LSTM Based Remaining Useful Life Prediction for Lithium-Ion EV Batteries

by

Sapna Pandey

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Lithium-ion batteries are commonly used in electric vehicles (EVs) because of their high energy density, ability to provide good efficiency, and being lightweight. Predicting the Remaining Useful Life (RUL) is critical in lithium-ion batteries as it helps optimize efficiency and timely replacement of these batteries. To optimize the battery performance, it is critical to predict RUL and lithium-ion batteries' End of Life (EOL). There are several approaches for RUL estimation in lithium-ion batteries, such as model-based, data-driven, and hybrid approaches. Out of all the approaches, data-driven approaches, such as, Recurrent Neural Networks (RNN), Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) have gained popularity due to their less complexity and adaptability. In this study, we have investigated the LSTM networks for RUL estimation in lithium-ion batteries.

The first part of the study shows the effect of different parameter changes, such as hidden nodes and window size, in LSTM networks. The investigation reveals that increasing the number of hidden nodes before encountering overfitting improves prediction accuracy and lowers the Root Mean Square Error (RMSE) from 0.2 to 0.03. The experiments to find the ideal window size for the LSTM model used in this study illustrate that the model shows improvement with a higher window size up to a maximum of 14. Moreover, in the second part of this study, we propose an incremental LSTM model for time series forecasting that incorporates the newly available data at each time step and updates itself to make better predictions of future capacity values. The proposed incremental LSTM model improves the RMSE by 17.6% compared to the baseline LSTM model. For further analysis of real-time RUL estimation where there is limited data, and the user wants to predict the RUL at any moment, another LSTM model that considers these assumptions is proposed. The models are trained and tested with the help of two publicly available datasets: The lithium-ion battery aging dataset by NASA Ames Prognostics Center of Excellence (PCoE) and the battery dataset by the Center for Advanced Life Cycle Engineering (CALCE). This research proposes LSTM models that are useful for accurately estimating RUL in lithium-ion batteries, which is critical for Electric Vehicles (EVs).

Acknowledgements

I extend my heartfelt appreciation and thanks to my thesis advisor, Dr. Sagar Naik, whose unwavering guidance, expertise, and steadfast support have been instrumental throughout this research journey.

My deepest gratitude goes to my family for their enduring love, encouragement, and unwavering support. Their guidance and sacrifices have motivated me to succeed. I want to especially thank my sister for motivating me throughout the duration of my thesis and for believing and pushing me to do the best I can.

Finally, I sincerely thank the University of Waterloo faculty members. Your contributions have made a meaningful impact, and I am truly thankful for your involvement.

Dedication

This thesis is dedicated to the infinite possibilities in the advancement of technology. It celebrates the innovative spirit that drives progress and the limitless potential that technology offers to reshape to make this world a better place to live.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The demand for electric vehicles (EVs) and, by extension, EV batteries is rising as the trend toward electric transportation continues to gain attention. Due to the many advantages of using lithium-ion batteries, like their ability to supply high voltage, high energy density, and less self-discharge, they are suitable for various applications. However, it is crucial to guarantee the security and dependability of lithium-ion batteries because any flaw can lead to decreased performance, equipment malfunctions, and potentially dangerous circumstances such as fires or explosions. It is also important to maintain these batteries in a timely manner to ensure they last a long time.

The battery goes through continuous charging and discharging. A battery has completed a cycle when it has gone through a full charge and discharge cycle and has reached its initial state. In simple terms, the battery is fully charged at the start, and then as it is used to power a device or application, it gets discharged and recharged again to its full capacity. The RUL for charge-discharge cycles is the number of full charge/discharge cycles a battery will go through before it reaches its end of life (EOL). The EOL can be described as the moment or the precise number of charge-discharge cycles at which the battery's characteristic properties have reached a critical level needing replacement [48]. The (RUL) spans from the present to the end of life when the battery has used more than 70 to 80 percent of its capacity [36]. To properly assess the condition of batteries and maintain their health, the RUL must be accurately predicted. We need continuous tracking of RUL to ensure that the battery is operating correctly without any anomalous behaviour and that the battery management system (BMS) is working properly. The term system's prognostic and health management (PHM) refers to this evaluation. This technique provides us with all the data that can be used for predicting if the battery health is all right and will alert the users to ensure that the battery is replaced before it reaches its end of life (EOL) [17].

