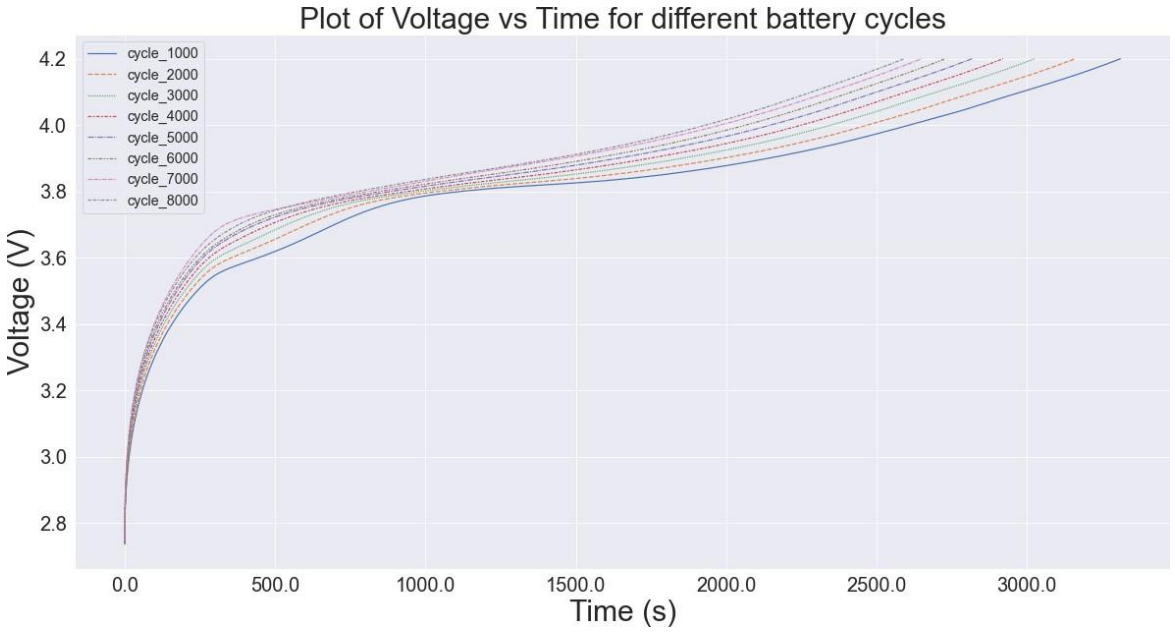


Figure 2.4: Plot of Voltage vs Time of a battery in Oxford data set.

2.2.4 Toyota Research Institute Data set

This lithium-ion battery degradation data set is published by Toyota Research Institute and it contains the run to failure (RTF) degradation data of 124 commercial lithium-ion batteries where each battery contains several lithium-ion phosphate (LFP) / graphite cells. The specifications of the cell used is shown in table B.2. The batteries were cycled at a constant temperature of 30° C and the temperature was controlled by a temperature chamber.

The main objective behind this data acquisition was to optimize the fast charging of the lithium- ion batteries. All the cells used to acquire the data are either charged with a single-step or two-step fast charging policy. The format of the charging policy is" C1(Q1)-C2", where C1 and C2 are the first and second constant current (CC) charging steps and Q1 is the state of charge (SOC) in percentage at which the currents switch. C2, which is the second current step in the fast-charging policy, ends at 80% SOC followed by constant current constant voltage (CCCV) charging at the C-rate of 1C. All the cells discharge at a C-rate of 4C (Severson, 2019).

Data generated due to the cycling of the batteries were logged from cycle 2 until the end of life (EOL) of the batteries. A SOH level of 80% is considered to be the EOL of the battery. The acquired data contains in-cycle measurements of current, voltage, charge capacity, discharge capacity and temperature. In addition to the in-cycle measurements, the acquired data contains the per cycle measurements of internal resistance, capacity and charge time. A plot of discharge capacity vs cycles for the data set is shown in Figure 2.5

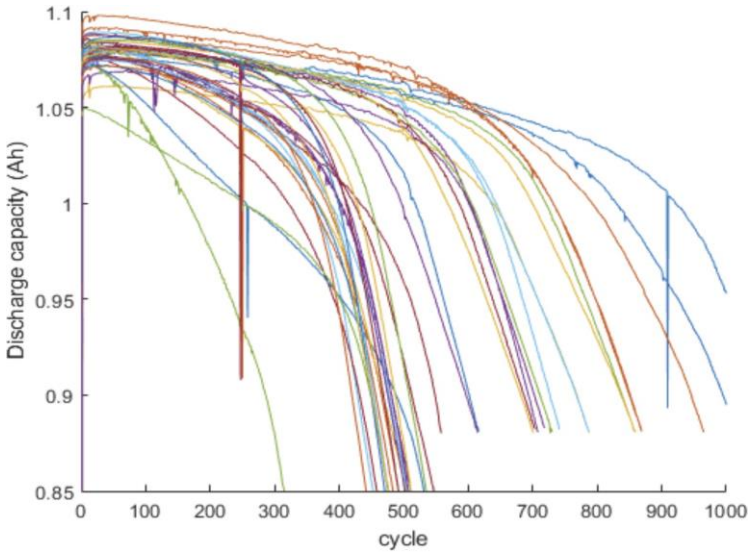


Figure 2.5: Plot of discharge capacity vs cycles of the lithium-ion battery degradation in the Toyota Research Institute (TRI) data set.

2.3 A review on RUL prediction methodologies

The RUL prediction methodology can be broadly classified into four methodologies: *physics-based methodology, experimental-based methodology, data-driven methodology and hybrid methodology*.

(Ahmadzadeh and Lundberg, 2014). This is illustrated in Figure 2.9.

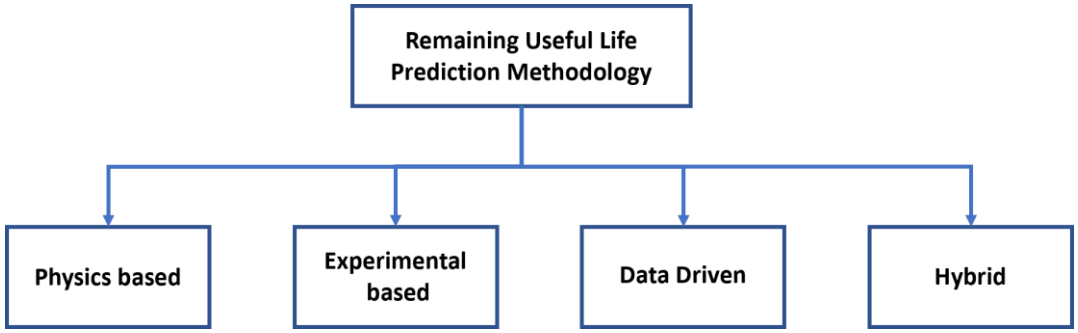


Figure 2.6: An overview of different RUL methodologies

Physics-based methodology for Remaining Useful Life (RUL) prediction of a system can be used, given that there is an accurate theoretical model of the system. With a better understanding of the system degradation, these models can be updated accordingly, which will, in turn, improve the overall accuracy of the RUL prediction. This approach for RUL prediction is typically used to model the crack propagation in gears and wear corrosion.

Experimental-based methodology for RUL prediction uses probabilistic models and/or stochastic models which model the life cycle or the degradation of the system under consideration. One of the key features of this methodology is that it also considers the data and knowledge regarding the system gathered by experience. This approach is used to predict the RUL life of systems over a wide range of domains such as chemical industries, energy industries, manufacturing industries, etc.

Data-driven methodology relies heavily on the data to predict the RUL of the system. This approach does not require product knowledge to predict the RUL. The performance and accuracy of the RUL prediction of a system using this approach depends on the quality and quantity of the available data. This approach leverages data analysis and machine learning techniques to generate predictions for the RUL of the system. Some of the machine learning techniques used in this methodology are Support Vector

Machines (SVM), Gaussian Process Regressor (GPR), Artificial Neural Network (ANN), Auto-Regressive Integrated Moving Average (ARIMA) and Decision Tree Regressor. This methodology can be used to predict the RUL of a wide range of systems such as batteries, wind turbines. A detailed overview of several existing RUL prediction models based on the data-driven methodology is presented in Section 2.4

In hybrid methodology, the RUL prediction systems generally make use of more than one RUL prediction methodology for better performance. However, they are computationally expensive. A detailed overview of several existing RUL prediction models based on the hybrid methodology is presented in Section 2.5.

2.4 RUL prediction using data-driven methodology

2.4.1 Non-Probabilistic Models

Non-Probabilistic models are a type of machine learning model and they only return a point estimate when used for regression. Some of the non-probabilistic models Remaining Useful Life (RUL) prediction are ARIMA model, Artificial Neural Network (ANN) models and Support Vector Machine (SVM) models.

2.4.1.1 ARIMA Model

Auto-regressive (AR) based modeling is a non-probabilistic time series model. This model uses a linear combination of observations from previous time steps as input to predict the future time step values. The main advantages of the AR model is its low computational complexity and these models can be expressed using a small number of model parameters.

The AR model is linear while the battery capacity fading process is generally nonlinear. This can lead to under-fitting, especially for the long-term prediction. To solve this problem, an Auto- Regressive Integrated Moving Average (ARIMA) framework was proposed that combines the AR model and the Moving Average method. Instead of using past values of the forecast variable in a regression, the moving average uses the past forecast errors in a regression-like model.

A group of researchers proposed an RUL prediction for LIBs using ARIMA (Y. Zhou and Huang, 2016). The researchers applied Empirical mode decomposition (EMD) on the SOH time series to decouple the global trend and the capacity regeneration values. The forecasts were predicted on each of the decoupled series and the forecasts are then combined to obtain the final result. Commonly used.

2.4.1.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is an ML model in which a group of computational units called neurons are stacked atop one another to form a layer. A neuron is the basic building block of a neural network. It typically accepts one or more inputs and produces one or more outputs. Many such layers are placed one after the other and all the layers are densely connected., there is a many to many connections between each neuron in a pair of layers.

In research conducted by (Jorge et al., 2020), an RUL prediction system using Artificial Neural Network (ANN) was developed. The system was tested on the battery aging data made publicly available by the department of chemical engineering of the Massachusetts Institute of Technology (Severson, 2019).

In another research conducted by (D. Zhou et al., 2020), the authors implemented an RUL prediction system using Temporal Convolution Network (TCN) (Lea et al., 2016). They observed that their RUL system performed better than the existing systems.

(Khumprom and Yodo, 2019) and team implemented an RUL prediction system for Lithium io batteries using Deep Neural Networks. (Adib, Angela and Lim, 2020) proposed an RUL prediction system for lithium-ion batteries that uses a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to forecast the RUL of the batteries. (T. Sun et al., 2019) proposed a similar model that uses Particle Filter (PF) and Multi-Layer Perceptron (MLP), resulting in the PFMLP model. The researchers then compared the performance of PFELM model, the standard Extreme Learning Machine (ELM) model and the model using the Bat Particle Filter with Neural Network (BATPFNN). The researchers concluded that the proposed BATPFNN model outperforms the standard ELM model and the PFMLP model.

2.4.1.3 Support Vector Machines

Support Vector Machine (SVM) is a non-parametric supervised learning technique. When used in a classification problem, it performs classification by searching for the hyperplane separating classes of interest with a maximal margin. When dealing with non-linear features, kernel functions are often used in SVM. Kernel functions transform the non-linear features in a low dimensional space into a linear feature in a higher dimensional space. (X. Li et al., 2017) and team developed an RUL prediction system that uses the SVM model to forecast the RUL.

2.4.1.4 Similarity based models

In the domain of prognostics and health management (PHM), Similarity based models (SBM) are one the commonly used data-driven models to predict the remaining useful life of a system under consideration. SBM is widely used to predict the remaining useful life of a test system given that the run-to-failure (RTF) data of similar systems are available. The data from similar systems are utilized to create a database of degradation profiles. During the remaining useful life prediction of a system under test, the degradation profile of the system under test will be matched against the degradation profiles of the systems available in the database. The actual life of the most similar systems that match with the system under test will be used in the prediction of the RUL of the system under test. In this section, we review some of the available literature regarding the use of similarity-based models (SBM) in remaining useful life prediction of lithium-ion batteries.

(S. Kim, N. H. Kim and Choi, 2020) used dynamic time warping (DTW) to augment the existing run-to-failure (RTF) data. The augmented data is then used to train a neural network model to predict the RUL of the system under test. When run to failure (RTF) data of similar systems with different degradation patterns under different failure modes are available, the results show that the proposed model predicts RUL with better certainty than the neural network models that predict RUL without data augmentation.

A group of researchers used similarity-based data-driven prognostic methodology for predicting of the fatigue life or remaining life of the structures (Eker, Camci and Jennions, 2014). The method used was originally developed by (Zio and Maio, 2010). The model was tested on three data sets - Virkler fatigue crack growth data set (Virkler, Hillberry and Goel, 1979), a drilling process degradation data set and a sliding chair degradation of a turnout system data set. They evaluated the RMSE and found out that the model produced lesser RMSE for two out of three models considered for research.

(P´erez et al., 2018) proposed a similarity-based approach for predicting the RUL of lithium-ion batteries. The research was conducted on the data obtained after cycling Sony US18650 1.4 Ah LIBs using different discharge rates. They also used Monte Carlo simulations to analyze the lifespan of batteries which helps to show the usage of the batteries in a more realistic way. The researchers claim that the suggested SBM technique can be extended to other types of LIBs as the results are normalized and scaling factors are used.

In another conducted by (Soons et al., 2020), the researchers used a similarity-based model to predict the RUL of a filter in a chemical industry. The data for the research

is gathered by Chemelot Industrial Site. The researchers claim that their model works well even if the operational conditions are unknown.

2.4.2 Probabilistic Models

Probabilistic models represent a set of machine learning models and they return a point estimate as well as confidence interval when used in regression. Some of the probabilistic models that are commonly used for RUL prediction are Gaussian Process Regression (GPR) model, Relevance Vector Machine (RVM) model.

2.4.2.1 Gaussian Process Regression

Gaussian Process Regressor (GPR) is a kernel-based machine learning technique, which can realize prognostics combined with prior knowledge based on a Bayesian framework and provide variance around its mean prediction to describe the associated uncertainty. The Gaussian process can be seen as a collection of a limited number of random variables which have a joint multivariate Gaussian distribution.
In research conducted by (Jia et al., 2020) and his colleagues, developed an RUL prediction system using Gaussian Process Regressor (GPR). The dataset used for this research is sourced by NASA Ames Prognostics Center of Excellence (PCoE) database (Saha and Goebel, 2007). (Liu and Z. Chen, 2019) also developed an RUL prediction system that uses Gaussian Process Regressor (GPR) to predict the RUL of LIBs.

2.4.2.2 Relevance Vector Machines

Relevance Vector Machine (RVM) is identical to Support Vector Machine (SVM). When used in a regression problem, a relevance vector machine returns a point estimate of the prediction as well as a confidence interval. The RVM employs a Bayesian framework to infer the weights with which the probability distribution functions (PDFs) of the outputs instead of point estimates can be obtained.
(Qin et al., 2017) conducted research where they implemented an RUL prediction system using Relevance Vector Machine (RVM). The authors used the Battery dataset published by NASA (Saha and Goebel, 2007).

2.5 RUL prediction using hybrid methodology

(Wei, Dong and Zonghai Chen, 2018) proposed an RUL prediction model for lithium-ion batteries (LIB) which uses Support Vector Regressor (SVR) and Particle Filter (PF). (Wu et al., 2019) developed an RUL prediction model for lithium-ion batteries and it uses Neural Network and Bat Based Particle Filter. The research shows that the accuracy obtained using the combined model exceeds the accuracy obtained using either only Neural Network or only Particle Filter (Moral, 1997).

(L. Li et al., 2018) developed an RUL prediction model for lithium-ion batteries and it uses Relevance Vector Machine (RVM) and Particle Filter (PF) and Auto Regression (AR) model. The implemented algorithm was tested on a private data set as well as on a public data set published by NASA (Saha and Goebel, 2007). The researchers conclude that the combined Hybrid model yields effective results.

(Y. Zhang et al., 2019) proposed an RUL prediction model which combines Relevance Vector Machine (RVM) and Particle Filer (PF). This procedure will significantly reduce the size of the training data required to train the model. Finally, the researchers use Monte Carlo method to calibrate the number of particles and to model the noise level of the Particle Filter.

(S. Wang et al., 2021) researched by analyzing, reviewing, classifying and comparing various adaptive mathematical models especially on deep learning algorithms for the task of remaining useful life prediction. Extreme Machine Learning (EML) models are used for the cycle life prediction of lithium-ion batteries. EML models possess a single hidden layer as compared to the feed-forward neural network.

Convolutional Neural Network (DCNN), Deep Neural Network (DNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), DCNN - Ensemble Transfer Learning (DCNN-ETL), DCNN with Ensemble Learning (DCNN-EL), DCNN with Transfer Learning (DCNN-TL), Adaptive LSTM

(ALSTM), ALSTM with an attention mechanism, Average Euclidean distance Stacked Denoising Autoencoder (AED-SDA) and Gated Recurrent Unit kernel - Gaussian Process Regression (GRU-GPR) models were compared. Evaluation metrics such as Root Mean Squared Error, Max E, Speed and accuracy for all the above models are compared. They make a careful analysis of different DL modeling methods for accurate RUL prediction. Their experiments show that the DCNN-ELM algorithm gave better results than other models.

(D. Wang et al., 2017) proposed a model with non-linear drifted Brownian motion with many hidden states. The proposed method is more efficient than the models that used particle filtering. Private data of 26 rechargeable batteries were analyzed. The researchers performed the comparisons of their proposed model with the standard PF (particle filter) based prognostic method, the spherical cubature PF based prognostic method and the classic Bayesian prognostic method. They finally concluded that the proposed model in their research, which is, nonlinear-

drifted Brownian motion with the multiple hidden states has lower average prediction errors than the other models.

(H. Wang, Ma and Zhao, 2019) proposed a model where an improvised version of Wiener process model with adaptive drift and diffusion was developed for RUL prediction. They used an algorithm to eliminate the abnormal monitoring data based on the 3-sigma criterion. They built a complete framework for online RUL prediction.

(Shen et al., 2021) worked on predicting the RUL of lithium-ion batteries considering the random variable discharge current. They analyzed the impact of the variable discharge current on Li-ion batteries. They designed a two stage Wiener model to explain the lithium-ion battery degradation. A widely known open-source battery data set by Oxford has been used by the team for their research. The proposed method's performance metrics such as RMSE, MAPE, MAE, R-squared and Absolute Error are calculated and compared with the work conducted by (H. Wang, Ma and Zhao, 2019) and (D. Wang et al., 2017). The metrics were calculated for the three models taken for comparison and they were calculated for different failure thresholds of different batteries in the data set. It was concluded that the proposed model performed better than the models proposed by (H. Wang, Ma and Zhao, 2019) and (D. Wang et al., 2017).

(X. Xu et al., 2021) proposed an RUL prediction model considering the time varying temperature condition. The proposed model was developed using the Bayesian framework. The researchers observed that time-varying temperature condition has a major impact on the discharge capacity and aging of a lithium-ion battery. They started by proposing a stochastic degradation rate model based on the Arrhenius temperature model. Then, using Wiener process, an aging model of lithium-ion battery was developed. Then, an unbiased estimation method based on maximum like-lihood estimation (MLE) combined with genetic algorithm (GA) is proposed. Under the Bayesian framework, the random parameter is updated. Probability Density Function (PDF) of the RUL for lithium-ion battery under given time-varying temperature condition is derived. Finally, the performance of the model thus built is evaluated and it was concluded that the model built had higher accuracy and lesser uncertainty. The data set used to check the performance of the model is CALCE (Pecht, n.d.).

(Hong et al., 2020) and team used deep learning model to predict the RUL of lithium-ion batteries in a much faster way. They claim that their model is the first end-to-end deep learning framework for RUL prediction. Their framework swiftly predicts the battery RUL with only four cycles. It is 25 times faster than the previous models. They analyzed the temporal patterns of terminal voltage, temperature of the cell and current. The framework proposed is interpretable. They used dilated CNN for the prediction of RUL. They have used dilated CNN instead of using RNN and its variants such as LSTM, GRUs because of their limited ability to capture considerably long-term relationships. Whereas dilated CNNs have proven examples where they have robustly captured long term relationships in a time series data. The researchers used the lithium-ion battery data set from MIT-Stanford. They built several DL based models such as shallow MLP, MLP, CNN,

CNN + LSTM and dilated CNN. Dilated CNN performs the best among other models and has given an error rate of 10.6

("Remaining useful life prediction of lithium-ion batteries based on Monte Carlo Dropout and gated recurrent unit" 2021) developed an RUL prediction model for lithium-ion batteries using Monte Carlo Dropout and GRU. To test the performance of this model, the battery data set provided by NASA PCoE Research Center is used (Saha and Goebel, 2007). The GRU is the model is used to overcome the gradient vanishing problem which is generally faced in Deep Neural Networks (DNN). The dropout is used in the proposed model to overcome the problem of over-fitting of the data. The performance of the proposed model is compared with two other models - Back Propogation Neural Network – Particle Swarm Optimization (BPNN-PSO) and Least Squares – Support Vector Machine (LS-SVM). RMSE, MAE and MAPE of all three models are calculated. It was found that the proposed model gave better results than BPNN-PSO and LS-SVM models.
(Zewang Chen et al., 2021) proposed a hybrid model which combines a broad learning system with a relevance vector machine (RVM) regressor. Empirical mode decomposition (EMD) is used to extract the features of the data. The training data is then sent into the BLS network, where multiple prediction starting points are chosen, and the associated prediction data is generated. To train the RVM, all prediction data is transformed into a matrix. The RVM is used as the pre- diction layer of the proposed model. The output generated by the RVM is the predicted RUL of the model. Three different lithium-ion battery data sets are used to check the performance of the model – NASA battery data set (Saha and Goebel, 2007), CALCE (Pecht, n.d.) and independent data set. The results suggest that this hybrid technique can forecast lithium-ion batteries with great precision. It has less errors, higher long-term prediction and generalization capabilities than a single model. RMSE, MAE and error in RUL are evaluated. The proposed hybrid model gave better results.