There are several approaches to predict the RUL, such as model-based, data-driven, and hybrid. In the model-based approach, mathematical models are constructed to calculate the RUl. These models are complex, have limited adaptability, and require domain knowledge for construction. On the other hand, data-driven methods rely on past or historical data without requiring specialized domain knowledge or model configuration. The data-driven approach has recently received a lot of attention in the field of lithium-ion battery RUL prediction. These models can forecast the state of charge (SOC), state of health (SOH), and RUL of the batteries using advanced machine-learning techniques, which make them less complex to understand, and they generalize well to different data and applications [52] [26].

Neural networks are one type of machine learning model that can be used to predict such parameters. To predict the RUL of lithium-ion batteries, it is important to estimate the capacity of these batteries over charge-discharge cycles. Models using feed-forward neural networks, convolutional neural networks (CNN), and long short-term memory networks (LSTM)are gaining popularity for such predictions [32]. Long Short-Term Memory (LSTM) neural network is a type of deep learning network commonly used for applications involving precise estimation of battery capacity and Remaining Useful Life (RUL) based on observed battery aging data. The main benefit of LSTM networks is their effective information retention and updating over long periods without encountering the vanishing gradient problem. This study will use the LSTM technique for our analysis and model development [53] [56].

Parameter estimation is crucial in model building and accurately predicting the RUL. The parameters, namely, number of hidden nodes, window size, hidden layers, optimizer, and activation function, can greatly affect the accuracy of LSTM models. In the first part of this study, we investigated the effect of parameter changes in the LSTM model by varying the number of hidden nodes and window size [5].

This study analyzes existing LSTM models and gives a variant for improved time series forecasting. This study presents an incremental LSTM method that dynamically adds new input to the model to improve its forecasting performance in real-time circumstances. In the first stage of our study, a conventional LSTM model is built. Through thorough experimentation and analysis, we seek to determine the best-performing LSTM configuration with the lowest Root Mean Squared Error (RMSE) value. We use this improved LSTM model called LSTM-1 as our starting point for all subsequent comparisons. In the second phase, we provide an incremental LSTM method incorporating fresh data in real-time forecasting called the LTSM-2 model. We also give a variant of LSTM-2, which we call LSTM-3, showcasing a better runtime value than the LSTM-2. Later in this report, we explore the scenario where there is less availability of data because of which we use the

trained model and the predicted value for forecasting the RUL, and we call this model PICE-LSTM, which is LSTM model that uses Predictive Iterative Capacity Estimation.

We look at four different model types:

1. LSTM-1: The LSTM model without retraining is the first model that uses new incoming data to forecast cycle capacity. This model is trained once with the training data, and for prediction on test data, it takes capacity values from the test set to make future predictions of capacity values.

2. LSTM-2: Incremental LSTM model with retraining is the second model, which retrains the LSTM model to estimate capacity by including the new data into its training set. After completing the training, it takes the new capacity values and adds them to its training set for future predictions. The most current data point is used for this retraining, which is done after each cycle.

3. LSTM-3: The incremental LSTM model with retraining updates itself after an interval of cycles. It uses a wait-and-retrain approach, gathering data over the following Z cycles before upgrading the LSTM model, where Z can be noted as an update interval.

4. PICE-LSTM: Predictive Iterative Capacity Estimation LSTM model is trained on some defined train data, which are the capacity values of the battery recorded using charge and discharge cycles. In the inference phase, the model uses predicted values to forecast the RUL and does not use the test or the actual data. The benefit of this model is that when there is less availability of future data, and the user wants to know the RUL at a point in time, this model can give the forecasted RUL. This model adopts a pragmatic approach by using a focused training set and selecting a smaller batch of data preceding the prediction initiation point. This selective utilization of recent data streamlines training, overcoming data volume challenges, and enhances relevance and effectiveness.

We investigate the forecasting performance of the incremental LSTM models with the conventional LSTM. Our findings demonstrate the possibility of adaptive forecasting through an incremental approach in practical settings. This report uses the NASA dataset from the Prognostics Centre of Excellence (PCoE) to conduct our analysis in this paper. This dataset has been extensively used to examine lithium-ion battery aging properties [23]. To check the generalization of the model development methods, the same experiments are repeated with another publicly available dataset, the CALCE dataset. Our verification process involved comparing three crucial performance metrics: RMSE values, runtimes, and predicted RUL values.