2.6 Conclusions

In this chapter, we reviewed some of the most commonly used publicly available lithium-ion battery (LIB) degradation data sets. We also reviewed four RUL prediction methodologies - *physics-based, experience-based, data-driven and hybrid methodologies*.

Physics-based methodology for RUL prediction of LIB gives accurate predictions provided there is an accurate model of an LIB capacity degradation. However, in real-time scenarios, the degradation of LIB in an electric vehicle is affected by various factors and various drive cycle patterns and a theoretical model cannot does not take into account all the important factors that affect battery degradation. Also, physics-based models rely

heavily on the model assumptions. Hence, we do not prefer physics-based methodology to predict RUL of LIB.

Data-driven models rely only on the observed system data and it does not make any assumptions about the underlying system. The effects of various factors that affect the RUL of the system are captured in the data and data-driven machine learning models can be developed accordingly to predict RUL with high accuracy. Based on the literature review, we implement SVR model, LSTM network and similarity-based model to predict the remaining useful life.

2.7 Research Gap

Out of the many research articles published on RUL prediction of lithium-ion batteries, very rarely the articles address about implementing the RUL prediction algorithm in real-time in an EV. Ideally, an RUL prediction model should start predicting the RUL right after the first charge- discharge cycle of the battery. Hence, we try to address this research gap as part of this research.

Although, many researchers have used similarity-based models to predict the RUL of a variety of systems, not many research has been published, where SBM is implemented to predict the RUL of LIBs using dynamic time warping (DTW) distance. In this research work, we also try to implement a SBM with DTW as the distance metric.

2.8 Objective

Design and develop SW system that predicts the Remaining useful life (RUL) of lithium-ion battery and ensure battery performance, reliability and safety. Enable predictive maintenance of Lithium-Ion batteries. By accurately predicting the remaining useful life of a battery, it is possible to schedule maintenance activities in advance.

By accurately predicting the RUL, it is possible to avoid unnecessary replacements or repairs. RUL estimation can also help to improve the performance of Lithium-Ion batteries.

Chapter 3

Design Analysis

3.1 Model Architecture

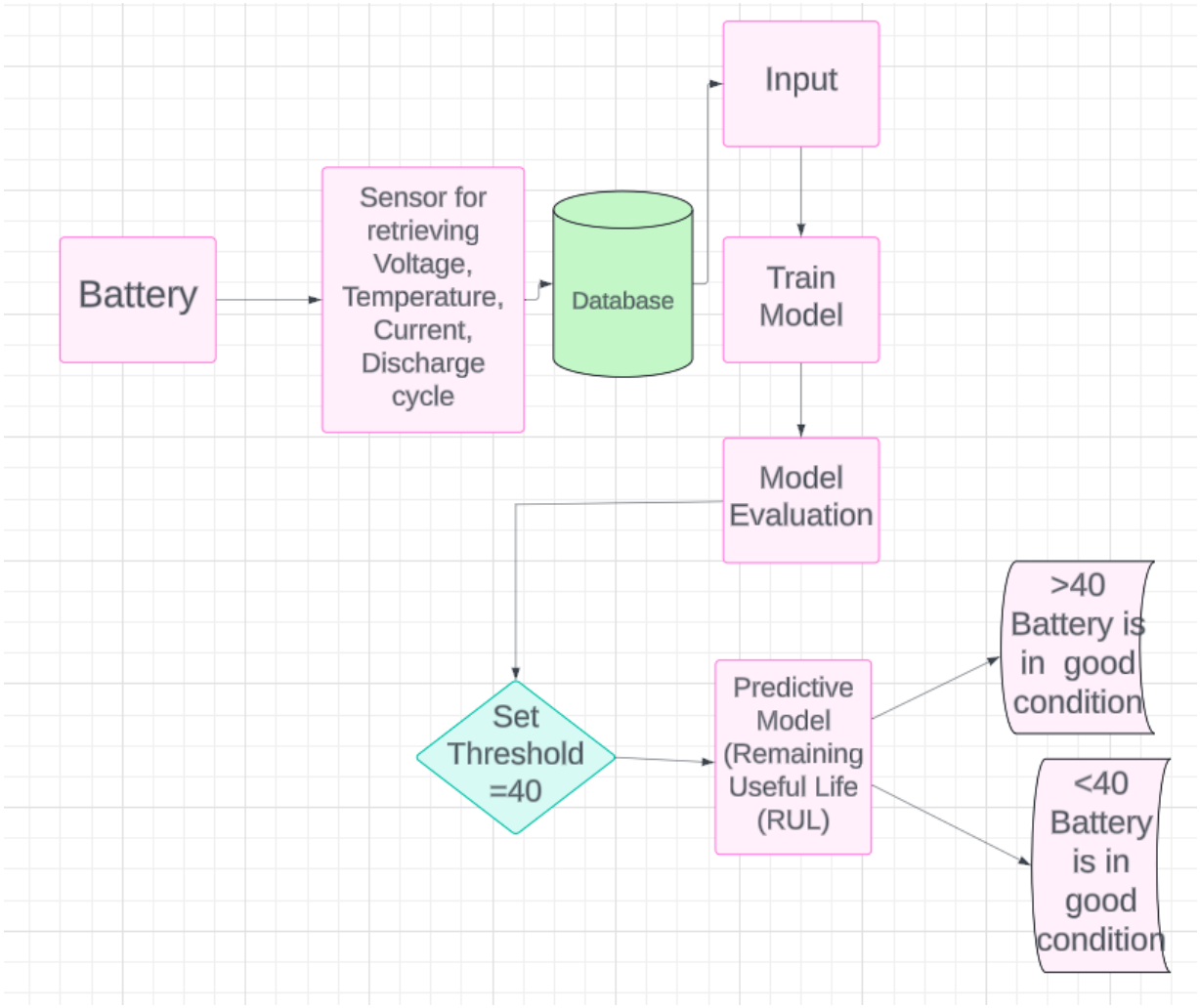


Fig 3.1: Model Architecture

3.2 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development.

3.2.1 Use-case Diagram

The **Use Case Diagram** outlines the interactions between users (actors) and the system for predicting the Remaining Useful Life (RUL) of Li-ion batteries. It involves actors like the User and Maintenance Engineer. The User inputs battery data and initiates the training of the neural network model. The system uses the Bat-PF algorithm to optimize parameters and predict RUL, which the Maintenance Engineer then views to schedule maintenance.

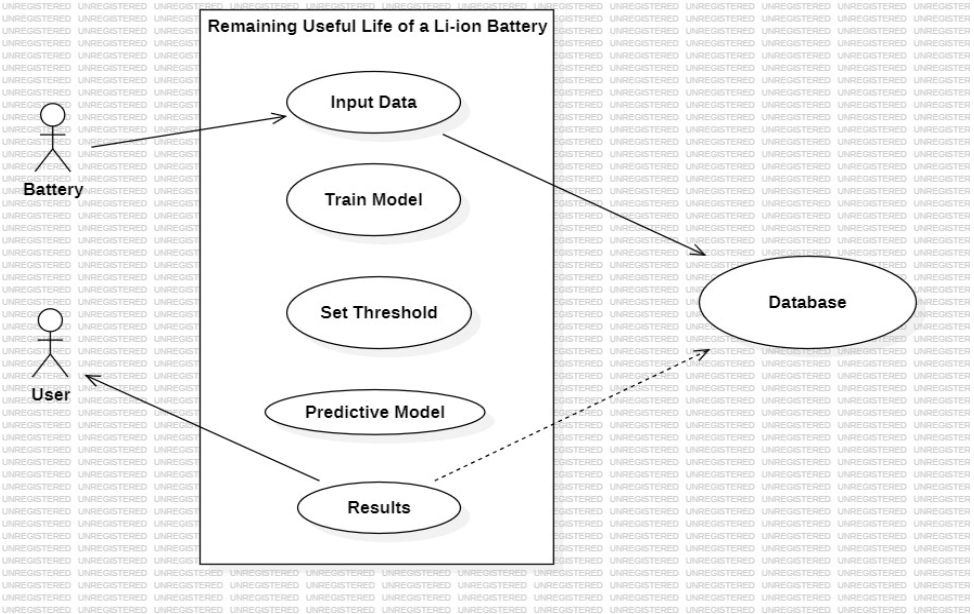


Fig:3.2 Use case Diagram

3.2.2 Sequence Diagram

The **Sequence Diagram** demonstrates the sequence of interactions between objects to achieve RUL prediction. The process starts with the User inputting battery data through the UserInterface, which is then stored in BatteryData. The UserInterface triggers the NeuralNetworkModel to train with historical data, which subsequently engages the BatPFAlgorithm for parameter optimization. The optimized parameters are used by RULPredictor to predict RUL, which is then displayed to the Maintenance Engineer by the UserInterface.

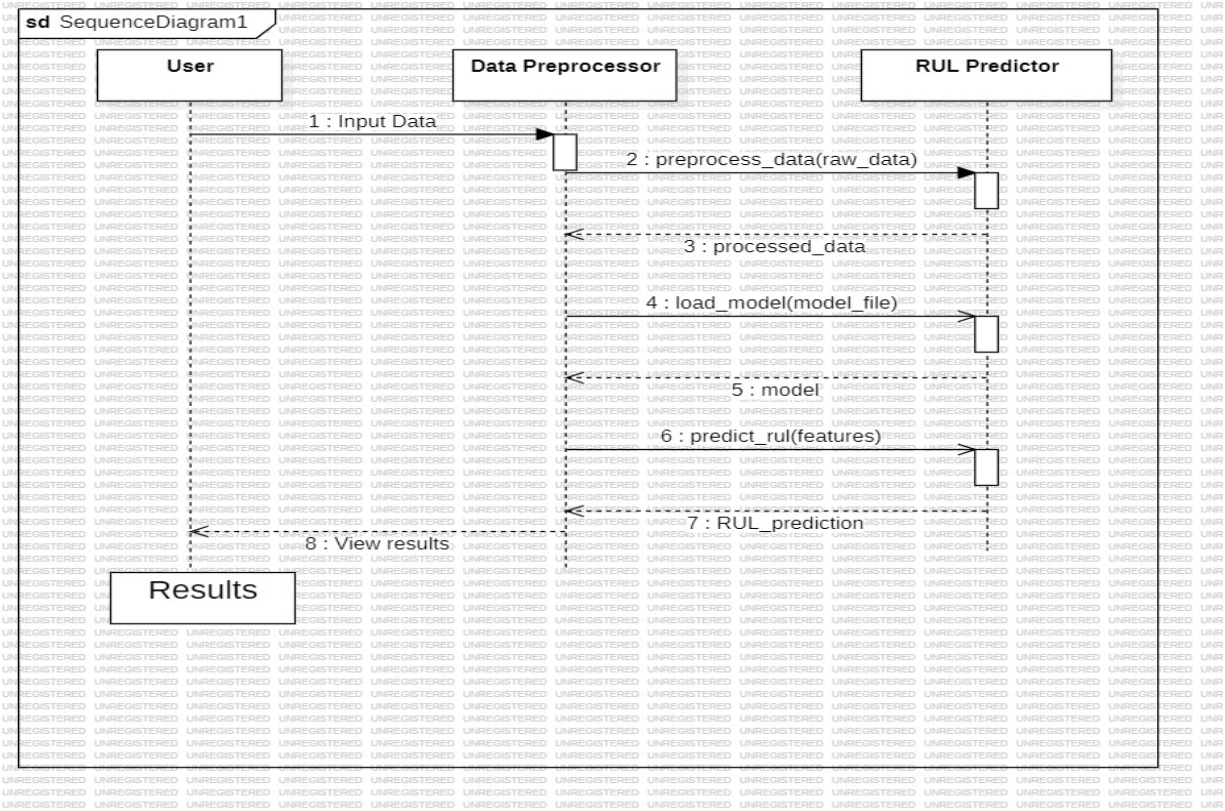


Fig: 3.3 Sequence Diagram

- **User** -> **UserInterface**: Input battery data.
- **UserInterface** -> **BatteryData**: Store the input data.
- **UserInterface** -> **NeuralNetworkModel**: Train the model using historical data.
- **NeuralNetworkModel** -> **BatPFAlgorithm**: Optimize the parameters using Bat-PF.
- **BatPFAlgorithm** -> **NeuralNetworkModel**: Return optimized parameters.
- **UserInterface** -> **RULPredictor**: Request RUL prediction.
- **RULPredictor** -> **NeuralNetworkModel**: Use the neural network to predict RUL.

3.2.3 Activity Diagram

The **Activity Diagram** maps out the workflow for predicting RUL. It begins with the User inputting battery data, followed by data preprocessing. The system then trains the neural network, optimizes parameters using the Bat-PF algorithm, and predicts RUL. Finally, the predictions are displayed to the Maintenance Engineer, facilitating maintenance scheduling.

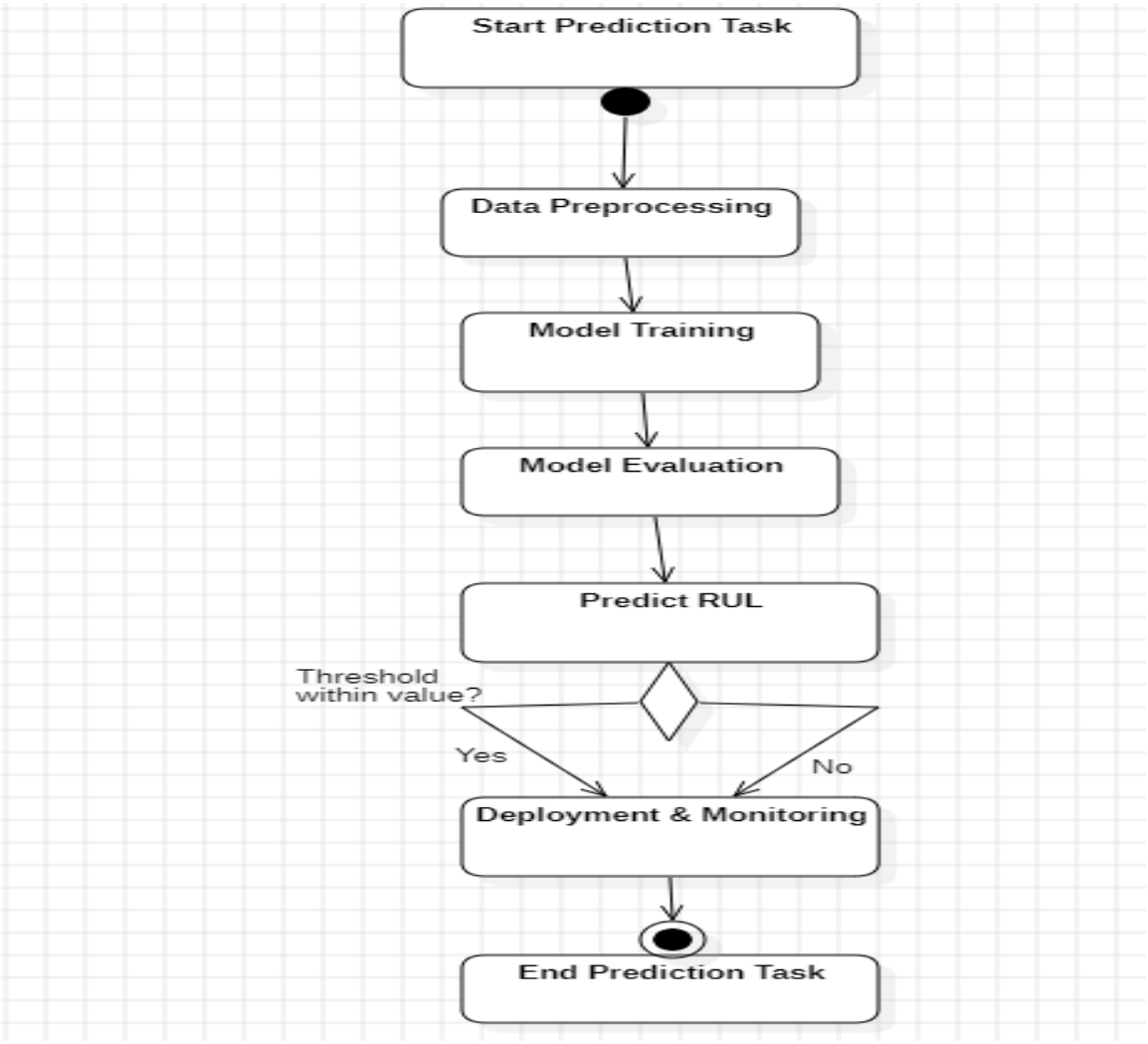


Fig: 3.4 Activity Diagram

Input Battery Data

- User inputs data via the user interface.

Preprocess Data

- System normalizes and prepares data for training.

Train Neural Network

- Neural network model is trained using historical data.

Optimize Parameters

- Bat-PF algorithm optimizes the Particle Filter parameters.

Predict RUL

- RUL is predicted using the trained neural network and optimized parameters.

Display Results

- RUL predictions are displayed to the maintenance engineer.

3.2.4 Component Diagram

A Component Diagram illustrates the structural relationships and interactions between software components in the system. For predicting the Remaining Useful Life (RUL) of Li-ion batteries, the components involved might include data management, model training, optimization, and user interaction.

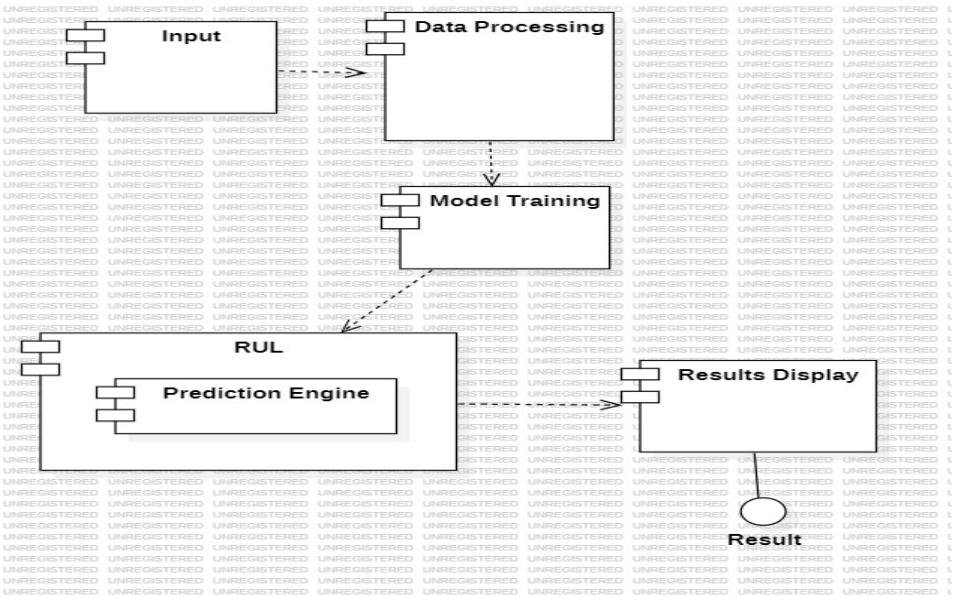


Fig: 3.5 Component Diagram

Components:

- **User Interface Component**: Manages interactions with the user.

- **Data Management Component**: Handles the storage, retrieval, and pre-processing of battery data.
- **Neural Network Component**: Manages the creation, training, and prediction processes of the neural network.
- **Optimization Component**: Implements the Bat-PF algorithm for optimizing the parameters.
- **Prediction Component**: Integrates the neural network and optimization results to predict the RUL.
- **Maintenance Scheduler Component**: Uses RUL predictions to schedule maintenance.

3.2.5 Data Flow Diagram (Level 0)

The Data Flow Diagram (DFD) for the prediction of the Remaining Useful Life (RUL) of Li-ion batteries illustrates the flow of data within the system and its interactions with external entities. At the highest level, the Level 0 DFD, the system is depicted as a single process interacting with two primary external entities: the User and the Maintenance Engineer. The User inputs battery data into the system, which stores this data and processes it to predict the RUL. The predicted RUL is then communicated to the Maintenance Engineer, who uses this information to schedule maintenance activities.