Chapter 2

Literature review

This section presents an extensive study of the lithium-ion battery literature. The first half contains information about important battery terminologies such as battery capacity, C-rate, State of Health (SOH), etc. The second half describes the difference between cells, modules, and packs of lithium-ion batteries and offers detailed insights into a lithium-ion cell's internal components.

2.1 Lithium-ion battery terminologies

This section introduces key battery terms, providing a clear understanding of fundamental concepts in lithium-ion technology. Each terminology provides important insights about the battery characteristics, condition, and lifespan, laying the groundwork for predicting their behaviour and performance.

- **Battery capacity:** Battery capacity refers to the overall electric charge a battery can deliver before it is completely discharged, typically measured in ampere-hours (Ah). For instance, a battery with a capacity of 30 Ah can supply a current of 30 A for one hour or 1 A for 30 hours. Temperature and the discharge current affect the initial discharge capacity of a new battery. The initial discharge capacity is the amount of power the battery can deliver before any deterioration in the capacity is visible. The discharge capacity depends on the discharge current inside the battery. The Peukert equation in Equation 2.1 gives the relation between discharge capacity C_d and discharge current I.

$$C_d = K * I^{1-n} \tag{2.1}$$

The equation has empirical constants K and n. The design characteristics of the electrodes impact the coefficient n's value. The coefficient n tends to become closer to a value near 1 in the case of lithium-ion batteries, where the electrodes have incredibly thin plates [31].

- **C-rate:** C-rate refers to a battery's charging or discharging rate in relation to its nominal capacity. It measures how quickly a battery can be charged or discharged. A $'n'$ C-rate battery will discharge in $1/n$ hours. For instance, a fully charged battery with a 1 C-rate ($1C$) will take 1 hour to discharge completely. The units for C-rate are h^{-1}. The capacity can be categorized into two:

  a) Initial Maximum Capacity (C_i): The maximum electrical energy that a new battery can store and is completely charged to power a system is the initial maximum capacity. This capacity is in kilowatt-hours (kWh) for electric vehicles. It is the battery's theoretical maximum capacity. C_i is also known as the rated capacity of a battery.
  b) Current Maximum Capacity (C_c): The maximum electrical energy a battery can currently store and supply, taking into account its age, usage, and deterioration over time, is referred to as its current maximum capacity. The current maximum capacity of a battery indicates the steady reduction in capacity that occurs as a result of aging and repeated cycles of charge and discharge.

- **State of Charge (SOC):** State of Charge (SOC) refers to the proportion of available charge in a battery compared to its maximum capacity. SOC is typically represented as a percentage. The quantity describing how much charge is currently stored in a battery is denoted by Q_a. The total amount of charge that a battery is intended to store and provide under typical operating conditions is Q_r, also known as the rated or maximum charge capacity. The manufacturer specifies a set value for it, and it is frequently written in units like kilowatt-hours (kWh). It is the battery's theoretical maximum capacity. The equation for SOC based on these two terms is provided in equation 2.2:

$$SOC(\%) = \frac{Q_a}{Q_r} \tag{2.2}$$

- **Depth of Discharge (DOD):** Depth of Discharge (DOD) refers to the extent of charge depletion in a battery during a single charge-discharge cycle, measured as the difference between the highest and the lowest State of Charge (SOC) levels. DOD is commonly expressed as a percentage, and the expression is provided in equation 2.3.

$$DOD(\%) = SOC_{high} - SOC_{low} \tag{2.3}$$

- **State of Health (SOH):** State of Health (SOH) is a metric that indicates a battery's current condition or health by comparing its current maximum capacity to its initial maximum capacity. It is expressed as a percentage of Initial Maximum capacity and Current Maximum capacity, given in equation 3. A battery with an SOH of 80% would have a maximum capacity of 80 Ah compared to its original maximum capacity of 100 Ah. SOH provides insight into the degradation and performance of the battery over time.

$$SOH(\%) = \frac{C_{Initial}}{C_{Current}} \tag{2.4}$$

- **Nominal Voltage (V):** Nominal Voltage (V) refers to a battery's reported voltage. It is a standardized value that represents the average voltage of the battery during normal operation. The nominal voltage provides a common reference point for understanding the battery's voltage characteristics.