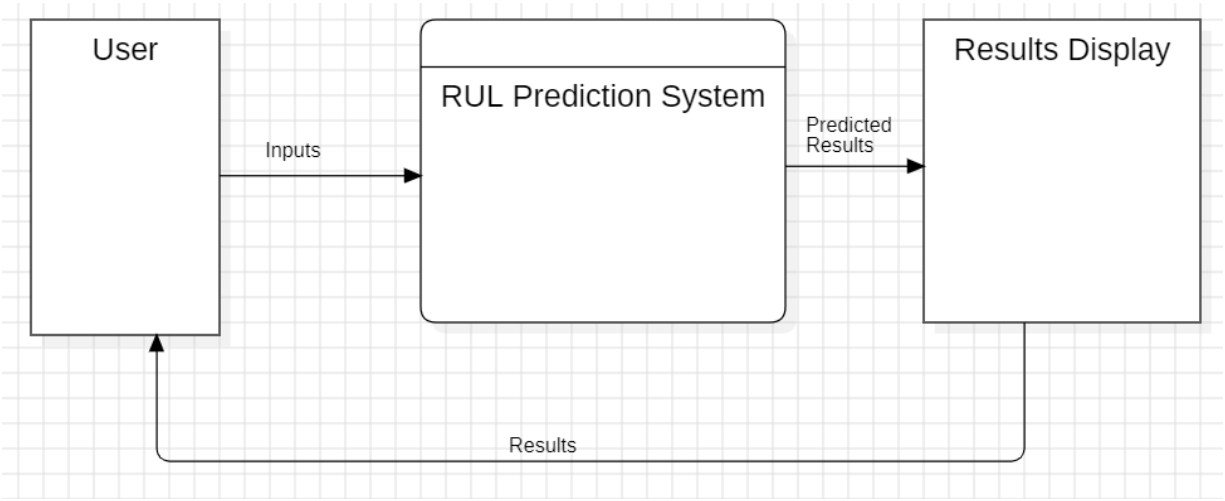


Fig: 3.6 Data Flow Diagram (Level 0)

Processes:

1. **Data Input**
2. **Data Pre-processing**

3. **Model Training**
4. **Parameter Optimization**
5. **RUL Prediction**
6. **Display Results**

3.2.6 Data Flow Diagram (Level 1)

Delving deeper, the Level 1 DFD decomposes the main process into more detailed sub-processes. These include Data Input, Data Preprocessing, Model Training, Parameter Optimization, RUL Prediction, and Display Results. The Data Input process captures and stores raw battery data in the Battery Data store. This data is then normalized and prepared for model training in the Data Preprocessing stage. The preprocessed data feeds into the Model Training process, where a neural network model is trained and validated. The trained model is optimized using the Bat-PF algorithm in the Parameter Optimization process, with the results stored in the Optimized Parameters store. The RUL Prediction process uses the trained model and optimized parameters to predict the battery's remaining useful life, storing these predictions in the RUL Predictions data store. Finally, the Display Results process generates a report of the predictions, which is communicated to the Maintenance Engineer.

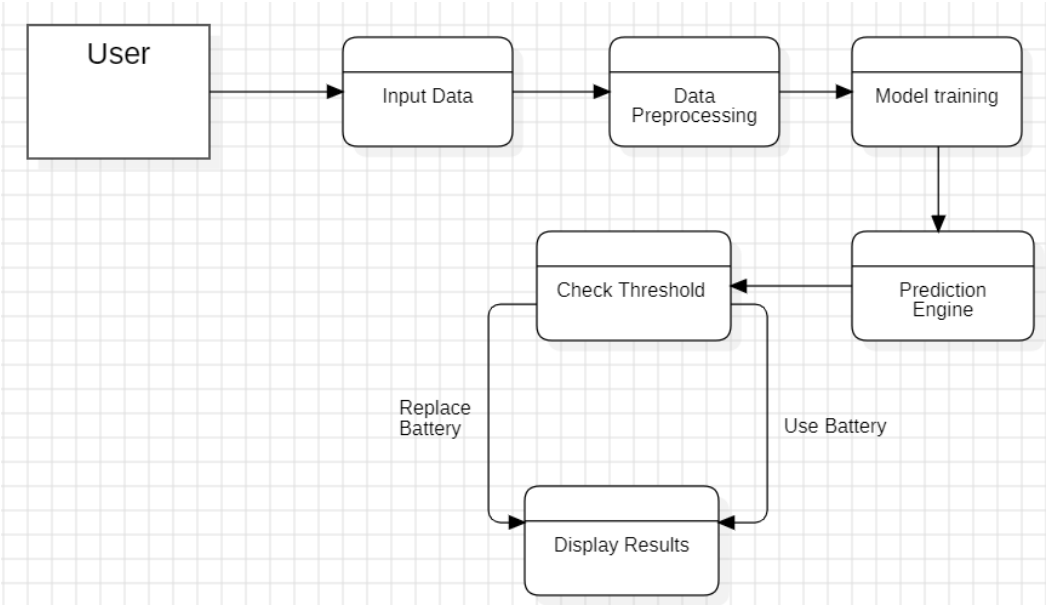


Fig: 3.7 Data Flow Diagram (Level 1)

Processes:

1. **Data Input**
2. **Data Pre-processing**

3. **Model Training**
4. **Parameter Optimization**
5. **RUL Prediction**
6. **Display Results**

3.2.7 Data Flow Diagram (Level 2)

The Level 2 DFD further breaks down each sub-process from Level 1 into more granular steps. For example, Data Input involves receiving user data and storing it, while Data Preprocessing includes normalizing data and splitting it for training and testing. Model Training is detailed into initializing the neural network, training it with data, and validating the model. Parameter Optimization is broken down into initializing particles, applying the Bat algorithm, and updating particles. RUL Prediction involves predicting using the neural network and integrating optimization results. Display Results consists of generating a prediction report and notifying the Maintenance Engineer.

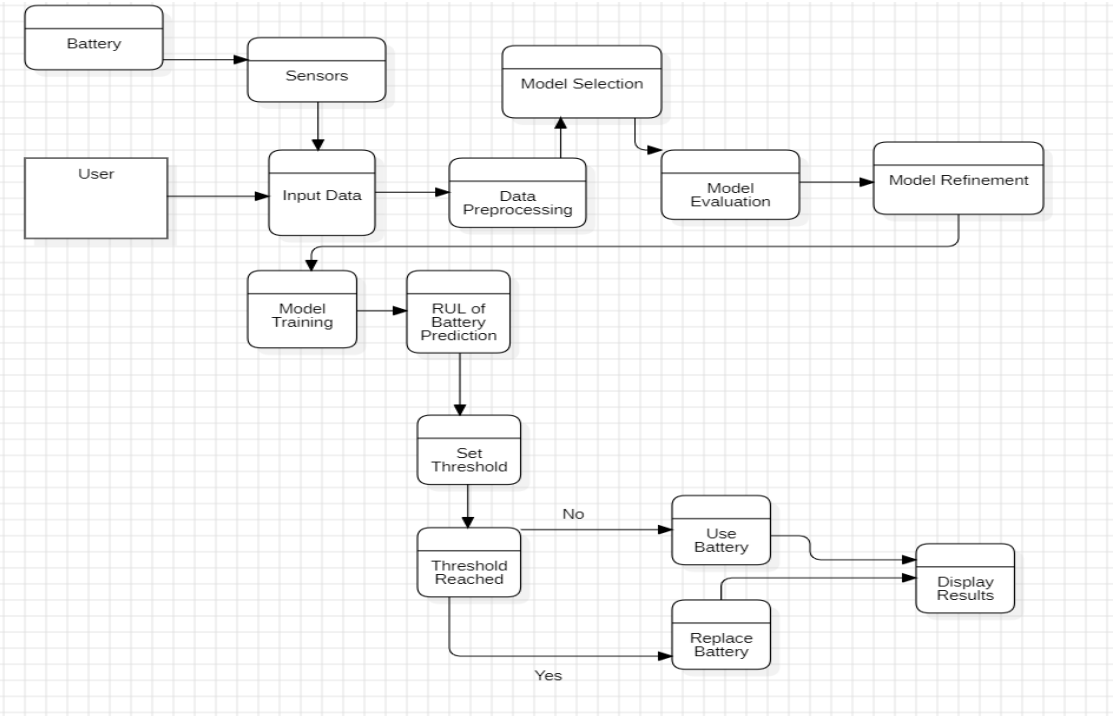


Fig: 3.8 Data Flow Diagram (Level 2)

Processes:

1. **Data Input**
    - Receive User Data

- Store Battery Data

2. **Data Pre-processing**
   - Normalize Data
   - Split Data for Training and Testing
3. **Model Training**
   - Initialize Neural Network
   - Train Network with Data
   - Validate Model
4. **Parameter Optimization**
   - Initialize Particles
   - Apply Bat Algorithm
   - Update Particles
5. **RUL Prediction**
   - Predict Using Neural Network
   - Integrate Optimization Results
6. **Display Results**
   - Generate Prediction Report
   - Notify Maintenance Engineer

Overall, the DFD provides a detailed visualization of how data flows through the system, from initial user input to the final output of RUL predictions, highlighting the various processes and data stores involved at each stage. This structured approach helps in understanding the system's functionality and identifying areas for optimization and improvement.

Chapter 4

Feature Engineering

In this chapter, we describe, explore, pre-process and create new features from the TNO data set. Then, In section 4.1, we briefly discuss the data description of the TNO data set. In section 4.2, we perform an exploratory data analysis (EDA) on the data set. Finally, In section 4.3, we pre-process and create new features from the TNO data set. These new features, along with the existing features will be used to build machine learning (ML) models to predict the remaining useful life (RUL).

4.1 TNO Data set: Data Description

This is a private data set provided by TNO as part of this research. This is a lab-generated data set and it was acquired by subjecting eight lithium-ion batteries, namely *A1*, *A2*, *A3*, *A4*, *A9*, *A10*, *A11*, *A12*, to diverse charging (load) cycles and discharging (drive) cycles. Generally, batteries are said to have reached their end of life (EOL), when their capacity value reaches 80% of the initial capacity (80% SOH). Except for battery *A9*, all the remaining batteries are cycled starting from their beginning of life (BOL) until they reach the end of life (EOL). The degradation history data for each of the batteries in the data set consists of features described in Section 4.1.1.

4.2 Data Description

The following are the features that were acquired for each of the cells in the TNO data set.

1. *Time* is the recorded end time expressed as a time entry.
2. *SOC init p* is the initial State of Charge (SOC) value based on the SOC vs Open Circuit Voltage (OCV) relationship. It is expressed as a percentage.
3. *SOC avg p* is the average State of Charge (SOC) is based on the mean SOC value. It is expressed as a percentage.
4. *SOC end p* is the end State of Charge (SOC) value is based on Coulomb Counting (CC). It is expressed as a percentage.
5. *DOD p* is the Depth of Discharge (DOD) and it is the difference between the initial SOC value and the end SOC value. It is expressed as a percentage.
6. *T cell min C* is the minimum cell temperature of the cycle. It is expressed as degree Celsius.
7. *T cell avg C* is the average cell temperature of the cycle. It is expressed as degree Celsius.
8. *T cell max C* is the maximum cell temperature of the cycle. It is expressed as degree Celsius.

9. *T-amb avg C* is the average ambient temperature of the cycle. It is expressed as degree Celsius.
10. *Crate avg* is the average current value of the cycle based on the initial capacity. It is expressed as per hour.
11. *Crate RMS* is the root mean square (RMS) current value of the cycle based on the initial capacity. It is expressed as per hour.
12. *Crate max* is the maximum current value of the cycle based on the initial capacity. It is expressed as per hour.
13. *Ctp CC As* is the constant current part of the charge throughput of the cycle based on coulomb counting. It is expressed as ampere second (As).
14. *Ctp As* is the charge throughput of the cycle based on Coulomb Counting (CC). It is ex- pressed as ampere second (As).
15. *Capacity As* is the last recorded capacity of the battery. It is expressed as ampere second (As).
16. *FileName* is the name of the file that contains data of the cycle measured at a high sampling frequency.
17. *Resistance Ohm* is the last recorded resistance at 50% SOC, where the cell temperature is between 22.5-25.5 $\circ C$. It is expressed as Ohm.
18. *TempCellResistance C* is the recorded cell temperature of *Resistance Ohm*. It is expressed as $\circ C$ (degree Celsius).
19. *Resistance mea Ohm* is the last recorded resistance at 50% SOC. Resistance measurement for LF Test might be extrapolated to get values at 50% SOC. It is expressed as Ohm.
20. *TempCellResistance mea C* is the recorded cell temperature for during resistance measurement. It is expressed as $\circ C$ (degree Celsius).
21. *dist km* is the estimated distance traveled based on the charge throughput and total distance covered. It is expressed as a kilometer (km).
22. *Total dist km* is the cumulative distance. It is expressed as a kilometer (km).
23. *Total Ctp As* is the cumulative charge throughput. It is expressed as ampere second (As).
24. *duration* is the duration of the cycle. It is expressed as a time entry.
25. *Total duration* is the cumulative duration. It is expressed as a time entry.

In addition to the degradation data of the 8 lithium-ion batteries (LIB), the TNO data set also contains data acquired at a very high sampling rate during charging and discharging. The acquired data for every charging instance and discharging instance for all the batteries are saved in separate
.mat files. A .mat file is a binary file used to save and load MATLAB® workspace variables. There are around 44004 .mat files. Each .mat file contains the following features.

1. *Current Cell* represents the name of the battery for which the data was acquired.
2. *Time* represents the time at which a particular sample was measured.

3. *Current* represents the value of the current measured. It is expressed as ampere (A).

Voltage represents the value of the voltage measured. It is expressed as volt (V)

4.2.2 Exploratory Data Analysis

In this section, we perform exploratory data analysis (EDA) on the TNO data set. As mentioned earlier, the data set contains the entire degradation history of 8 lithium-ion batteries (LIB), namely *A1, A2, A3, A4, A9, A10, A11 and A12*. These batteries are subjected to repeated charging, discharging and calendar aging until they reach the end of life (EOL). In most research regarding LIBs, a battery that has been degraded up to 80% SOH is considered to have reached it EOL. In this research, we consider that a battery has reached its EOL after it has degraded up to 81% SOH. The battery *A9* in the TNO data set is subjected to degradation up to 92% SOH and the battery has not been degraded up to its EOL. As seen from the Figure 4.1, the SOH of the battery *A9* (highlighted in bold red) is degraded only up to 92% of SOH. Hence, we do not consider the battery *A9* in our research.

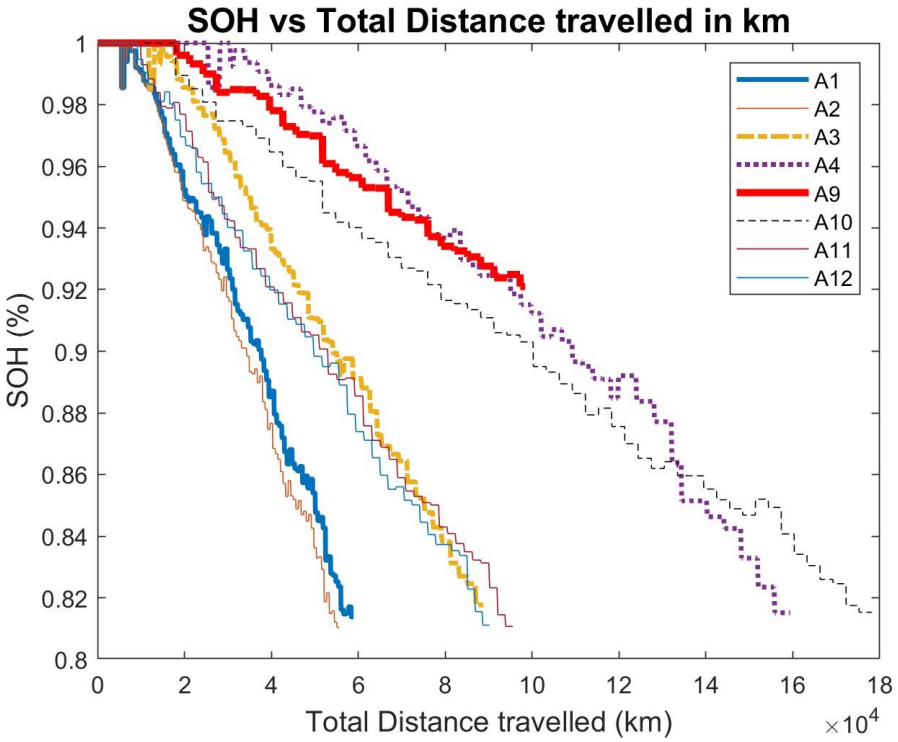


Figure 4.1: SOH (%) vs Total distance traveled (km)

Battery Id	A1	A2	A3	A4	A10	A11	A12
Maximum value of *Total_dist km* **before the end of life**	58922	55615	88798	159410	178220	95573	90189

Table 4.1: Table showing the maximum *Total dist km* values of all the cells in the TNO data set.

4.3 Data Pre-processing and Feature Engineering

4.3.2.1 Removing unwanted columns

The degradation history of batteries *A10, A11* and *A12* contained some duplicate features. These duplicate columns were removed. After the removal of duplicate columns, the degradation histories of all the batteries contain the same number of features.

4.3.3 Handling the missing values

The data set from TNO does not have any missing values. However, there were some instances of Inf present in the data set. Inf in MATLAB represents Infinity. These values were replaced by the average values computed by a moving average window length of 3.

4.3.4 Normalizing the capacity values

The range of values of the feature *Capacity As* in the degradation histories of different batteries present in the TNO data set are different. The maximum values of the feature *Capacity As* for all the batteries in the TNO data set are shown in Table 4.2.

The *Capacity As* feature of all the batteries in the TNO data set is normalized such that they range from 0 to 1. This is achieved using min-max scaling. Min-max scaling is a data pre-processing technique that scales a given feature such that the scaled feature ranges from 0 to 1. The equation for min-max scaling is shown in Equation 5.1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where *X'* is the scaled feature, *X* is the input feature, X_{max} is the maximum value of the input feature *X* and X_{min} is the minimum value of the input feature *X*. After the normalization the *Capacity As* values of all cells, they range from [0, 1]. Since during normalization the *Capacity As* values are divided by the maximum value, the scaled *Capacity As* values represent the state of health (SOH) of the cells.

Battery Id	**A1**	**A2**	**A3**	**A4**	**A10**	**A11**	**A12**
Maximum value of *Capacity As*	7231.33	7210.43	7062.24	7004.10	7106.28	7125.06	7047.30

Table 4.2: Table showing the maximum *Capacity As* values of all the cells in the TNO data set.

4.3.5 Adding new features

Three new columns are added to the degradation history of each cell. They are *voltage, current and temperature*. The values in these columns are filled by average values of voltage, current and temperature measurements recorded in the .mat files that are referred to in the *FileName* column of the degradation history. Whenever each cell is subjected to rest, it is recorded in the *FileName* column as *REST*. The rest duration of a cell is computed using the *duration* and the *FileName* columns and a new feature *rest duration* is created.

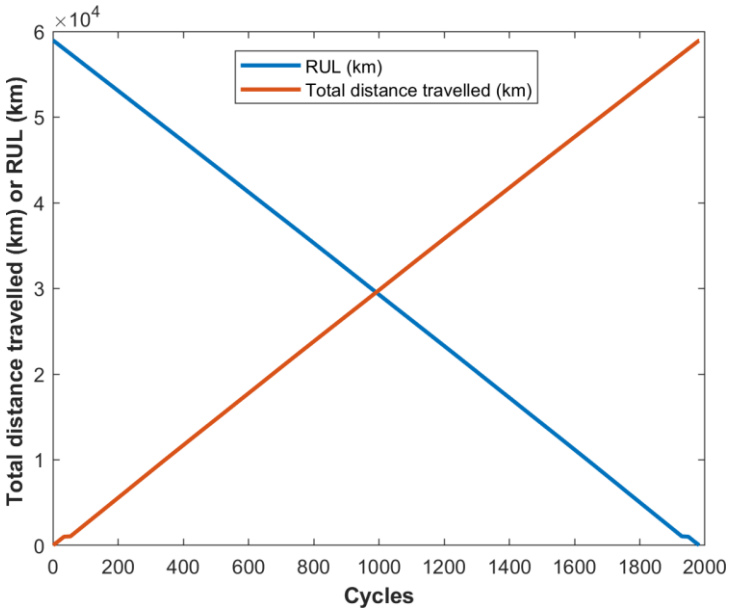


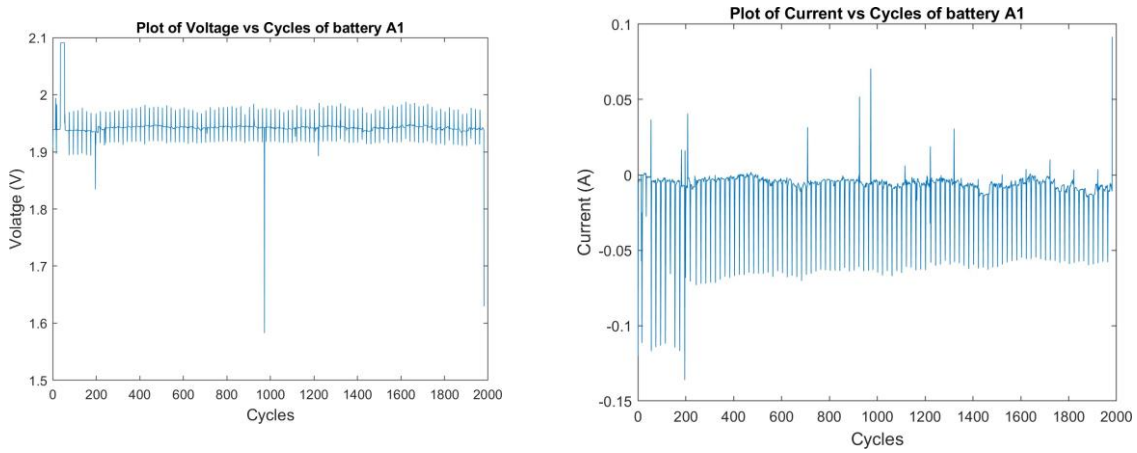
Figure 4.2: Total distance traveled (km) and RUL (km) for cell A1.