- **Cut-off Voltage:** Cut-off Voltage refers to the minimum allowable voltage of a battery, typically indicating its empty state. It serves as a threshold value that determines when the battery is considered to be fully discharged or depleted. Setting a cut-off voltage helps protect the battery from over-discharge, potentially damaging its cells and affecting its overall performance.

- **Cycle Life:** To a specific Depth of Discharge (DOD), cycle life refers to the number of discharge-charge cycles a battery can undergo while still meeting specific performance standards. It represents the durability and longevity of the battery, indicating how many times it can be discharged and recharged before its performance starts to decline. Generally, batteries with higher DOD tend to have a lower cycle life, as the deeper discharge levels can stress the battery's chemistry and components more.

- **End of Life (EOL):** EOL denotes the stage of a lithium-ion battery's life when the battery is unable to continue operating at the desired level or meet functional requirements. Generally, the EOL of lithium-ion batteries is described as when the

capacity drops below 70 to 80 % of its rated capacity. This stage marks the end of a lithium-ion battery's life when it can no longer sustain operations at the desired level. In this study, N_{EOL} signifies the cycle number at which the battery's capacity degrades to its failure threshold, denoted as EOL. The battery becomes unusable when its capacity drops below a critical percentage, represented as θ, of its rated capacity. To calculate the EOL, we utilize the initial capacity C_i and the critical percentage θ in Equation (2.5). The formula for N_{EOL} is expressed in Equation (2.6) [8] [54] [41].

$$EOL = \theta \times C_i \tag{2.5}$$

$$N_{EOL} = \min\{N \mid \text{Capacity at cycle } N \leq EOL\} \tag{2.6}$$

Here, N represents the cycle number, and the expression denotes the minimum cycle number at which the battery capacity falls below or equals the calculated EOL.

Figure 2.1 provides insight into estimating EOL and N_{EOL} for the NASA battery dataset. In Figure 2.1, the EOL is denoted as a red horizontal dashed line. The point at which the battery capacity reaches EOL is the point of failure of the battery. The cycle number associated with this capacity is denoted with a blue dotted point as N_{EOL}. The precise calculations and the values of θ, C_i, and N_{EOL} are discussed in detail in the later sections.

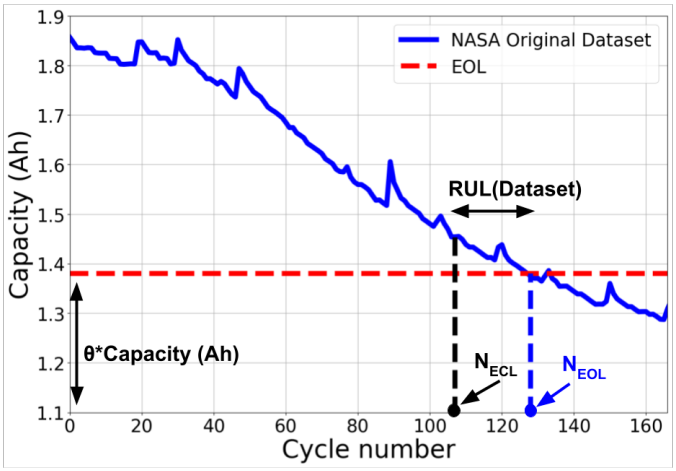

Figure 2.1: NASA Battery Dataset: Capacity Vs Cycle Number

- *Remaining Useful Life (RUL):* It refers to the battery's remaining operational lifespan, indicating how much more usage it can sustain before its capacity degrades to the EOL threshold. The Remaining Useful Life (RUL) of a lithium-ion battery is calculated using the Equation (2.7).

$$RUL(t_c) = N_{EOL} - N_{ECL}(t_c) \tag{2.7}$$

The $RUL(t_c)$ represents the remaining lifespan of the battery in terms of cycle number, which represents total charge-discharge cycles calculated at time t_c. N_{EOL} denotes the cycle number at which the battery capacity degrades to its EOL level, which can be calculated using Equation (2.6). $N_{ECL}(t_c)$ is the prediction starting cycle number at time t_c, which can be interpreted as the cycle number marking the commencement of the prediction or inference phase [41]. The Figure 2.1 shows N_{ECL} as a black point on the x-axis. It shows the cycle number at which we start predicting at any given time t_c. Refer to Section 6.4 for insights into RUL's conceptual and actual estimation.