The *Total dist km* feature in the battery degradation history data set represents the simulated total distance in kilometers traveled by the EV. Sorting the *Total dist km* column in the descending
order gives the remaining distance in kilometers that the battery can power the EV before reaching the end of life. This is shown in Figure 5.2. The new column created by sorting the *Total dist km* is named as *rul km*.

4.3.6 Feature selection and Aggregation

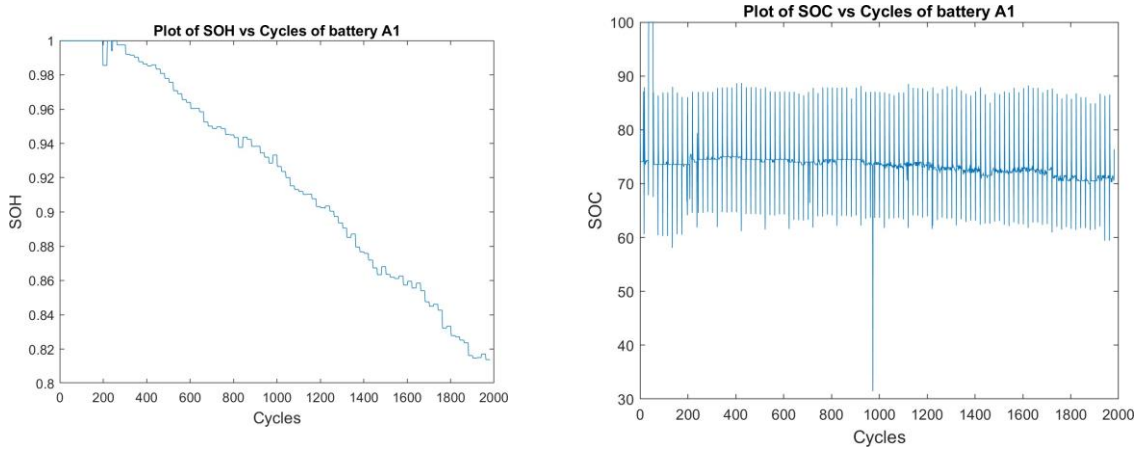
The RUL of a battery is influenced by various factors. The battery state of health (SOH) and state of charge (SOC) are some of the important factors that affect the RUL of a LIB (Nuhic et al., 2013). Hence, we also select *SOC* and *SOH* as the input features for our RUL prediction model. These values are generally computed by the battery management system (BMS). Battery internal resistance is one of the important health indicators by which we can infer the SOH of a battery. However, in an EV, battery internal resistance is measured occasionally by subjecting the battery to impulse tests. Thus, the battery internal resistance is not computed frequently. Hence, we do not choose this feature to develop our RUL prediction model.

SOC of a LIB is dependent on the voltage, current and temperature of the battery (Omariba, L. Zhang and D. Sun, 2018). Also, we want to consider the effects of calendar aging on the RUL of the battery. Hence, we also select *voltage, current, temperature and resting duration* as the input features for our RUL prediction model. We aggregate the following features by grouping them by the battery cycle identifier - *SOC, SOH, voltage, current, temperature, rest duration and rul km*. The *voltage, current, temperature, SOC* and *SOH* values of battery *A1* are grouped by the battery cycle identifier. The resulting features are shown in Figures 4.3, 4.4 and 4.5.



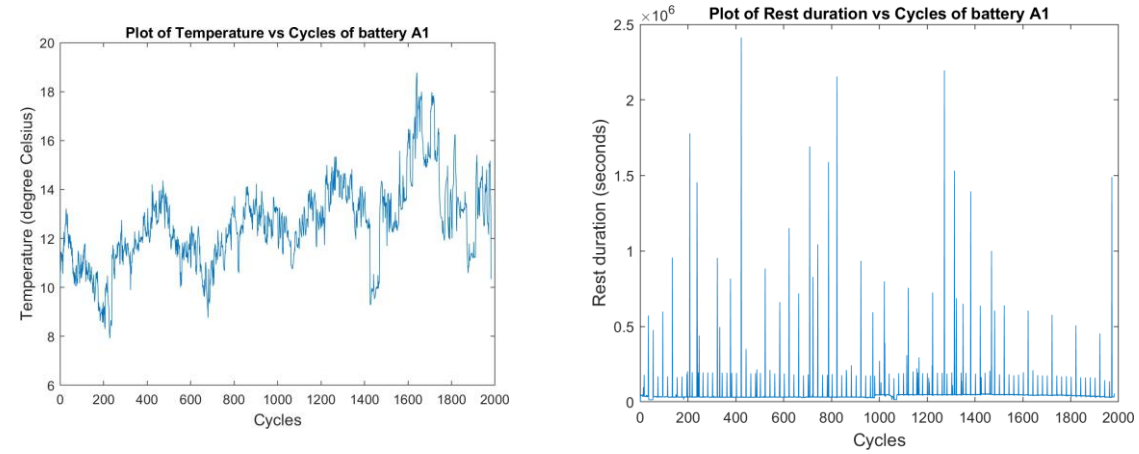
(a) (b)

Figure 4.3: a) Plot of Voltage vs Cycles of battery A1 b) Plot of Current vs Cycles of battery A1.



(a) (b)

Figure 5.4: a) Plot of SOH vs Cycles of battery A1 b) Plot of SOC vs Cycles of battery A1.



(a) (b)

Figure 5.5: a) Plot of Temperature vs Cycles of battery A1 b) Plot of SOC vs Cycles of battery A1.

Chapter 5

Machine Learning Models

In this chapter, we describe the details of StandardScaler, Artificial Neural Network (ANN), and the evaluation process to predict the remaining useful life (RUL) of lithium-ion batteries (LIB). In Section 5.1, we describe the working principle and the implementation details of the StandardScaler for feature scaling. In Section 5.3, we elaborate on the Artificial Neural Network (ANN) and its implementation details. In Section 5.5, we briefly describe the evaluation process for RUL prediction.

5.1 StandardScaler for Feature Scaling

Feature scaling is an essential step in data preprocessing, particularly in machine learning and deep learning algorithms. The performance and convergence speed of these algorithms can be significantly improved by ensuring that the input features are on a similar scale. One of the most commonly used techniques for feature scaling is the StandardScaler, which is a part of the sklearn.preprocessing module in Python.

The StandardScaler standardizes the features by removing the mean and scaling to unit variance. The formula for standardization used by StandardScaler is:

$$z = \frac{x - \mu}{\sigma}$$

where:

- x is an individual feature value,
- μ is the mean of the feature values,
- σ is the standard deviation of the feature values,
- z is the standardized value.

By transforming the data in this way, each feature will have a mean of 0 and a standard deviation of 1. This ensures that each feature contributes equally to the model's performance and prevents features with larger magnitudes from dominating the learning process.

By ensuring that all features are on a similar scale, we prevent any one feature from disproportionately influencing the model, leading to more accurate and reliable

predictions. This preprocessing step is particularly crucial when working with algorithms like neural networks, which benefit greatly from standardized inputs.

Implementation Details:

1. **Loading the Data**: We first load the training and testing datasets using Pandas and separate the features (X) from the target variable (y).

```
import pandas as pd

dataset = pd.read_csv('training_data.csv')
X_train = dataset.iloc[:, 0:5].values
y_train = dataset.iloc[:, 5].values

test_data = pd.read_csv('testing_data.csv')
X_005 = test_data.iloc[:, 0:5].values
y_005 = test_data.iloc[:, 5].values
```

Fig 5.1: Loading the data

2. **Applying StandardScaler**: We initialize the StandardScaler and apply it to the training and testing datasets.

```
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
sc = StandardScaler()

# Fit and transform the training data
X_train = sc.fit_transform(X_train)

# Transform the test data using the same scaler
X_005 = sc.transform(X_005)
```

Fig 5.2: Applying StandardScalar

5.2 Artificial Neural Network (ANN)

An Artificial Neural Network (ANN) is a type of deep learning model inspired by the human brain's structure. It consists of layers of neurons, including input layers, hidden layers, and output layers. Each neuron processes inputs through an activation function and passes the output to the next layer. This hierarchical structure enables the ANN to learn complex, non-linear relationships in the data.

ANNs are composed of layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron receives input, processes it through an activation function, and passes the output to the next layer. The network learns by adjusting the weights of the connections between neurons based on the error of the predictions.

However, they are prone to overfitting, especially when the tree becomes too deep. Pruning methods and ensemble techniques like Random Forests can mitigate this issue. Random Forests, by combining multiple decision trees trained on different subsets of data, enhance predictive accuracy and robustness. This ensemble approach reduces overfitting and increases the model's ability to generalize to new data.

- Initialization: We create an instance of the StandardScaler.
- Fitting and Transforming Training Data: We use 'fit_transform' on the training data. The 'fit' method calculates the mean and standard deviation for each feature, and the transform method applies the standardization formula to the data.
- Transforming Test Data: We use transform on the test data. It is crucial to use the same scaler fitted on the training data to transform the test data to ensure that both datasets are scaled in the same way.

- Improved Model Performance: By standardizing features, the model can converge faster during training, leading to better performance.
- Enhanced Gradient Descent Optimization: For neural networks, standardized inputs help in smoothening the gradient descent optimization process, thereby avoiding issues like vanishing or exploding gradients.

Implementation of ANN:

1. **Initialization**: We start by importing necessary libraries and initializing a Sequential model from Keras.

```
from keras.models import Sequential
from keras.layers import Dense
```

Fig 5.3: Initialization of Keras Libraries

- **Keras**: A high-level neural networks API that allows easy and fast prototyping. It runs on top of TensorFlow, CNTK, or Theano.
- **Sequential Model**: A linear stack of layers.

2. **Adding Layers**: We build the network by adding layers to the Sequential model.

```
# Initializing the ANN
regressor = Sequential()

# Adding the input layer and first hidden layer
regressor.add(Dense(output_dim=10, init='uniform', activation='tanh', input_dim=5))

# Adding the second hidden layer
regressor.add(Dense(output_dim=5, init='uniform', activation='tanh'))

# Adding the output layer
regressor.add(Dense(output_dim=1, init='uniform', activation='linear'))
```

Fig 5.4: Adding Neural Network Layers

- **Input Layer and First Hidden Layer**:

- Dense: A fully connected layer where each neuron receives input from all neurons of the previous layer.
- output_dim: Number of neurons in the layer (10 in the first hidden layer).
- init: Initialization method for the weights (uniform distribution).
- activation: Activation function ('tanh' for non-linearity).
- input_dim: Number of input features (5).

3. **Compiling the Model:** We compile the model by specifying the optimizer, loss function, and metrics.

```
#Compile the ANN
regressor.compile(optimizer='sgd', loss='mean_squared_error', metrics=['mean_absolute_error', threshold_accuracy])
```

Fig 5.5: Compiling the Model

 Optimizer: Stochastic Gradient Descent (SGD), which updates weights iteratively to minimize the loss.
 Loss Function: Mean Squared Error (MSE), suitable for regression tasks.
 Metrics: Mean Absolute Error (MAE), used to monitor the training process.

4. **Training the Model:** We fit the model to the training data.

```
# Fitting the ANN to the Training set
regressor.fit(X_train, y_train, batch_size=1, nb_epoch=2000)
```

Fig 5.6: Training the Model

- **Batch Size**: Number of samples processed before the model is updated. Smaller batch sizes lead to more frequent updates.
- **Epochs**: Number of complete passes through the training dataset. 20 epochs mean the model will see the entire dataset 20 times.

5. **Making Predictions**: After training, we use the model to make predictions on the test data.

```
# Predicting the Test set result
y_pred = regressor.predict(X_005)
```

Fig 5.7: Making Predictions

The predict method generates predictions for the input test data X_005.

5.3 Support Vector Regression - Implementation

Two sets of experiments were conducted for each cell (except cell A9) in the TNO data set. In the first experiment, X_{CA} (refer Section 5.3.8) was fed as the input and in the second experiment, $X_{no\ CA}$ was given as the input. In both experiments, separate SVR models were trained for each cell with y_{RUL} as the target variable. The SVR models were trained on 70% of the degradation history data of the cell and were tested on the remaining 30% of the data.

The SVR models were trained with automatic hyperparameter optimization. The optimal hyper- parameters were selected with the help of Bayesian optimization (Snoek, Larochelle and Adams, 2012). Bayesian optimization is a commonly used optimization algorithm to find the global maximum or the minimum of a function (Agnihotri and Batra, 2020). The optimal hyperparameters for the SVR models are listed in Tables 6.1 and 6.2.

The predictions made by the LSTM network models are transformed back by multiplying the values with their corresponding standard deviation values and adding the mean value.

Battery Id	Kernel	Polynomial Order	Epsilon
A1	Polynomial	2	0.0026
A2	Linear	-	0.0955
A3	Polynomial	2	0.0024
A4	Linear	-	0.0008
A10	Polynomial	3	0.0009
A11	Polynomial	2	0.001
A12	Polynomial	2	0.04

Table 5.1: Optimal Hyperparameters of SVR models (with calendar aging)

Battery Id	Kernel	Polynomial Order	Epsilon
A1	Polynomial	2	0.0009
A2	Linear	-	0.0505
A3	Polynomial	3	0.0019
A4	Linear	-	0.0078
A10	Gaussian	-	0.0009
A11	Polynomial	3	0.0104
A12	Polynomial	2	0.0405

Table 5.2: Optimal Hyperparameters of SVR models (without calendar aging)

5.4 Evaluation Process

The evaluation process is a critical step in machine learning and deep learning projects. It involves assessing the model's performance on unseen data to ensure that it generalizes well and does not overfit the training data. In this section, we describe the methods used to evaluate the Artificial Neural Network (ANN) implemented in the provided code for predicting the remaining useful life (RUL) of lithium-ion batteries.

The main goal of the evaluation process is to compare the model's predictions against the actual values in the test dataset. This comparison helps in understanding how well the model has learned the underlying patterns in the data and how accurately it can predict new, unseen instances. Key components of the evaluation process include:

- **Prediction**: Using the trained model to generate predictions for the test dataset.
- **Comparison**: Comparing the predicted values with the actual values.
- **Visualization**: Plotting the results to visualize the model's performance.
- **Error Metrics**: Calculating metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) to quantify the prediction errors.

1. **Implementation Details:**
   - **Prediction:** After training the ANN, we use it to make predictions on the test data X_005. The predict method is used for this purpose.

```
# Predicting the Test set result
y_pred = regressor.predict(X_005)
```

Fig 5.8: Prediction from ANN

   - **y_pred**: This variable stores the predicted RUL values for the test data.

2. **Initialization Capacity:**

Before plotting, we calculate the initial capacity based on the first test instance, scaled by a factor (0.7 in this case). This step is specific to the application and is used to provide a reference line in the plot.

```
listt = X_005[0]
init_capacity = 0.7 * regressor.predict(listt)
```

Fig 5.9: Initializing Capacity

- listt: The first instance from the test data.
- init_capacity: Initial capacity value for reference in the plot.

3. **Plotting The Results:**
   We visualize the training data, actual test data, and predicted test data using Matplotlib.point.

```
import matplotlib.pyplot as plt

# Prepare the x-axis values for plotting
x_inp = [i for i in range(1, len(y_pred) + 1)]

# Plot the training data, actual test data, and predicted test data
plt.plot(X_train, y_train, color='violet', label='Training Data')
plt.plot(x_inp, y_005, color='red', label='Actual Test Data')
plt.plot(x_inp, y_pred, color='blue', label='Predicted Test Data')
plt.title('RUL Prediction')
plt.xlabel('Cycle')
plt.ylabel('Capacity (Ah)')
plt.legend()
plt.show()
```

Fig 5.10: Plotting the Results

- **x_inp**: A list of cycle numbers corresponding to the test data instances.
- **plt.plot()**: Used to plot the different datasets. Each plot is labelled for clarity.
- **plt.legend()**: Adds a legend to the plot for better understanding.

4. **Error Metrics Calculation:**

Although not explicitly mentioned in the initial code, it is standard practice to calculate error metrics to quantitatively evaluate the model's performance. Two common metrics are:

- **Mean Absolute Error (MAE)**: The average of the absolute errors between the predicted and actual values.
- **Mean Squared Error (MSE)**: The average of the squared errors between the predicted and actual values.

These can be calculated as follows:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Calculate MAE and MSE
mae = mean_absolute_error(y_005, y_pred)
mse = mean_squared_error(y_005, y_pred)

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
```

Fig 5.11: Error Metrics Calculation

- **mean_absolute_error(y_005, y_pred)**: Computes the MAE between the actual and predicted test data.
- **mean_squared_error(y_005, y_pred)**: Computes the MSE between the actual and predicted test data.

5. **Interpreting the Results:**

- **Visualization**: The plots provide a visual comparison of the predicted and actual RUL values. A good model will have predicted values (blue line) closely following the actual values (red line).
- **Error Metrics**: Lower values of MAE and MSE indicate better model performance. These metrics help quantify the accuracy of the predictions and can be used to compare different models or configurations.

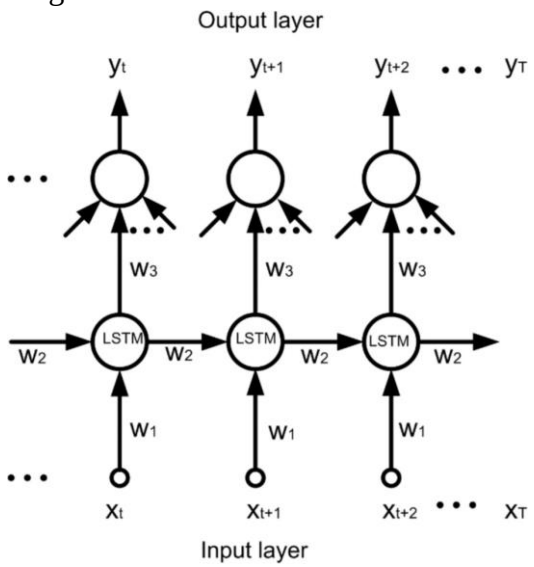


Figure 5.12: Representation of forward propagation through time in ann neural network

5.5 Bat-PF

Two sets of experiments were conducted for each cell (except cell A9) in the TNO data set. In the first experiment, X_{CA} (refer Section 5.3.8) was fed as the input and in

the second experiment, $X_{no\ CA}$ was given as the input. In both experiments, separate LSTM models were trained for each cell with y_{RUL} as the target variable. The SVR models were trained on 70% of the degradation history data of the cell and were tested on the remaining 30% of the data.

The architecture of the implemented LSTM network is shown in Figure 4.4. The same architecture is used for each LSTM model implemented. The LSTM network consists of an input layer. The input layer is connected to the first LSTM layer which consists of 150 LSTM cells. This LSTM layer is connected to a dropout layer. Dropout is regularization technique proposed by (Srivastava et al., 2014) and it is commonly used to prevent deep learning models from overfitting. Dropout probability is set to 0.1. The dropout layer is connected to a LSTM layer which has 300 LSTM cells. This LSTM layer is then connected to a dense layer, followed by a regression layer which generates the prediction. Two experiments were conducted on the LSTM network. In both experiments, each model is run for 500 epochs. The learning rate of the model is set to 0.005. The learning rate is multiplied by a factor of 0.2 after every 250 epochs. The gradient descent during the training procedure of the models is optimized using the Adam optimizer (Kingma and Ba, 2015).

In the first experiment, which considers the effects of calendar aging, the number of trainable parameters in the different layers of the network are as follows - the first LSTM layer, the second LSTM layer and the dense layer had 94200, 541200 and 300 trainable parameters respectively. The total number of trainable parameters are 635700.

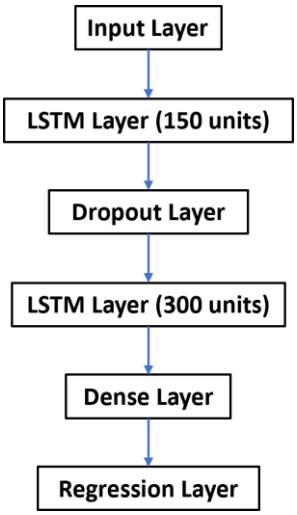


Figure 5.4: The architecture of the implemented LSTM model for RUL prediction.

In the second experiment without considering the effects of calendar aging, the number of trainable parameters in the different layers of the network are as follows - the first LSTM layer, the second LSTM layer and the dense layer had 93600, 541200 and 300

trainable parameters respectively. The total number of trainable parameters are 635100. The results and performance of the two LSTM models are discussed in detail in Section 7.3.

The predictions made by the LSTM network models are transformed back by multiplying the values with their corresponding standard deviation values and adding the mean value.

5.6 Similarity Based Model

There is a category of models that can predict the remaining useful life (RUL) of a test lithium-ion battery based on the degradation profiles of the lithium-ion batteries that are similar. Such models are called similarity-based models (SBM).

The main intuition behind similarity-based model for RUL prediction is that systems that have similar degradation histories have similar RULs. This is evident from Figure 5.1 and Table 5.1. Cells *A1* and *A2* have very similar degradation profiles. Cell *A1* and *A2* reach EOL after powering the vehicle for 58922 km and 55615 km respectively. A similar pattern can be observed between cells *A11* and *A12*.

5.6.1 Working principle of a similarity-based model (SBM)

In this section, we describe the basic working principle of a similarity-based model for RUL pre- diction.
Let $TS = \{T_1, T_2, T_3, \quad T_N\}$ be the library of time series representing the complete degradation
histories of N similar systems namely $S_1, S_2, S_3, \quad S_N$. In other words, TS represents the run-to-
failure (RTF) data of N similar systems. Each time series in TS can be univariate or multivariate. In the context of this research, every element of the set TS represents the SOH
Let T_Q be the degradation history of a test system Q. The system Q has not reached its end of life. We want to predict the RUL of T_Q. Let *val* be the value of the present system health of the system Q.
For every degradation history T_i in TS where $i \in [1, N]$, let $T_i(val)$ represent the portion of T_i from its beginning of life (BOL) to *val*. Since, TS represents the RTF data of N systems, we know the RUL of these N systems at any value of their recorded system health. Let $y_{Si}(val)$ represent the RUL of system S_i at the system health value of *val*, where $i \in [1, N]$.

The RUL of the test system Q can be computed using SBM as follows:

1. We have to define a function" dist" to compute the distance between two degradation histories. If two degradation histories are similar, the *dist* function should return a smaller value and it should return a larger value otherwise.
2. Compute the distance between the test degradation history Q and all the degradation his- tories in the set TS, .i.e,

$$[d_i] \leftarrow dist(T_i(val), T_Q(val)), \forall i \in [1, N] \qquad (5.13)$$

where $[d_i]$ represents the list of distances between the test system Q and N training systems.
degradation curve of a lithium-ion battery.

where $[d_i]$ represents the list of distances between the test system Q and N training systems.

1. Sort the distances in ascending order. Let $[d_{i'}]$ be the list of sorted distances where i represents the sorted index and $i' \in [1,N]$. The
2. Select the k smallest distances. This implies that we are selecting k nearest degradation histories that are similar to the degradation history of Q.

$$[d_{i'}], \forall i \in [1, k] \qquad (5.14)$$

3. The RUL of the test system Q that has system health value of *val* is computed as follows:

$$y_Q(val) \leftarrow \frac{1}{k} \sum_{i'=1}^{k} y_{S_{i'}}(val) \qquad (5.15)$$

Here, the RUL of the test system Q is calculated by taking the average of the RUL values of k systems that are similar to the test system Q. The RUL values of the k similar systems have an equal weight of 1. Therefore, in the rest of this research, we refer to this strategy of computing RUL as SBM with uniform weights.

Also, the RUL values of k similar systems that are similar to the test system Q can be weighted such that the RUL values of the systems that are the most similar to the test system are given higher weights and the RUL values of the systems that are the least similar to the test system can be given lower weights. The weighted average of these RUL values gives the RUL of the test system. This is mathematically expressed as follows:

$$y_Q(val) \leftarrow \frac{\sum_{i'=1}^{k} w_{i'} y_{S_{i'}}(val)}{\sum_{i'=1}^{k} w_{i'}(val)}$$

$$w_{i'} = \frac{1}{d_{i'}} \qquad (5.16)$$

In the rest of this research, we refer to this strategy of computing RUL as *SBM with inverse distance weights*. *SBM with uniform weights* is a special case of *SBM with inverse distance weights*, when all the k nearest degradation histories are at an exactly same distance from the query degradation history.

Earlier, while describing the steps to predict RUL using SBM, we defined a function *dist*, which is used to compute the distance between two degradation histories. If the two degradation histories are of equal length, then pointwise Euclidean distance or absolute distance can be easily calculated. However, the lengths of two degradation histories will not be equal in most cases. Hence, we have to define the *dist* function such that it can compute the distance between two degradation histories that have different lengths. One of the most commonly used distance functions to compute the distance between two time series (degradation histories) is Dynamic Time Warping (DTW). The working principle of DTW is explained in detail in Section 5.5.2

5.6.2 Dynamic Time Warping (DTW)

Dynamic Time Warping is one of the most commonly used algorithms to compute the similarity between two different asynchronous time series. DTW was first proposed by (Sakoe and Chiba, 1978), where the researchers used it for spoken word recognition. DTW compares two time series and measures the similarity between them by computing the optimal warping path between them. DTW has been commonly used in the field of speech recognition, human computer interaction, data mining and signal processing.

Let $X = \{x_1, x_2, ...x_N\}$ and $Y = \{y_1, y_2, \quad y_M\}$ be two univariate time series of lengths N and

M respectively. In the context of this research, X and Y represent the degradation profiles of two lithium-ion batteries. The main idea behind DTW is to compute the optimal warping path W, between X and Y. Once the warping path is computed, X and Y are extended by warping non-linearly with respect to time. The similarity between extended X and Y are can be computed easily as they have an equal number of points after warping.

Given X and Y, the optimal warping path W is expressed as below:

$$W = \begin{matrix} W_x(k) \\ W_y(k) \end{matrix}, k = 1, 2, \cdots, p \qquad (5.18)$$

where $W_x(k)$ corresponds to an index of X and $W_y(k)$ corresponds to an index of Y. The length of the warping path W is p.

The computed warping path W satisfies the following conditions. Firstly, all the indices of X and Y should be used in the warping path W. Secondly, the warping path W should be continuous and it should be monotonically increasing. Given that the warping path W satisfies these conditions, the starting point of W has the indices $(1,1)^T$ and the ending point of W has the indices $(N,M)^T$. Along with these conditions, any two adjacent points $W(k)$ and $W(k+1)$ in the warping path W should satisfy the following inequality.

$$W_x(k) \leq W_x(k+1) \leq W_x(k) + 1 \qquad (5.19)$$

$$W_y(k) \leq W_y(k+1) \leq + 1$$

Hence, W(k+1) can take only the following values: $(W_x(k), W_y(k+1))^T$, $(W_x(k+1), W_y(k))^T$ and $(W_x(k+1), W_y(k+1))^T$. This implies that the warping path W can move either horizontally or vertically or diagonally and it can never go backward. The DTW algorithm chooses the direc- tion which has the lowest cost. Also, the length of W .i.e, $p \in [max(M, N), M + N]$.

By using the optimal warping path W, the two input time series X and Y can be extended to two new time series $\tilde{x}$ and $\tilde{y}$ respectively which are expressed as below:

$\tilde{X} = X(Wx(k))$

$\tilde{Y} = Y(Wy(k))$ $\quad k = 1, 2, \cdots, p$

Now that $\tilde{X}$ and $\tilde{Y}$ are of same length, the pointwise distance between them can be calculated as follows.

$$DTW(X, Y) = D(\tilde{X}, \tilde{Y}) = \sum_{k=1}^{p} (W_x(k)), Y(W_y(k))) \qquad (5.21)$$

In Equation 5.21, D is a distance metric. The following distance metrics are commonly used.

- Euclidean distance, which is computed as the root sum of squared differences between the points.

$$pD(\tilde{X}, \tilde{Y}) = \sqrt{(\tilde{X}(k) - \tilde{Y}(k))(\tilde{X}(k) - \tilde{Y}(k))} \qquad (5.22)$$

- Absolute distance, which is computed as the sum of absolute differences between the points. It is also known as the Manhattan distance or city block distance or taxicab distance metric.

$$D(\tilde{X}, \tilde{Y}) = \sum_{k=1}^{p} |\tilde{X}(k) - \tilde{Y}(k)| = \sum_{k=1}^{p} \sqrt{(\tilde{X}(k) - \tilde{Y}(k))(\tilde{X}(k) - \tilde{Y}(k))} \qquad (6.23)$$

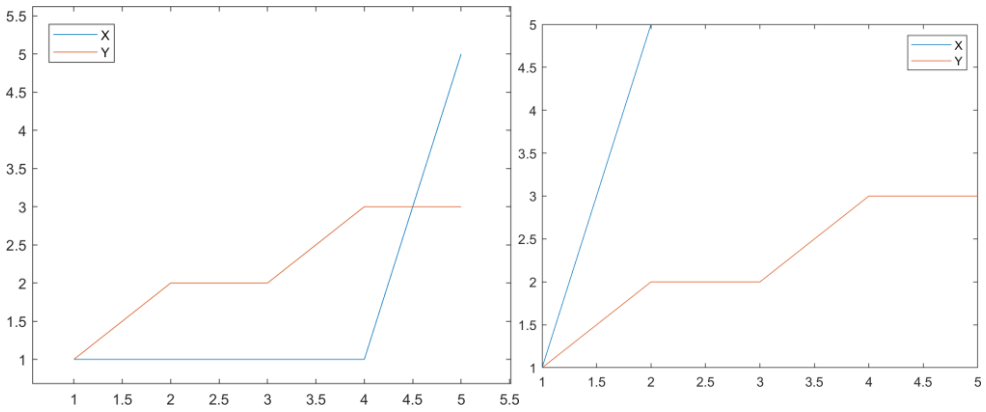
The warping path is computed as follows. Firstly, a cost matrix $C(i, j), where i \in 1, 2, ...N$ and

$j \in 1, 2, ...M$ is created. Each element of $C(i, j)$ is computed using equation 5.25.

$$C(i, j) = D(X(i), Y(j)) + \min \begin{cases} C(i-1, j-1) \\ C(i-1, j) \\ C(i, j-1) \end{cases} \qquad (5.25)$$

where $C(1, 1) = D(X(1), Y(1))$. The path with the lowest cost that connects $C(1, 1)$ and $C(M, N)$ is the optimal path W.

An illustration of DTW is shown in Figure 5.5 a) and Figure 5.5 b). Figure 5.5 a) consists of two input time series $X = [1, 5]$ and $Y = [1, 2, 2, 3, 3]$. We can observe that X and Y have different lengths. Hence, pointwise distance can be calculated on the warped time series. The DTW distance between X and Y is 6, when Euclidean distance metric is used.



(a) (b)

Figure 5.5: Plot of two sample time series X and Y a) before DTW b) after DTW.

5.7 Working of Neural network using all Parameters

Predicting the Remaining Useful Life (RUL) of lithium-ion batteries is critical for ensuring the reliability and efficiency of systems that depend on these batteries, such as electric vehicles and portable electronics. Using neural networks for this task leverages their ability to model complex, non-linear relationships among various operational parameters and degradation indicators. The input parameters typically include features such as voltage, current, temperature, charge/discharge cycles, and capacity fade, among others. These features collectively provide a comprehensive picture of the battery's health and performance over time.

A neural network can be designed with an input layer corresponding to the number of features used for prediction. For instance, if five parameters (voltage, current, temperature, charge cycles, and capacity) are monitored, the input layer would have five nodes. The neural network can then include one or more hidden layers, each with a certain number of neurons and activation functions like tanh or ReLU (Rectified Linear Unit). These hidden layers enable the network to learn intricate patterns and interactions between the input features that are crucial for accurate RUL prediction.

The training process involves feeding historical data of battery usage and corresponding RUL values into the neural network. This data is often standardized to ensure that each feature contributes equally to the learning process, preventing any single feature from disproportionately influencing the predictions. The network is trained by adjusting the weights and biases of the neurons through a process called backpropagation, which minimizes the error between the predicted and actual RUL values. The error is typically measured using mean squared error (MSE), which quantifies the average squared difference between predicted and true RUL values.
To optimize the training process, techniques such as stochastic gradient descent (SGD) are employed. SGD updates the model parameters iteratively for each training example, which can lead to faster convergence, especially in large datasets. Training over multiple epochs ensures that the model has sufficiently learned the underlying patterns in the data. Regularization methods and dropout techniques may also be incorporated to prevent overfitting, ensuring that the model generalizes well to new, unseen data.
Once trained, the neural network can predict the RUL of a lithium-ion battery based on real-time input parameters. This predictive capability is crucial for proactive maintenance and operational planning, allowing for timely interventions before the battery fails. For instance, in electric vehicles, accurately predicting the RUL can optimize battery usage and extend its lifecycle, thereby reducing costs and enhancing performance.

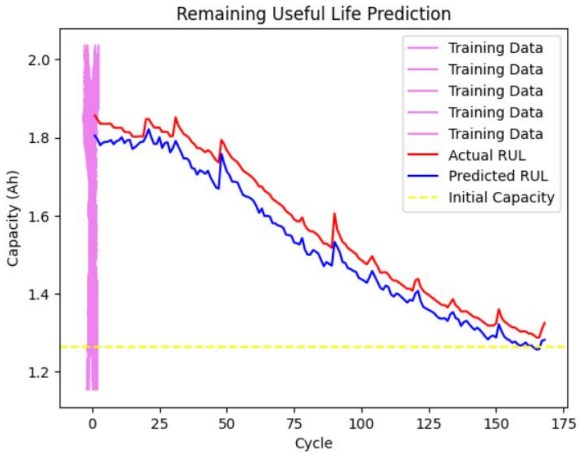


Fig: 5.6 Prediction Graph of Battery showing initial capacity

Visualization of the predicted RUL against actual values provides a clear understanding of the model's accuracy. In practice, this involves plotting the predicted RUL values over time and comparing them with the true RUL values observed during testing. Such visualizations can highlight the model's strengths and areas for improvement, guiding further refinement of the neural network architecture and training process.

Chapter 6

Results

In the last chapter, we implemented Support Vector Regressor (SVR), Long Short-Term Memory (LSTM) and Similarity based Model (SBM) to predict the RUL of the batteries. In this chapter, we present the results obtained from the various machine learning models that were implemented and we discuss the results. In Section 7.1, we define evaluation metrics which we will use to evaluate the performance of the RUL prediction models. In Section 7.2, we discuss the results of the SVR model. In Section 7.3, we discuss the results of the LSTM model. The results of SBM model are discussed in Section 7.4.

6.1 Model performance metrics

In this section, we discuss some of the model evaluation metrics which are used to evaluate the performance of the implemented prediction models. There are numerous model evaluation metrics proposed in the literature, however, each metric has its own merits and demerits to interpret a model. Thus, we choose some of these model evaluation metrics based on their significance and explain ability of the results.