2.2 Lithium-ion cell operation

EVs aren't powered by one big battery but rather thousands of smaller cells. Each cell has four key components that comprise a battery: an anode, a cathode, a separator, and an electrolyte. To power a device like a car, charged atoms or molecules, called ions, move from the anode to the cathode through the electrolyte, releasing their extra electrons along the way and producing electricity. The opposite happens when charging a battery: electrons flow into the battery, and the ions flow back from the cathode to the anode, creating potential energy that the battery can later discharge.

Cells, Modules and Packs:
In the realm of battery technology, there are three key components: cells, modules, and packs.

- A cell is the fundamental building block of a battery. It is characterized by its compact size and high capacity per unit volume, enabling optimal performance within limited space.

- Modules are formed by connecting multiple cells in series or parallel configurations. This clustering of cells provides added protection against external factors such as heat and vibration.

- Battery packs are created by connecting clusters of modules in series or parallel. A battery pack includes modules, a battery management system (BMS), and other devices that are responsible for controlling aspects such as cooling, temperature, voltage, and current.

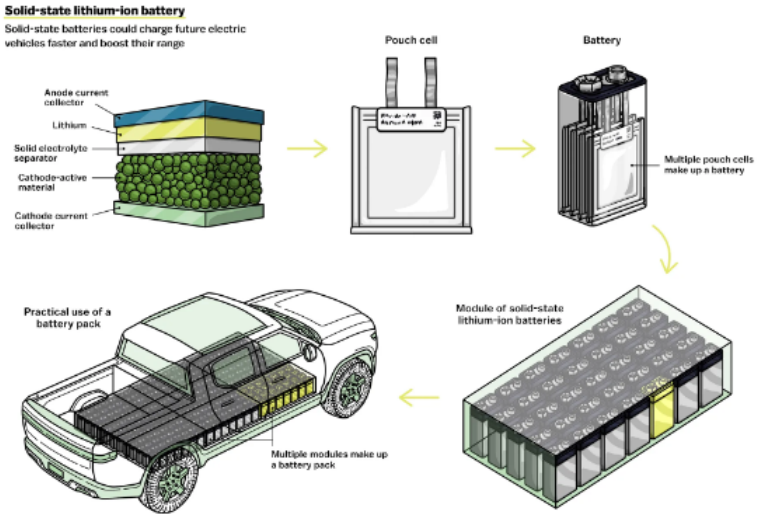

Figure 2.2: lithium-ion Battery in EVs and Practical use of a battery pack.

Components of Lithium-ion battery: Various important parts make up a lithium-ion battery, and each one is essential to the battery's operation. Among these elements are:

Cathode and Anode:
The cathode acts as the positive electrode and is usually made of lithium oxide coated on an aluminum current collector, whereas the anode functions as the negative electrode and is often composed of carbon coated on a copper current collector. Commercial lithium-ion batteries have traditionally used graphite anodes because of their steady performance at most of the voltage values [25]. The cathode's chemical processes and the current passing through the circuit are made possible by an active substance in the anode. A separator between these electrodes permits the interchange of lithium ions while blocking the flow of electrons [9].

Electrolytes:
They significantly impact the overall performance and safety of lithium-ion batteries, making them an essential component in their development. Due to their high conductivity

and durability, traditional liquid electrolytes are made of lithium salts mixed in organic solvents. Solid electrolytes are a promising development that provides increased energy density, dendrite prevention, and safety. The selection of electrolyte type is an important factor in attaining the intended performance and safety attributes of lithium-ion batteries (LIBs) [37].

Separator:
The main function of the separator in a lithium-ion battery is to keep enough space between the anode and cathode to avoid any possible short circuits. It acts as a thin, porous membrane that allows ions to move during the charging and discharging stages. Lithium-ion battery performance is highly dependent on the behaviour and structure of the separator, which affects important factors, including cycle life, energy density, battery safety, power density, and more [24].

Chapter 3

Battery capacity estimation approaches

This chapter extensively covers various models employed for time series forecasting. The first half focuses on detailed insights into different model types, including physical, equivalent circuit, empirical, and data-driven models. The latter part delves into data-driven approaches, centring on applying Long Short-Term Memory (LSTM) networks for battery capacity estimation for our research application. The discussion delves into the internal mechanics of an LSTM cell, presenting equations for its gates crucial in real-time predictions. Additionally, it explores how LSTM handles time-series data and introduces the concept of windows, laying the foundation for subsequent studies. The concluding section emphasizes the significance of parameter estimation for optimal LSTM model performance.