6.2 Mean Absolute Error (MAE)

Mean Absolute Error is computed as the average of the absolute value of the error. MAE is calculated as follows:

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \quad (6.1)$$

where y_i and $\hat{y}_i$ are the actual values of the target and the predicted values of the target respectively, at the i^{th} sample.

6.3 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is one of the commonly used model evaluation metrics. MAPE is calculated as follows:

$$MAPE = \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{|y_i|} \quad (6.2)$$

where y_i and $\hat{y}_i$ are the actual values of the target and the predicted values of the target respectively, at the i^{th} sample. MAPE is expressed as a percentage. We can observe that, if the actual value of a target is zero, then MAPE will be infinity.

6.4 Root Mean Squared Error (RMSE)

It is one of the most commonly used metrics to evaluate the performance of regression and fore- casting models. As the name suggests, RMSE is the square root of the square of the mean of the squared errors. The mathematical formula to compute RMSE is given below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \quad (6.3)$$

where y is the actual value of the target and $\hat{y}$ is the predicted value of the target. N is the length of $\hat{y}$ and also y.

6.5 Coefficient of Determination (R^2)

This metric indicates the goodness of fit of a regression model. In a regression problem, R^2 metric is used to explain how well the regression line fits the input data. When comparing the performance of two or more regression models, a model with higher value of R^2, fits the input data better than the others. Typically, R^2 of a model is in the range [0, 1]. However, in some cases, the R^2 metric of a model is negative, which implies that the model has an opposite trend as the input data.

R^2 is computed as follows. Let y be the actual target of length N. Let $\hat{y}$ be the predicted target. The mean of the actual values is computed as below.

$$\bar{y} = \frac{\sum_{i=1}^{N} y_i}{N} \quad (6.4)$$

The total sum of squares is computed as $SS_{tot} = \sum_{i=1}^{N} (y_i - \bar{y}_i)^2$. The sum of squares of the residual is computed as $SS_{res} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$. Using SS_{tot} and SS_{res}, R^2 is computed as below.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \quad (6.5)$$

One of the main advantages of using R^2 as a model evaluation metric is that it is scale independent. However, it is sensitive to the variance in the input data.

6.6 Model Running Time

Model running time is an important model evaluation metric. Generally, most machine learning (ML) models take a lot of time and computational resources to generate predictions. In the context of RUL prediction of lithium-ion batteries in electric vehicles, the ML model to compute the RUL will be deployed in the control unit of the EV's battery management system (BMS). The BMS of an EV is an embedded hardware and has very fewer computing resources as compared to a personal computer. Hence the deployed ML model must make use of the computational resources of the BMS with high efficiency. Hence, an ideal RUL prediction model should take as less time and consume as less computational resources as possible.

6.7 Support Vector Regression Model

In this section, we present the results obtained using the Support Vector Regression (SVR) model. Figures 7.1 a) to d) and Figures 7.2 a) to c) illustrate the RUL prediction made by the SVR model for all the batteries in the TNO data set (except battery A9). Table 7.1 shows the performance of SVR model to predict RUL for each cell when the effects of calendar aging is considered and Table 7.2 shows the performance of SVR model to predict RUL for each cell when the effects of calendar aging is not considered.

For cell *A1*, both the models with calendar aging and without calendar aging have a decent R^2 value of around 90%. However, for the cells *A3, A11 and A12*, the prediction made by the model without calendar aging effect do not follow the trend of the actual target values. Hence, the R^2 value of these models for cell *A3* are negative. The SVR model with calendar aging for cell *A3* also has a negative R^2 value.

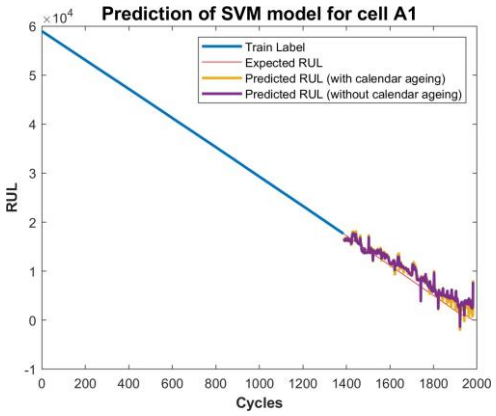
For cell *A10*, the SVR model with calendar aging effects almost follows the trend of the actual tar- get values. But the prediction made by the model during the last cycles deviates too much from the expected target. Hence, the R^2 of the SVR model with calendar aging for cell *A10* is lesser than that of the SVR model without calendar aging. For cell *A4*, the predicted RUL has a slightly sim- ilar trend to that of the actual target, but the predictions deviate too much from the actual target.
For the batteries *A3, A11* and *A12*, the SVR model without calendar aging effects has a negative R^2 score, which indicates that the model incorrectly fits the data. But, the SVR model with calendar aging effects has a negative R^2 score for only one out of 7 batteries. Thus, the SVR model with calendar aging effects is a better fitting model than the SVR model without calendar aging effects.

Battery Id	MAE (km)	MAPE	RMSE	R^2	MRT(s)
A1	1369.4	99.21	1580.8	0.9074	642.84
A2	3222.1	Inf	3439.5	0.4893	667.97
A3	7744.7	337.44	9785.6	-1.0052	580.56
A4	8630.6	82.11	9968.8	0.4495	534.96
A10	6589.4	518.61	10116	0.5803	1638.71
A11	4209.3	170.67	5255.1	0.4533	468.08
A12	4480.5	225.34	5800.4	0.1152	410.86

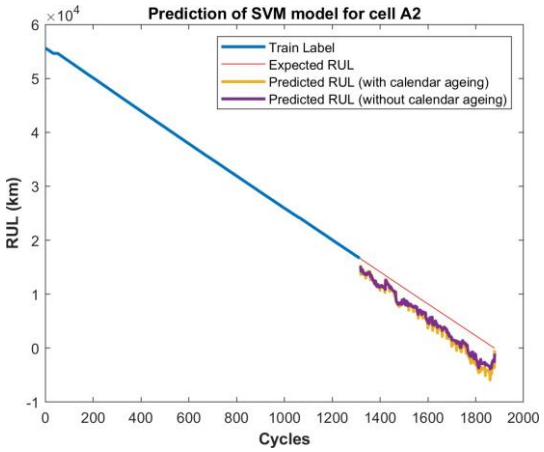
Table 6.1: Performance metrics of SVM regress

Battery Id	MAE (km)	MAPE	RMSE	R^2	MRT(s)
A1	1447.1	123.30	1681.3	0.8953	721.80
A2	2848.8	Inf	2976.3	0.6176	820.67
A3	20942	750.80	26817	-14.05	593.62
A4	9365.3	94.70	10850	0.3478	650.78
A10	6859.6	144.94	8349.3	0.7141	981.35
A11	15285	625.01	19472	-6.5065	391.99
A12	6042.9	290.19	7644.7	-0.537	470.67

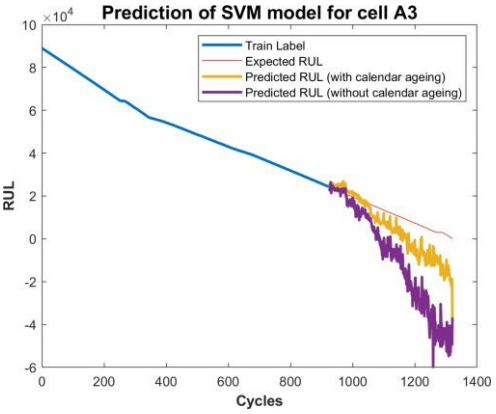
Table 6.2: Performance metrics of SVM regression models without calendar aging ion models with calendar aging.



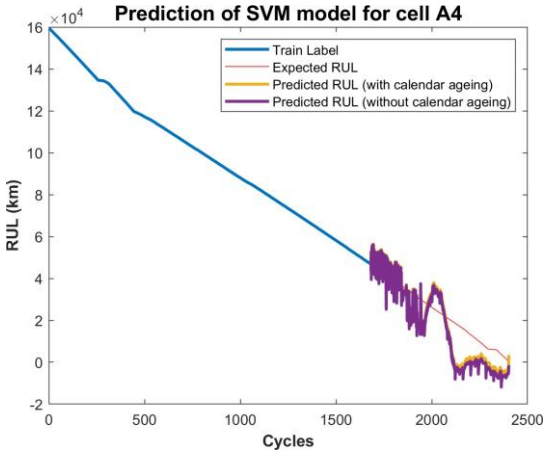
(a)



(b)



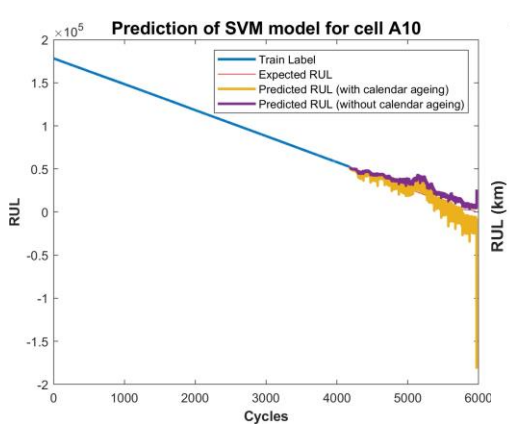
(c)



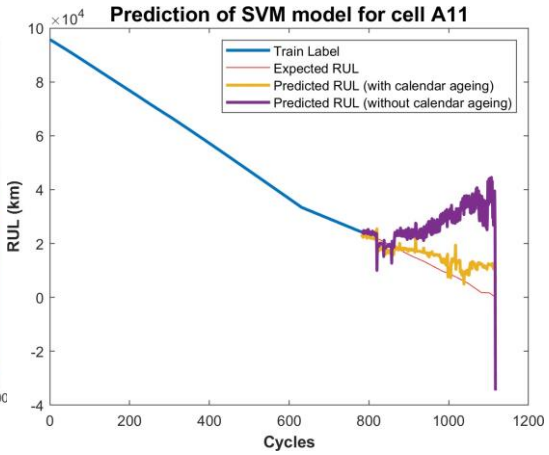
(d)

Figure 6.1: RUL prediction of LSTM model for cell a) A1 b) A2 and c) A3 and d) A4

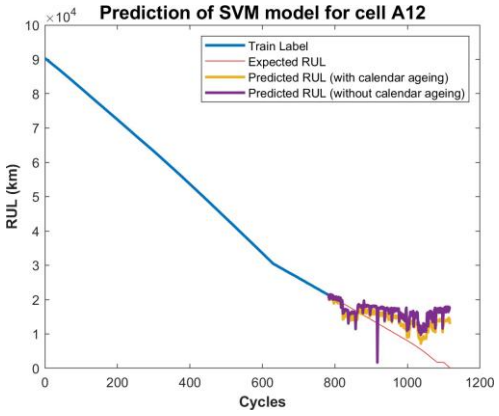
Also, for most batteries (except batteries *A10* and *A11)*, the model running time (MRT) of the SVR model with calendar aging effects is lower than the SVR model without calendar aging effects. Hence, we can prefer the former model over the latter. In Tables 6.1 and 6.2, the MAPE value for battery A2 is infinity. This is because, the actual value of the RUL at the last cycle of the battery is zero and while computing the MAPE, the absolute error term in the numerator is being divided by zero.



(a)



(b)



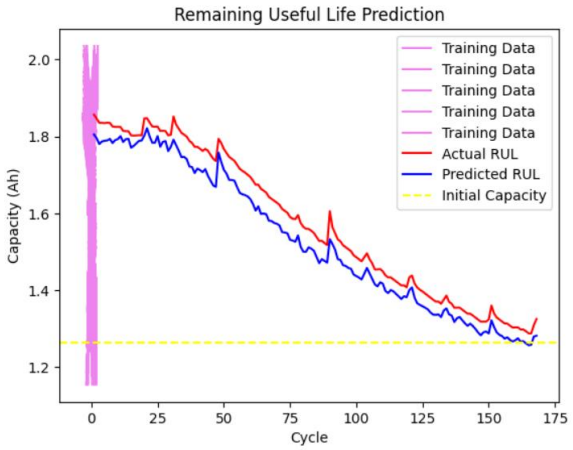
(c)

Figure 6.2: RUL prediction of SVM Regressor for cell a) A10 b) A11 and c) A12

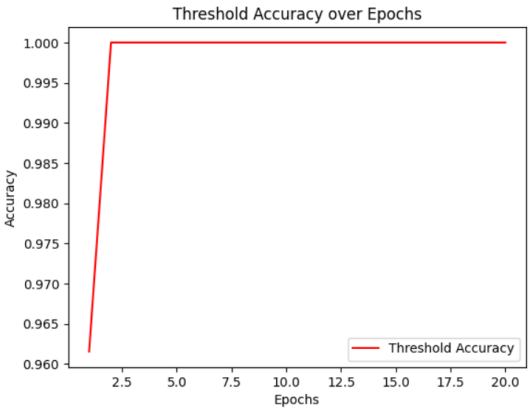
6.6 Findings

If we consider all the parameters (i.e cycle number, voltage, current, temperature and time) as input to the neural network to calculate the capacity for a particular cycle then the accuracy of the model increases i.e R2=0.9766 RMSE=0.0303 This capacity can be used for further calculations.

For battery *A3*, the MAPE value is 42.30%, which is significantly lesser as compared to the MAPE values of the LSTM models for other batteries. Overall, if we considering all the model evaluation metrics in Table 6.3, we can conclude that the LSTM model with calendar aging effects is a better model than the LSTM model without calendar aging. The RUL predictions made by the LSTM model are relatively good as compared to the predictions made by the SVR model. Hence, among SVR model and LSTM model, LSTM is a better performing model.

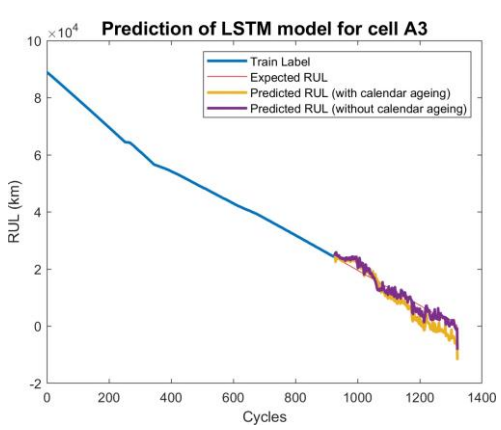


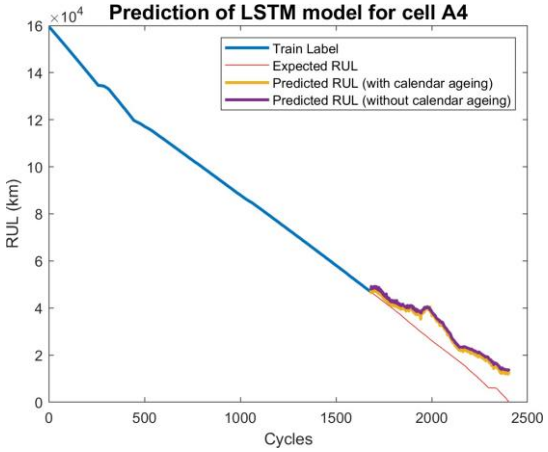
(a)



(b)

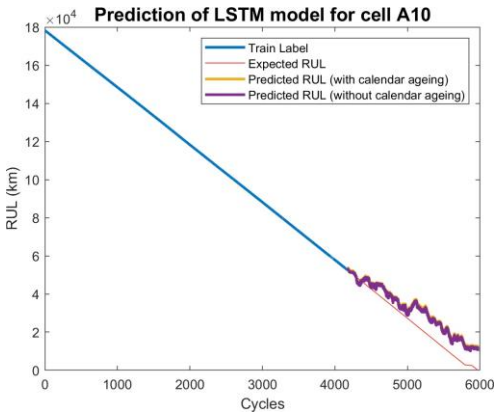
Fig: 6.2 Epochs vs Accuracy (b) Cycle vs Capacity



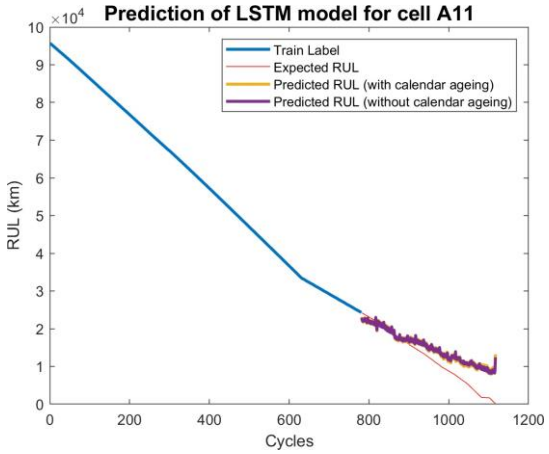


(c) (d)

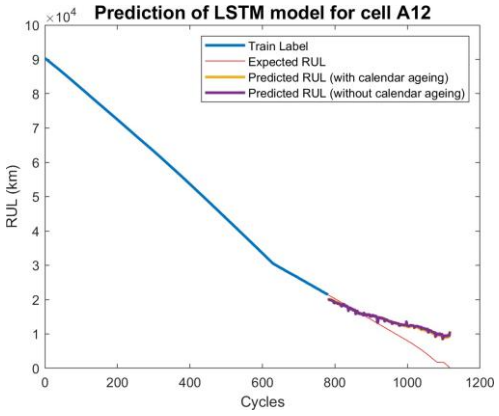
Figure 6.3: RUL prediction of LSTM model for cell a) A1 b) A2 and c) A3 and d) A4.



(a)



(b)



(c)

Figure 6.4: RUL prediction of LSTM model for cell a) A10 b) A11 and c) A12

Battery Id	MAE (km)	MAPE	RMSE	R^2	MRT(s)
A1	2512.9	194.71	2967.6	0.6737	113.10
A2	1746.4	Inf	2144.3	0.8015	116.26
A3	3152.7	112.13	3731.8	0.7084	84.91
A4	7065.6	112.90	7830.9	0.6603	154.95
A10	7471.3	174.85	8438.4	0.7080	376.51
A11	3260.1	159.65	4157.0	0.6579	77.28
A12	3357.1	163.59	4382.2	0.4950	75.79

Table 6.3: Performance metrics of LSTM models with calendar aging

Battery Id	MAE (km)	MAPE	RMSE	R^2	MRT(s)
A1	1817.5	155.70	2198.1	0.8210	125.76
A2	1954.3	Inf	2443.9	0.7422	141.59
A3	1571.3	42.30	1923.4	0.9225	162.76
A4	8246.2	126.09	8874.6	0.5637	319.82
A10	6678.1	161.21	7649.4	0.7600	836.42
A11	3254.4	155.09	4129.2	0.6625	181.22
A12	3468.9	167.16	4530.3	0.4602	196.23

Table 6.4: Performance metrics of LSTM models without calendar aging.

6.7 Neural Network using all parameters results:

In this section, we present the results of the Neural networks using all parameters TNO data set is used to evaluate the performance of the NN for RUL prediction. Except for battery *A9*, all the other batteries are used for training and testing.

In Table 6.7, given $k = 1$, the weighting scheme is inverse distance weights and the available range of SOH values is between 100% to 95%, i.e., if very less samples are available in the degradation history of the test data, the MAPE is higher and it reduces when the battery is nearing its EOL. The MAPE is 21.9% at 95% SOH and it reduces to 15.7% at 85% SOH. This can be observed for other values of k. The MAPE is the lowest at 85% SOH.

We can see that when $k = 1$, the values of MAPE in both tables 6.7 and 6.8, are equal. This is because, when $k = 1$, RUL predicted by using the inverse distance weight scheme and uniform weight scheme is equal. This can be verified from equations 5.15 and 5.16. From Tables 6.5, 6.6, 6.7 and 6.8, we can observe that for $k = 1$ and at 85% value of SOH of the test degradation battery, SBM model has the lowest values of MAE and MAPE. Thus, we can conclude that the optimal SBM model to predict the RUL in the TNO data set is a SBM with $k = 1$, and at 85% value of SOH value of the test degradation history.

From the above tables, we can also observe that the MAPE values are lower when the inverse distance weighting scheme is used as compared to uniform weights. This is because, when an in- verse distance weighting scheme is used, the less similar degradation histories have lesser influence on the predicted RUL as compared to a uniform weighting scheme, where all the k degradation histories have equal influence on the predicted RUL.

Figures 6.5, 6.6 and 6.7 show the boxplots representing the spread of the MAPE for each battery in the data set, when inverse distance weighting scheme is used. Figures 6.8, 6.9 and 6.10 show the boxplots representing the spread of the MAPE for each battery in the data set, when uniform weighting scheme is used. From these figures, it can be observed that, in general, the spread of the MAPE of the SBM with inverse weighting scheme is less than the spread of the MAPE of the SBM with uniform weighting scheme.

From Figures 6.5, 6.6 and 6.7, we can observe that for the batteries *A11* and *A12*, the median value of the MAPE is very less (less than 10%) and the spread of the MAPE is very low. (The horizontal red line present inside the box of a boxplot, indicates the median value). This implies that different values of k of a SBM with inverse distance

weighting scheme, predicts similar RUL for batteries *A11* and *A12*. This is also evident from Figure 5.1, as *A11* and *A12* have a very similar degradation pattern. From Figures 6.5 to 6.7 , we can observe that for the batteries *A4* and *A10*, the median value of MAPE is around 40%. The main reason for this large MAPE is that there are no degradation histories in the data set that are very similar to the degradation histories of *A4* and *A10*.

From Figures 6.5 to 6.7 , we can observe that for the batteries *A1*, *A2* and *A3*, the median value of their MAPE is less than 15%. This is a reasonably good value of MAPE and since the degradation histories of batteries *A1*, *A2* and *A3* have degradation histories that are similar to them (Refer Figure 5.1).

One of the main takeaway from the boxplots shown in Figures 6.5 to 6.10 is that the *SBM with inverse distance weights* is better than *SBM with uniform weights*, as in the former, the extreme degradation histories has less influence on the predicted RUL and in the latter, all the *k* similar degradation histories have an equal influence on the predicted RUL.

SBM is an example of instance-based learning algorithm (IBL), where the model generates pre- dictions by comparing the test samples with train samples that are already known. Thus, when more samples are available, then the SBM model can generate better predictions. The MAPE of the predicted RUL can be reduced if more degradation histories are available in the RTF data.

The best model out of the SVR, LSTM and SBM model is the *SBM model with inverse distance weights* with *k*=1 and the MAPE of the model is 15.70% The optimal parameters *k* might change when the RUL is predicted with additional RTF data.

Range of SOH values considered	k=1	k=2	k=3	k=4	k=5	k=6
100% - 85%	2906.2	3560.6	3630.4	3520.6	3449.8	3387.3
100% - 90%	10643	11282	11301	11518	11303	10931
100% - 95%	17904	18367	19090	17218	17168	16930

Table 6.5: Mean Absolute Error (MAE) values of the similarity-based model (SBM) with inverse distance weights for k=1 to k=6

Range of SOH values considered	k=1	k=2	k=3	k=4	k=5	k=6
100% - 85%	2906.2	4409.7	5270.0	5635.1	5085.6	4942.4
100% - 90%	10643	13592	16383	17520	15954	15720
100% - 95%	17904	21220	25919	24332	23838	26768

Table 6.6: Mean Absolute Error (MAE) values of the SBM with uniform weights for k=1 to k=6

Range of SOH values considered	k=1	k=2	k=3	k=4	k=5	k=6
100% - 85%	15.70643	18.98143	19.36586	18.99714	19.13771	18.88543
100% - 90%	20.40329	20.62886	20.387	21.50329	21.51429	20.91229
100% - 95%	21.98886	20.94329	21.566	19.23043	19.27286	19.04943

Table 6.7: Mean Absolute Percentage Error (MAPE) values of the SBM with inverse distance weights for k=1 to k=6

Range of SOH values considered	k=1	k=2	k=3	k=4	k=5	k=6
100% - 85%	15.70643	26.90471	34.587	38.36429	36.49271	38.37414
100% - 90%	20.40329	28.79286	38.55157	43.136	41.01271	46.01614
100% - 95%	21.98886	25.99314	34.49671	34.49929	36.04871	45.04986

Table 6.8: Mean Absolute Percentage Error (MAPE) values of the SBM with uniform weights for k=1 to k=6

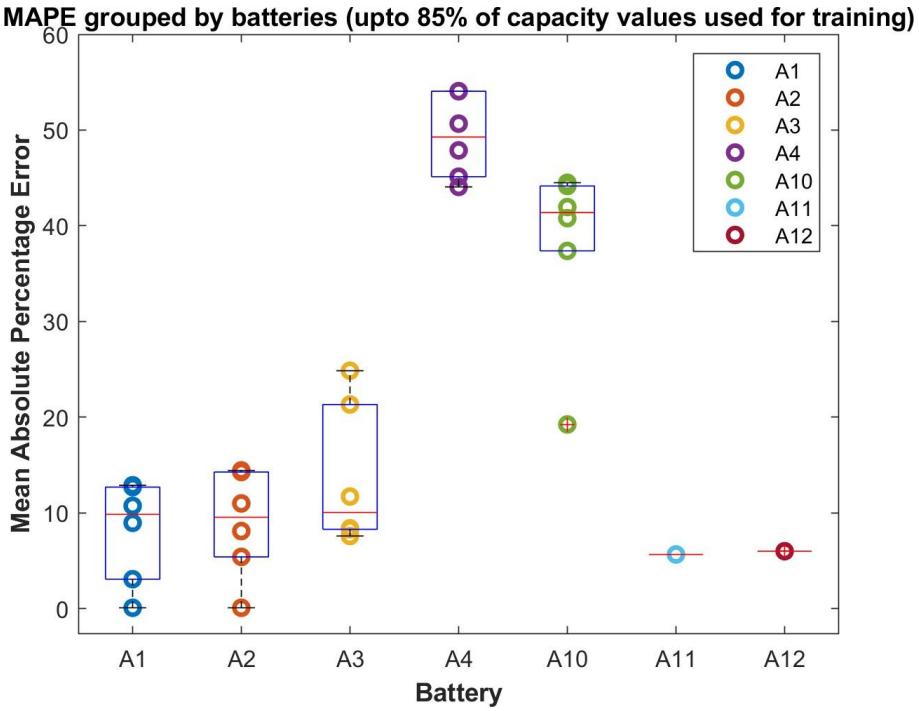


Figure 6.5: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using inverse distance weights (100% - 85% of capacity values used for training)

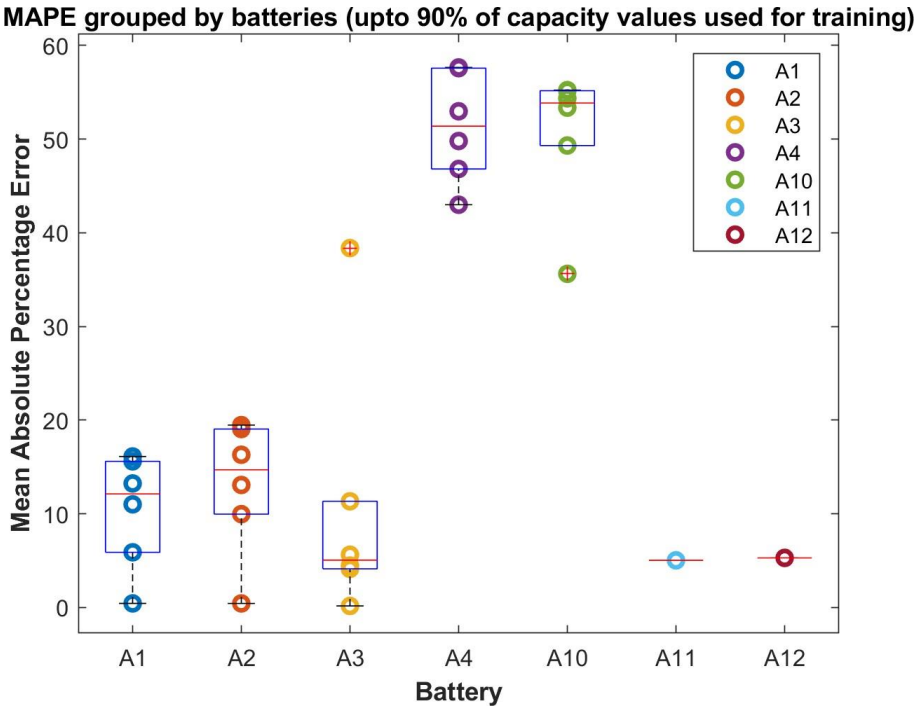


Figure 6.6: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using inverse distance weights (100% - 90% of capacity values used for training)

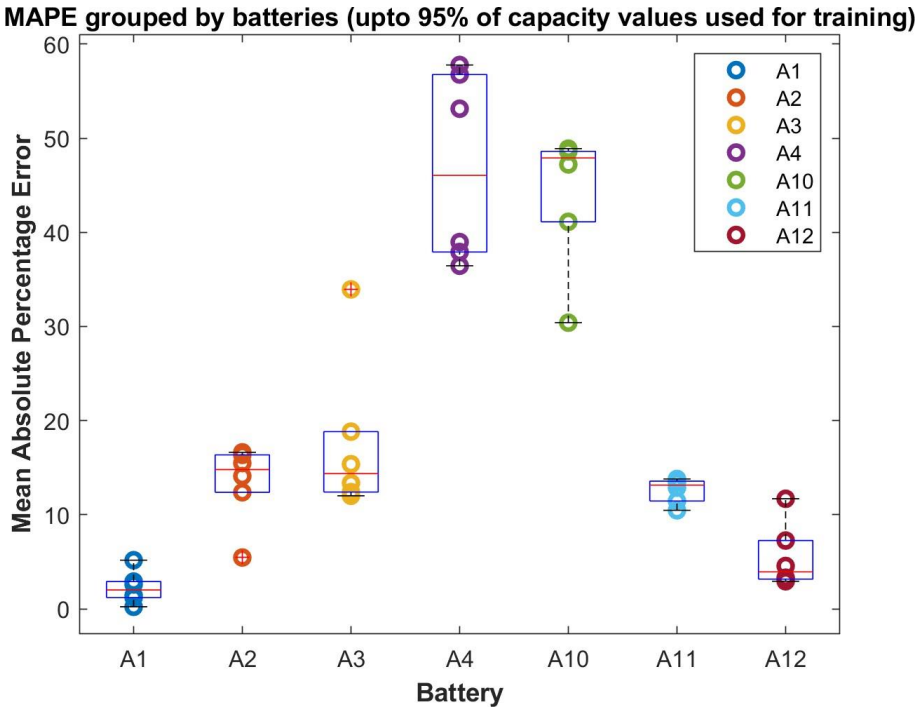


Figure 6.7: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using inverse distance weights (100% - 95% of capacity values used for training)

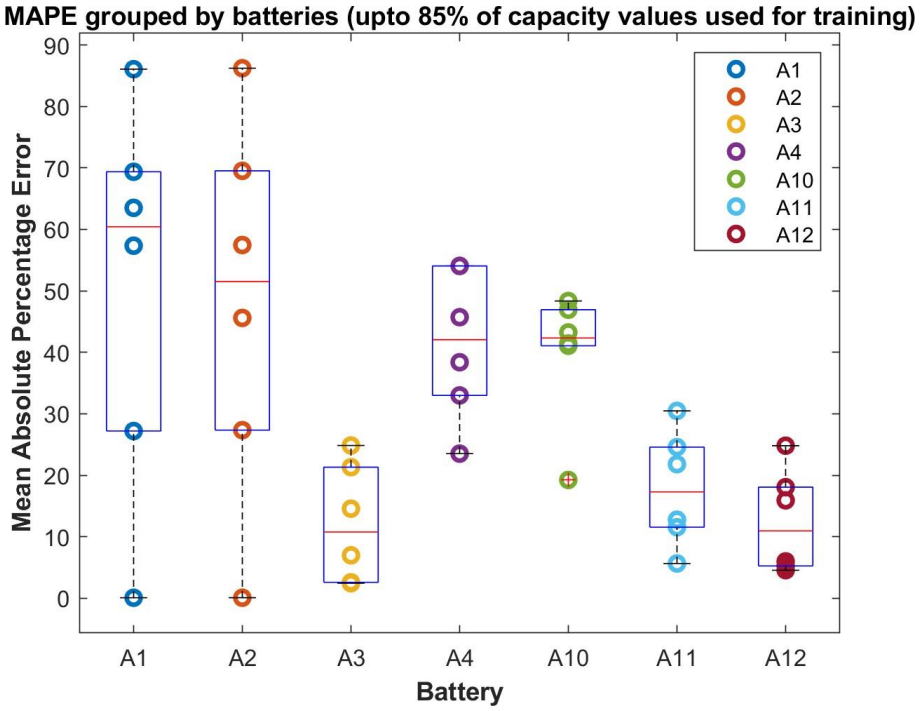


Figure 6.6: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using inverse distance weights (100% - 90% of capacity values used for training)

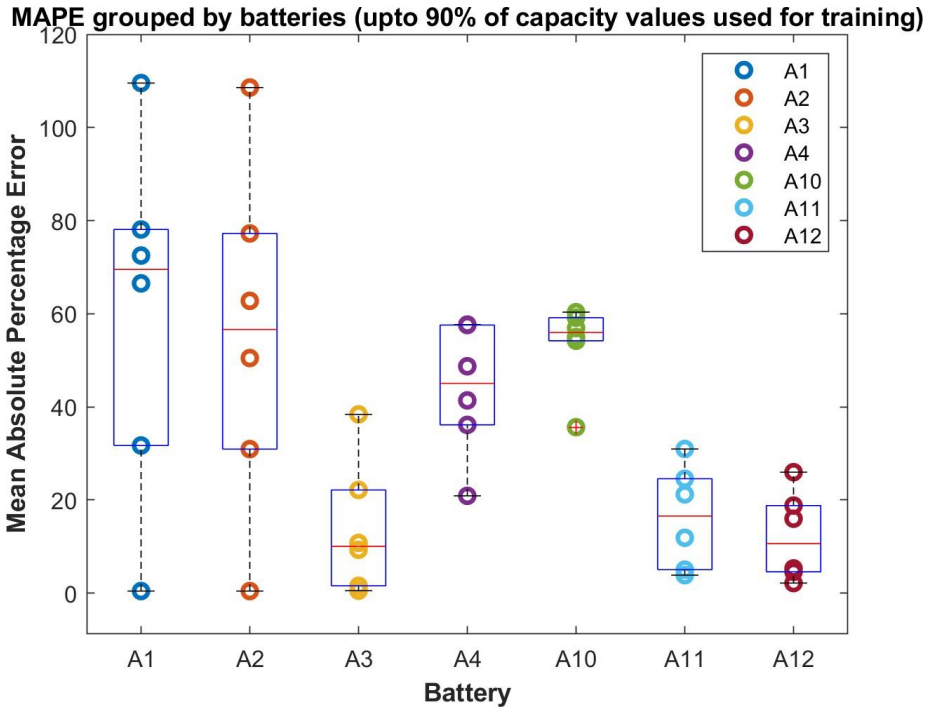


Figure 6.9: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using uniform weights (100% - 90% of capacity values used for training)

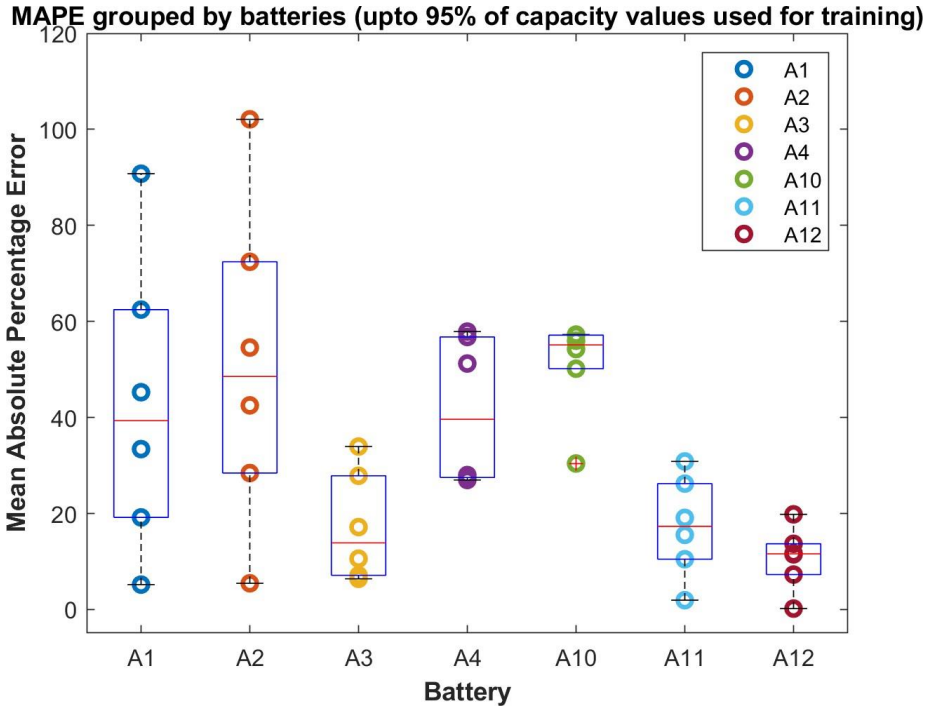


Figure 6.10: Boxplot showing the MAPE grouped by batteries for k=1 to 6. Target is computed using uniform weights (100% - 95% of capacity values used for training)

6.8 Review of research questions

In the problem formulation section of Chapter 1, we posed our main research question and other possible research questions that can be addressed as part of this research. In this section, we present the research findings for each of these questions.

- *How to predict the RUL of a lithium-ion battery onboard an EV in a real-time scenario?*

We implemented SVR models and LSTM models to predict the RUL of a LIB, given the degradation history of the battery. We split the degradation history of each battery in the data set into two subsets - the training subset and the testing subset. We trained a separate SVR model and LSTM model for each training subset and we also generated predictions from the models using the corresponding testing subsets.
We also implemented SBMs to predict the RUL of a LIB. SBMs are instance-based learning (IBL) models and they predict the RUL of a test battery by considering the degradation histories of similar batteries. This is unlike SVRs and LSTMs where we train using the training subset of degradation history data of a single battery.

- *Will the prediction model give better results when the impact of resting (calendar aging) on battery degradation is considered?*

We trained SVR models and LSTM models for RUL prediction. We trained the models without considering the effect of resting (calendar aging) and also with the effects of resting (calendar aging). We found that for some batteries, including the resting period of the battery (which is an indicative of calendar aging) as one of the input features, reduced the error. However, for some batteries, including the resting period increased the error.

- *Can the prediction model implemented return the remaining useful life of the battery in terms remaining range for which the EV can be driven?*

Yes, the implemented models predict the RUL of the LIBs in terms of remaining driving range (RDR).

- *What are the different model performance criteria to compare different RUL prediction models implemented in this study?*

We compared the different RUL prediction models based on the model evaluation metrics such as MAE, MAPE, RMSE, R^2 and model running time.

- *Can the prediction model predict the remaining driving range (RDR) of electric vehicles that are subjected to a wide range of driving profiles cycles?*

The implemented RUL prediction model should be able to predict the RDR of an electric vehicle that is subjected to different driving profiles, given that the degradation history of the LIB used is preprocessed as suggested in this research. However, this was not tested and verified as part of this research as the data of LIBs subjected to diverse driving profiles were not present in the data set.

Chapter 7

Conclusions

7.1 Concluding Summary

In this section, we present the concluding summary of this research.

7.1.1 Preliminaries

In this chapter, we discussed about the basic working principle and advantages of a lithium-ion battery. We defined the necessary terminology regarding LIBs. We also discussed briefly about the various battery form factors, battery management systems and battery aging.

7.1.2 Literature Review

We reviewed some of the commonly used publicly available lithium-ion battery degradation data sets. We reviewed several research papers regarding RUL useful prediction of LIBs. We concluded that data-driven methodology based such as SVR, LSTM and SBM are suitable for our requirement. We also identified the research gap and we addressed the same.

7.1.3 Feature Engineering and EDA

In this chapter, we preprocessed the data by removing unwanted columns, created new features from the existing features in the TNO data set.

7.1.4 Machine Learning models

In this chapter, we implemented machine learning models such as SVR, LSTM and SBM. We optimized the hyperparameters of the SVR using Bayesian optimization technique. We implemented an LSTM network to predict the RUL of LIBs. We developed SVR and LSTM models for each battery in the data set, and we tested the effect of calendar aging on a battery's RUL. Finally, we implemented a SBM that uses dynamic time warping as a distance metric to compute the RUL of LIBs.

7.2 Results

In this chapter, we discussed the results of the three models that we implemented as part of this, graph was plotted by training the NN by giving all the input parameters i.e., cycle number, voltage, current, temperature and time.

The capacity prediction gives a decent accuracy. The accuracy of predicted capacity was measured using R square(R2) and root mean square error (RMSE) functions.

The capacity prediction gives a decent accuracy. As per the methodology implemented in the above-mentioned research paper, capacity was predicted only using cycle number.

The accuracy of predicted capacity was measured using R square(R2) and root mean square error (RMSE) functions.

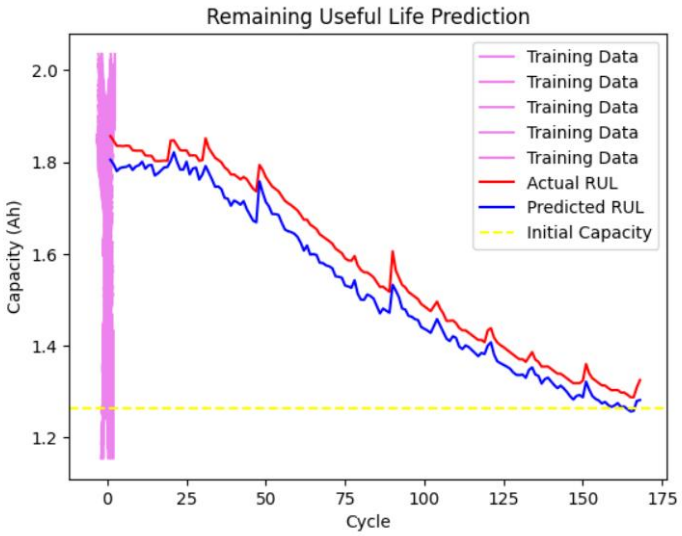


Fig 7.1: Prediction Graph of Battery showing initial capacity

This graph will plot the accuracy over epochs. Each point on the graph represents the accuracy achieved by the model on the validation set at the end of each epoch. Fig shows the battery has reached the defined threshold value, since it reached the threshold value, we need to replace the battery. If the battery didn't reach threshold value, then it shows to use the battery.

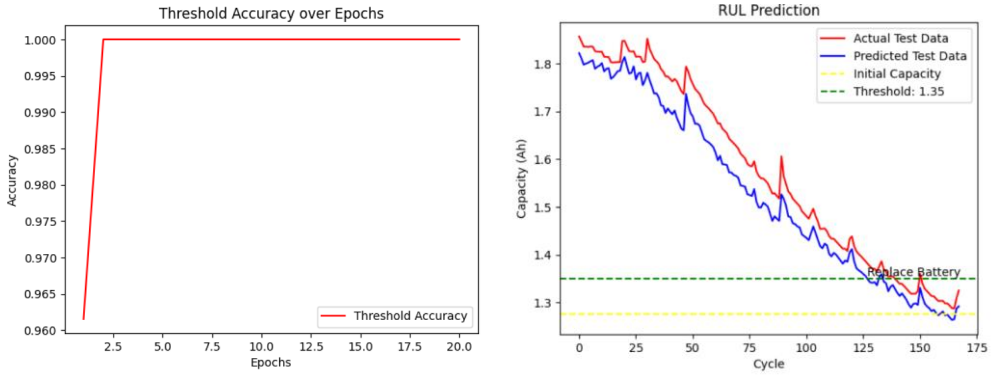


Fig: 7.2 (a) Epochs vs Accuracy (b) Cycles vs Capacity

Mean Absolute Error (MAE) and Mean Squared Error (MSE) are more commonly used for regression tasks.

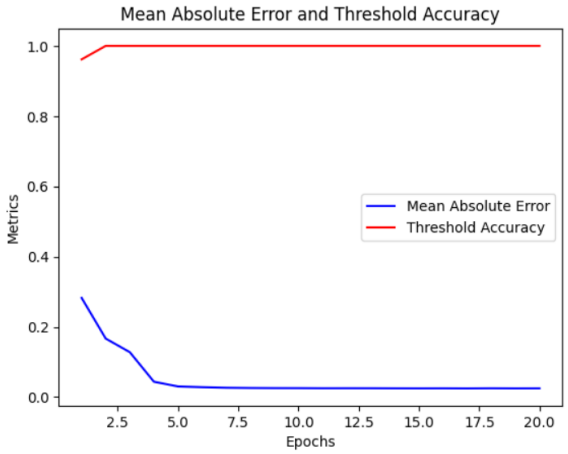


Fig: 7.3 Epochs vs Metrics

If Threshold value is greater than set threshold then the graph is given below:

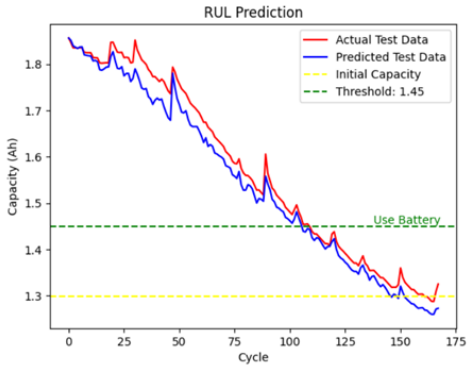


Fig 7.4 Cycles vs Capacity

7.3 Conclusions

The graph was plotted by training the NN by giving all the input parameters i.e cycle number, voltage, current, temperature and time.

The capacity prediction gives a decent accuracy. The accuracy of predicted capacity was measured using R square(R2) and root mean square error (RMSE) functions.

The capacity prediction gives a decent accuracy. As per the methodology implemented in the above-mentioned research paper, capacity was predicted only using cycle number.

The accuracy of predicted capacity was measured using R square(R2) and root mean square error (RMSE) functions.

7.4 Limitations and Future Work

The code implementing an Artificial Neural Network (ANN) for predicting the remaining useful life (RUL) of lithium-ion batteries exhibits several limitations and suggests avenues for future improvement. Firstly, the risk of overfitting persists due to the dataset's small size and the potentially excessive number of epochs used during training. Additionally, reliance on basic evaluation metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE) could lead to an incomplete assessment of the model's performance. Moreover, the scalability of the model may be limited, necessitating exploration into more sophisticated architectures or larger datasets. Hyperparameter selection remains somewhat arbitrary, and the absence of cross-validation inhibits a robust assessment of the model's generalization capacity. Future work could involve implementing cross-validation techniques and hyperparameter optimization to improve model stability and performance. Advanced architectures such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) could be explored for better feature extraction and temporal dependency modeling. Ensemble methods, regularization techniques, and more comprehensive feature engineering may also enhance model accuracy and reliability. Incorporating domain knowledge and real-time prediction capabilities could further augment the model's applicability in practical scenarios. Overall, addressing these limitations and pursuing suggested future avenues can significantly elevate the ANN's effectiveness in predicting the RUL of lithium-ion batteries. Additionally, developing the model to run in real-time on streaming data and accurately predict RUL while showcasing the prediction accuracy percentage in real-time would significantly enhance its utility and effectiveness.

Bibliography

1. Adib, K., C. Angela and W. Lim (2020). "SOH and RUL Prediction of Lithium-Ion Batteries Based on LSTM with Ensemble Health Indicators". In: (page 18).
2. Agnihotri, Apoorv and Nipun Batra (2020). "Exploring Bayesian Optimization". In: *Distill*. DOI: 10.23915/distill.00026. URL: https://distill.pub/2020/bayesian- optimization (page 37).
3. Ahmadzadeh, F and J Lundberg (2014). "Remaining useful life estimation: review". In: *Interna- tional Journal of System Assurance Engineering and Management* 5, pp. 461–474 (page 17).
4. Barr´e, Anthony et al. (2013). "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications". In: *Journal of Power Sources* 241, pp. 680–689 (page 8).
5. Baru, Aditya (2018). *Three Ways to Estimate Remaining Useful Life for Predictive Maintenance*.
6. Birkl, Christoph (2017). "Diagnosis and prognosis of degradation in lithium-ion batteries". PhD thesis. University of Oxford. URL: https : / / doi . org / 10 . 5287 / bodleian : KO2kdmYGg (page 14).
7. Chen, Zewang et al. (2021). "Lithium-ion batteries remaining useful life prediction based on BLS- RVM". In: *Energy* 234, p. 121269. ISSN: 0360-5442.
8. Cortes, Corinna and V. Vapnik (2004). "Support-Vector Networks". In: *Machine Learning* 20, pp. 273–297 (page 35).
9. Eker, Ö mer Faruk, Faith Camci and Ian K. Jennions (2014). "A Similarity-Based Prognostics Approach for Remaining Useful Life Prediction". In: (page 19).
10. Gelfusa, M. et al. (2015). "Advanced signal processing based on support vector regression for LIDAR applications". In: *Image and Signal Processing for Remote Sensing XXI*. Ed. by Lorenzo Bruzzone. Vol. 9643. International Society for Optics and Photonics. SPIE, pp. 135–145. DOI: 10.1117/12.2194501. URL: https://doi.org/10.1117/12.2194501 (page 36).
11. Herh, Michael (July 2021). *Hyundai Motor's electric truck porter EV catches fire while running*.
12. Hochreiter, S. and J. Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation*
13. Hong, Joonki et al. (2020). "Towards the swift prediction of the remaining useful life of lithium- ion batteries with end-to-end deep learning". In: *Applied Energy* 278, p. 115646. ISSN: 0306- 2619. DOI: https:// doi. org/ 10 .1016 / j. apenergy. 2020 .115646.

14. Jia, J. et al. (2020). "SOH and RUL Prediction of Lithium-Ion Batteries Based on Gaussian Process Regression with Indirect Health Indicators". In: *Energies* 13, p. 375 (page 20).
15. Jorge, I. et al. (2020). "New ANN results on a major benchmark for the prediction of RUL of Lithium-Ion batteries in electric vehicles". In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1246–1253 (page 18).
16. Khumprom, Phattara and Nita Yodo (2019). "A Data-Driven Predictive Prognostic Model for Lithium-ion Batteries based on a Deep Learning Algorithm". In: *Energies* 12, p. 660 (page 18).
17. Kim, Seokgoo, Nam H. Kim and Joo-Ho Choi (2020). "Prediction of remaining useful life by data augmentation technique based on dynamic time warping". In: *Mechanical Systems and Signal Processing* 136, p. 106486 (page 19).
18. Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (page 40).
19. Lea, Colin S. et al. (2016). "Temporal Convolutional Networks: A Unified Approach to Action Segmentation". In: *ArXiv* abs/1608.08242 (page 18).
20. Li, Lianbing et al. (2018). "Indirect remaining useful life prognostics for lithium-ion batteries". In:*2018 24th International Conference on Automation and Computing (ICAC)*, pp. 1–5 (page 20).
21. Li, X. et al. (2017). "An On-Board Remaining Useful Life Estimation Algorithm for Lithium-Ion Batteries of Electric Vehicles". In: *Energies* 10, p. 691 (page 19).
22. Liu, J. and Z. Chen (2019). "Remaining Useful Life Prediction of Lithium-Ion Batteries Based on Health Indicator and Gaussian Process Regression Model". In: *IEEE Access* 7, pp. 39474– 39484 (page 20).
23. Moral, P. (1997). "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus De L' Academie Des Sciences Serie I-mathematique* 325, pp. 653–658 (page 20).
24. Nuhic, Adnan et al. (2013). "Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods". In: *Journal of Power Sources* 239, pp. 680–688. ISSN: 0378-7753.
25. Omariba, Zachary Bosire, Lijun Zhang and Dongbai Sun (2018). "Review on Health Management System for Lithium-Ion Batteries of Electric Vehicles". In: *Electronics* 7, p. 72 (pages 7, 32).
26. Pecht, Michael (n.d.). *CALCE battery degradation data set*. URL: https://calce.umd .edu/ battery-data (pages 14, 21, 22).
27. P´erez, Aramis et al. (2018). "Characterizing the Degradation Process of Lithium-ion Batteries Using a Similarity-Based-Modeling Approach". In: (page 19).
28. Plett, Gregory L. (2004). "High-performance battery-pack power estimation using a dynamic cell model". In: *IEEE Transactions on Vehicular Technology* 53, pp. 1586–1593 (page 7).

29. Reis, G. D. et al. (2021). “Lithium-ion battery data and where to find it”. In: (page 13). “Remaining useful life prediction of lithium-ion batteries based on Monte Carlo Dropout and gated
30. recurrent unit” (2021). In: *Energy Reports* 7, pp. 2862–2871.
31. Saha, B. and K. Goebel (2007). “NASA Ames Research Center, Battery Data Set, NASA Ames Prognostics Data Repository”. In: URL: http://ti.arc.nasa.gov/project/prognostic- data-repository (pages 13, 20, 22).
32. Sakoe, Hiroaki and Seibi Chiba (1978). “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, pp. 159–165 (page 43).
33. Severson, K. A. (2019). “Data-driven prediction of battery cycle life before capacity degradation”. In: *Nature Energy* 4, pp. 383–391 (pages 16, 18).
34. Shen, Dongxu et al. (2021). “A novel online method for predicting the remaining useful life of lithium-ion batteries considering random variable discharge current”. In: *Energy*
35. Smola, Alex and B. Schölkopf (2004). “A tutorial on Support Vector Regression”. In: *Statistics and Computing* 14, pp. 199–222 (page 35).
36. Smuts, M., B. Scholtz and J. Wesson (2017). “A critical review of factors influencing the remaining driving range of electric vehicles”. In: *2017 1st International Conference on Next Generation Computing Applications (NextComp)*, pp. 196–201 (page 64).
37. Snoek, Jasper, H. Larochelle and Ryan P. Adams (2012). “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *NIPS* (page 37).
38. Soons, Youri et al. (2020). “Predicting Remaining Useful Life with Similarity-Based Priors”. In: *IDA* (page 19).
39. Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfit- ting”. In: *J. Mach. Learn. Res.* 15, pp. 1929–1958 (page 40).
40. Sun, Peiyi et al. (Jan. 2020). “A Review of Battery Fires in Electric Vehicles”. In: *Fire Technology*, pp. 1–50. DOI: 10.1007/s10694-019-00944-3 (page 1).
41. Sun, Tianfei et al. (2019). “A Novel Hybrid Prognostic Approach for Remaining Useful Life Es- timation of Lithium-Ion Batteries”. In: *Energies* 12, p. 3678 (page 18).Team, MIT EV (Dec. 2008). *A Guide to Understanding Battery Specifications*.
42. Vashisht, Raghav (Sept. 2021). *Machine Learning: When to perform a Feature Scaling?* URL:https://www.atoti.io/when-to-perform-a-feature-scaling/ (page 34).Virkler, Dennis A., B. M. Hillberry and Prem K. Goel (1979). “The Statistical Nature of Fatigue Crack Propagation”. In: *Journal of Engineering Materials and Technology-transactions of The Asme* 101, pp. 148–153 (page 19).
43. Wang, Dong et al. (2017). “Nonlinear-drifted Brownian motion with multiple hidden states for remaining useful life prediction of rechargeable batteries”. In: *Mechanical Systems and Sig- nal Processing* 93, pp. 531–544. ISSN: 0888-3270. DOI: https : / / doi . org / 10 . 1016 / j . ymssp. 2017 . 02 . 027. URL:

https://www.sciencedirect.com/science/article/pii/ S0888327017301012 (page 21).

44. Wang, Shunli et al. (2021). "A critical review of improved deep learning methods for the remaining useful life prediction of lithium-ion batteries". In: *Energy Reports* 7, pp. 5562–5574. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2021.08.182. URL: https://www. sciencedirect.com/science/article/pii/S235248472100785X (page 20).
45. Wei, Jingwen, Guangzhong Dong and Zonghai Chen (2018). "Remaining Useful Life Prediction and State of Health Diagnosis for Lithium-Ion Batteries Using Particle Filter and Support Vector Regression". In: *IEEE Transactions on Industrial Electronics* 65, pp. 5634–5643 (page 20).
46. Wu, Y. et al. (2019). "Remaining Useful Life Prediction of Lithium-Ion Batteries Using Neural Network and Bat-Based Particle Filter". In: *IEEE Access* 7, pp. 54843–54854 (page 20).
47. Xiang, Shun et al. (2018). "Lithium-Ion Battery Online Rapid State-of-Power Estimation under Multiple Constraints". In: *Energies* 11, p. 283 (page 7).
48. Xiong, Rui (2019). "Battery Management Algorithm for Electric Vehicles". In: (pages 3, 4, 8). Xu, Lei et al. (2019). "Modeling Tabular data using Conditional GAN". In: *Advances in Neural Information Processing Systems* (page 64).
49. Xu, Xiaodong et al. (2021). "Remaining Useful Life Prediction of Lithium-ion Batteries Based on Wiener Process Under Time-Varying Temperature Condition". In: *Reliability Engineering & System Safety* 214, p. 107675. ISSN: 0951-8320. DOI: https:// doi. org / 10 . 1016 / j . ress . 2021 . 107675. URL: https : / / www . sciencedirect . com / science / article / pii / S0951832021002131 (page 21).
50. Zhang, Yongzhi et al. (2019). "Validation and verification of a hybrid method for remaining useful life prediction of lithium-ion batteries". In: *Journal of Cleaner Production* 212, pp. 240–249 (page 20).
51. Zhou, Danhua et al. (2020). "State of Health Monitoring and Remaining Useful Life Prediction of Lithium-Ion Batteries Based on Temporal Convolutional Network". In: *IEEE Access* 8, pp. 53307–53320 (page 18).
52. Zhou, Yapeng and Miaohua Huang (2016). "Lithium-ion batteries remaining useful life prediction based on a mixture of empirical mode decomposition and ARIMA model". In: *Microelectron. Reliab.* 65, pp. 265–273 (page 18).

Appendix A

System Configuration

A.1 Hardware specifications

- **Processor:** Intel(R) Core (TM) i9-9980HK CPU
- **Operating System:** Windows 10 Home
- **Clock Speed:** 2.40GHz
- **Number of Cores:** 8
- **Total RAM:** 16 GB

A.2 Software Used

- **MATLAB R2021a** - Student Edition
- **OPERATING SYSTEM** – Windows 10 or above, Linux
- **TECHNOLOGY** – Python
- **IDE** – Jupyter Notebook
- **PYTHON VERSION** – Python 3.9 or above

Appendix B

Battery Specifications

B.1 NASA data set

Parameters	Values
Nominal Voltage	3.6V
Nominal Capacity	2,850 mAh
Minimum Discharge Voltage 3V	3V
Maximum Discharge current	1C
Charging Voltage	4.2V (maximum)
Charging current	0.5C
Charging Time	3 hours (approx.)
Charging Method	CC and CV
Cell Weight	48g (approx.)
Cell Dimension	18.4mm (dia) and 65mm (height)

B.2 CALCE data set

Parameters	Values
Cell Dimensions(mm):	25.85mm(D)*65.15mm(H)
Cell Weight(g):	70
Nominal Capacity:	2300mAh
Voltage (nominal, V):	3.3V
Internal impedance (1kHz AC typical, mΩ):	6
HPPC 10 Sec Discharge Pulse Power 50% SOC:	200W
Recommended standard charge method:	1C to 3.6V CCCV, 45 min
Recommended fast charge Method to 80% SOC:	4C to 3.6V CC,12 min
Maximum continuous discharge(A):	70
Maximum Pulse discharge(10 seconds,A):	120
Cycle life at 10C discharge, 100% DOD:	Over 1,000 cycles
Operating temperature range:	-30°C to +55°C
Storage temperature range: -40°C to +60°C	-40°C to +60°C

B.3 Oxford data set

Parameters	Values
Typical Capacity	740 mAh
Nominal Voltage	3.7 V
Max. Current (Charge Condition)	1.48 A
Max. Voltage (Charge Condition)	4.2 V ± 0.03 V
Continuous Current (Discharge Condition)	14.8 A
Peak Current (Discharge Condition)	29.60 A
Cut-off Voltage (Discharge Condition)	4.2 V ± 0.03 V
Cycle Life	> 500 cycles
Operating Temperature (Charge)	0 to 40^{o}C
Operating Temperature (Discharge)	-20 to 60^{o}C

Table B.1: Battery specifications of the battery used in Oxford data set

B.4 Toyota Research Institute data set

Parameters	Values
Typical Capacity	1200 mAh
Nominal Voltage	3.3 V
Max. Current (Charge Condition)	2.0 A
Charge Time	2.0hrs
Max. Voltage (Charge Condition)	4.2 V ± 0.05 V
Max. Continuous Current (Charge Condition)	4 A
Max. Continuous Current (Discharge Condition)	30 A
Min. Voltage (Discharge Condition)	2V
Cut-off Voltage (Discharge Condition)	2.5 V
Temperature (Charge)	0 to 55˚C
Temperature (Discharge)	-30 to 55˚C
Temperature (Storage)	-40˚C to 60˚C

Table B.2: Battery specifications of the battery used in TNO data set

B.5 TNO data set

Parameters	Values
Typical Capacity	2000 mAh
Nominal Voltage	3.6 V
Max. Current (Charge Condition)	2.0 A
Charge Time	2.0 hrs
Max. Voltage (Charge Condition)	4.2 V ± 0.05 V
Continuous Current (Discharge Condition)	14.8 A
Peak Current (Discharge Condition)	6.0 A at 45^{o}C, 4.0 A at 60^{o}C
Cut-off Voltage (Discharge Condition)	2.5 V
Temperature (Charge)	0 to 45^{o}C
Temperature (Discharge)	-20 to 60^{o}C
Temperature (Storage)	< 35^{o}C

Table B.3: Battery specifications of the battery used in TNO data set

Appendix C

C. TECHNOLOGY USED

C.1 PYTHON

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum in the late 1980s, Python has gained immense popularity and has become one of the most widely used programming languages in the world. It has a vast ecosystem of libraries and frameworks that make it suitable for various applications, ranging from web development to data analysis and artificial intelligence.

Python's versatility and cross-platform compatibility are other notable characteristics. It supports multiple operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of users. Furthermore, Python integrates seamlessly with other programming languages, enabling developers to combine Python code with modules written in C, C++, or Java, among others.

The Python Standard Library provides a rich set of modules and functions that cover a wide range of tasks, from file manipulation to network programming and web development. This comprehensive library, combined with the extensive third-party packages available through the Python Package Index (PyPI), makes Python a powerful language for rapid application development.

Python's popularity in the field of data science and machine learning is particularly noteworthy. Libraries such as NumPy, Pandas, and Matplotlib provide efficient and convenient tools for data manipulation, analysis, and visualization. Additionally, frameworks like TensorFlow and PyTorch offer powerful capabilities for building and training machine learning models.

Web development is another domain where Python shines. Frameworks like Django and Flask provide developers with the necessary tools to build robust and scalable web applications. These frameworks offer features such as URL routing, template rendering, database integration, and user authentication, simplifying the development process and boosting productivity.

Python's community and ecosystem play a crucial role in its success. The Python community is known for its inclusivity and helpfulness, providing support through online forums, mailing lists, and conferences. The open-source nature of Python encourages collaboration and the sharing of code, resulting in a vast collection of libraries and resources that benefit developers worldwide.

Python's popularity and demand in the job market have been steadily increasing over the years. Many companies and organizations, including Google, Facebook, and NASA, rely on Python for their critical projects. Its ease of use, combined with its ability to handle complex tasks, makes it an excellent choice for both beginners and experienced programmers.

C.2 TENSORFLOW

TensorFlow is an open-source machine learning framework developed by Google. It has gained significant popularity and has become one of the most widely used frameworks for building and deploying machine learning models. TensorFlow provides a comprehensive set of tools, libraries, and resources that enable developers to create sophisticated artificial intelligence applications.

At its core, TensorFlow is based on the concept of computational graphs. In TensorFlow, you define a computational graph that represents the flow of data and operations in a model. Nodes in the graph represent mathematical operations, while edges represent tensors, which are multi-dimensional arrays or data structures. This graph-based approach allows for efficient execution and optimization of complex computations.

TensorFlow's distributed computing capabilities are also notable. It enables training and inference on multiple machines or devices, allowing for scalable and parallel processing. This feature is particularly valuable when dealing with large datasets or complex models that require significant computational power.

Another significant advantage of TensorFlow is its extensive ecosystem. It provides a vast collection of pre-built models and components through TensorFlow Hub, making it easier for developers to leverage existing solutions and accelerate their development process. Additionally, TensorFlow offers integration with other popular libraries and tools, such as NumPy and scikit-learn, further expanding its capabilities and compatibility.

The TensorFlow Extended (TFX) ecosystem is designed specifically for end-to-end machine learning workflows. TFX provides tools for data pre-processing, model training, evaluation, and serving. It enables the deployment of machine learning pipelines at scale, making it suitable for production-level applications.

TensorFlow is not limited to traditional machine learning. It also offers support for deep learning, a subfield of machine learning that focuses on neural networks with multiple layers. Deep learning has achieved remarkable success in areas such as computer vision, natural language processing, and speech recognition. TensorFlow's deep learning capabilities, coupled with its computational efficiency, make it a popular choice for building and training deep neural networks.

The TensorFlow community is vibrant and active. It has a large and growing user base that contributes to the development and improvement of the framework. The community provides extensive documentation, tutorials, and resources, making it easier for beginners to get started with TensorFlow and for experienced developers to explore advanced techniques.

C.3 PYTORCH

PyTorch is an open-source machine learning framework developed by Facebook's AI Research (FAIR) team. It has gained significant popularity and has become a widely adopted framework for building and training deep learning models. PyTorch offers a dynamic computational graph and a user-friendly interface, making it suitable for both beginners and experienced researchers in the field of artificial intelligence.

One of the key features of PyTorch is its dynamic computational graph. Unlike some other frameworks that use a static computational graph, PyTorch allows for dynamic graph construction during runtime. This flexibility enables easier debugging, experimentation, and model prototyping. Developers can define, modify, and execute the graph on-the-fly, which is especially beneficial in scenarios where the network architecture or input sizes may vary.

PyTorch provides a user-friendly and intuitive interface that simplifies the process of building neural networks. Its design philosophy emphasizes Pythonic syntax, which makes code concise and easy to read. The framework offers a high-level library called torch.nn, which provides pre-defined layers, loss functions, and utilities for building complex neural network architectures. Additionally, PyTorch seamlessly integrates with NumPy, a popular numerical computing library, allowing for efficient data manipulation and transformation.

PyTorch's deep learning capabilities are particularly robust. It provides a wide range of builtin functions and modules for implementing various neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers. PyTorch's deep learning ecosystem also includes popular libraries like torchvision, which provides pre-trained models and datasets for computer vision tasks.

PyTorch offers support for training models on multiple devices, including CPUs and GPUs. Its integration with CUDA, a parallel computing platform, enables efficient utilization of GPU resources, leading to faster training and inference times. This feature is especially crucial for training deep learning models, as they often require significant computational power.

The PyTorch community is vibrant and active, contributing to its development and growth. The community provides extensive documentation, tutorials, and resources, making it easier for users to learn and use the framework effectively. PyTorch also benefits from the strong academic presence in the field of deep learning, with many research papers and models being released using PyTorch.

PyTorch has gained popularity in both academia and industry due to its flexibility, ease of use, and strong research-oriented ecosystem. Many top research institutions, including universities and industry research labs, use PyTorch as their primary framework for cuttingedge deep learning research. In addition, PyTorch is widely adopted by industry practitioners, with companies like Facebook, Tesla, and NVIDIA incorporating PyTorch into their AI workflows.

C.4 PANDAS

Pandas is a powerful open-source data manipulation and analysis library for Python. It provides easy-to-use data structures and data analysis tools that make working with structured data, such as tabular or time series data, efficient and convenient. Developed by Wes McKinney in 2008, Pandas has become a go-to library for data scientists and analysts due to its versatility and extensive functionality.

At the heart of Pandas are two primary data structures: Series and DataFrame. A Series is a one-dimensional labeled array that can hold any data type. It is similar to a column in a spreadsheet or a single column of a SQL table. A DataFrame, on the other hand, is a two-dimensional labeled data structure, resembling a table or a spreadsheet, where each column can contain different data types. DataFrames are especially useful for handling structured data and performing complex data manipulations.

Pandas provides a wide range of functionalities to manipulate and transform data. It allows users to handle missing values, filter data based on conditions, sort and group data, merge and join datasets, and perform arithmetic and statistical operations. These operations can be performed with concise and expressive syntax, making data manipulation tasks more efficient.

One of the key strengths of Pandas is its ability to handle time series data effectively. It offers built-in functionality for time-based indexing, resampling, and time zone handling. This makes it a valuable tool for analyzing and processing data with temporal components, such as stock prices, weather data, or sensor readings.

Pandas integrates seamlessly with other popular Python libraries, such as NumPy, Matplotlib, and SciPy, enabling a comprehensive data analysis workflow. NumPy provides the underlying array processing capabilities, Matplotlib allows for data visualization, and SciPy extends the functionality with advanced statistical operations. Together with Pandas, these libraries form a powerful ecosystem for data analysis and scientific computing.

Pandas also supports reading and writing data from various file formats, including CSV, Excel, SQL databases, and more. This makes it easy to import data into Pandas from different sources, perform analysis, and export the results in a desired format. Additionally, Pandas supports interoperability with other data analysis tools and libraries, allowing for seamless integration into existing workflows.

The Pandas community is vibrant and actively contributes to the library's development and improvement. The community provides extensive documentation, tutorials, and resources, making it easier for users to learn and utilize Pandas effectively. This collaborative environment ensures that the library remains up-to-date, reliable, and continually evolves to meet the needs of its users.

Pandas' versatility and usability have made it a popular choice not only in data science and analytics but also in a wide range of industries. It is widely used in finance, marketing, social

sciences, and many other domains that require data manipulation and analysis. Its ability to handle large datasets efficiently has also made it a preferred tool for big data processing.

C.5 NUMPY

NumPy (Numerical Python) is a powerful open-source library for numerical computing in Python. It provides high-performance multidimensional array objects, along with a wide range of mathematical functions and tools for efficient array manipulation. NumPy serves as a fundamental building block for scientific computing and data analysis in Python.

At the core of NumPy is the ND array (n-dimensional array) object, which is a homogeneous container for storing and manipulating large datasets in a memory-efficient manner. The ND array enables efficient storage and vectorized operations on arrays, allowing for fast numerical computations. Arrays can have any number of dimensions, from one-dimensional vectors to multi-dimensional matrices.

NumPy provides a comprehensive set of functions and methods for creating, manipulating, and reshaping arrays. It supports a variety of data types, including integers, floats, and complex numbers. NumPy arrays can be easily reshaped, sliced, and indexed, allowing for efficient data extraction and manipulation.

One of the key advantages of NumPy is its ability to perform vectorized operations on arrays. Vectorization eliminates the need for explicit loops, resulting in faster and more concise code. NumPy provides a wide range of mathematical functions that can be applied elementwise to arrays, such as arithmetic operations, trigonometric functions, exponential and logarithmic functions, and more. These vectorized operations significantly speed up numerical computations.

NumPy also offers powerful tools for array broadcasting, which allows for efficient operations on arrays with different shapes and dimensions. Broadcasting automatically aligns the dimensions of arrays, eliminating the need for explicit array expansion or replication. This feature simplifies complex array operations and makes code more readable and efficient.

In addition to array manipulation, NumPy provides linear algebra routines, Fourier transform capabilities, random number generation, and tools for working with polynomials and integration. These functionalities make NumPy a versatile library for various scientific computing tasks, including signal processing, image analysis, optimization, and simulation.

NumPy seamlessly integrates with other scientific computing libraries in Python, such as SciPy, pandas, and matplotlib. SciPy builds on top of NumPy and provides additional functionality for scientific computing, including numerical integration, optimization, interpolation, and signal processing. pandas, a popular data manipulation library, relies on NumPy arrays as the underlying data structure for efficient data processing. matplotlib, a powerful visualization library, accepts NumPy arrays for generating high-quality plots and charts.

Resources