3.1 Types of models for capacity estimation

This section explores different forecasting models using time series data. We cover a range of approaches, including physical models, equivalent circuit models, empirical models, and data-driven models.

Physical Model-based approaches: These models are based on multi-physics and multi-scale material systems and are rooted in the intricate electrochemical processes within batteries [58]. A novel lithium-ion battery capacity prediction model utilizing the Transformer-Adversarial Discriminative Domain Adaptation (T-ADDA) architecture is presented by Liu et al. The model considers the charging voltage, charging current,

and charging temperature as input variables. They extract time series features from these inputs using a Transformer network as part of their methodology [22].

Equivalent Circuit Models (ECMs): These models use electrical circuit components like resistors and capacitors to simplify battery behaviour. ECMs are computationally efficient because they represent battery dynamics and degradation with fewer parameters. They are popular because of their less complex circuitry and low computational load requirements. One drawback of these models is the requirement of laboratory testing like electrochemical impedance spectroscopy (EIS) when it is required to update the model parameters [58] [6]. Internal resistance and resistor-capacitor (RC) pairs are among the parameters. These parameters, which are affected by temperature and state-of-charge (SOC), are frequently determined via hybrid pulse power characterization (HPPC) studies [45].

Empirical and Semi-Empirical Models: Empirical models fit a large amount of experimental data to create aging models. On the other hand, Semi-Empirical Models partially represent the aging mechanism by forecasting capacity using empirical data and mathematics. These models are developed for lithium-ion batteries to estimate the loss in battery cell life. These models provide information on the gradual reduction of battery capacity, an essential factor in studying battery performance. For instance, a battery is commonly thought to have reached its end of life stage when it reaches 70 to 80 percent of its initially rated maximum capacity. Chu et al. (2018) and Laresgoiti et al. have drawn attention to the semi-empirical lithium-ion battery degradation models proposed in recent years to evaluate battery cell life loss.

Battery life (L), a parameter introduced by the model, indicates how much of the battery's capacity has been lost. Several variables, including discharge time (t), discharge cycle depth (δ), average cycle state of charge (σ), and cell temperature (T_c), influence the per-cycle degradation rate (f_d). This rate varies throughout the battery's life and is not constant. An additional expression for the per-cycle degradation rate (f_d) is as follows:

$$fd = (1 - L) \cdot fd(t, \delta, \sigma, Tc, 1) \tag{3.1}$$

The idea that the deterioration rate and current battery life are related is introduced here (L). The deterioration rate (fd) drops as the battery ages (L rises). The battery life (L) is calculated by integrating the equation about L, considering the variations in capacity over cycles. To get L, the deterioration rate is integrated into every cycle. The battery life (L) is then expressed in terms of the initial battery life (L') by rearranging the equation, giving us:

$$L = 1 - (1 - L') \cdot e^{-fd} \tag{3.2}$$

In this context, L signifies the battery's lifetime, L' represents its initial lifetime, and f_d represents a linearized degradation rate per unit of time and cycle. This rate of degradation is influenced by factors such as discharge time (t), discharge cycle depth (δ), average cycle state of charge (σ), and cell temperature T_c. The equation captures the degradation process in a simplified mathematical manner.

We will replace the variable L (battery lifetime) with C (battery capacity) to modify the equation for practical use. Equation 3.2 then has the following form:

$$C = C_0 \cdot e^{-fd}. \tag{3.3}$$

In this case, C stands for the battery's capacity, and C_0 stands for initial capacity. The following is a close approximation to the expression fd:

$$fd = k \cdot \left(\frac{i \cdot Tc}{t_i} \right). \tag{3.4}$$

In this equation, i stands for the charge-discharge cycle, Tc stands for the cell temperature observed during the cycle, t_i stands for the length of time for discharging, and k is an empirical constant with a value of 0.13 [51] [19].

Data-Driven Approaches: While comparing the approaches for prediction, model-based approaches are challenging to execute practically because they call for a thorough understanding of physical chemistry and battery reaction processes. On the other hand, data-driven models like neural networks are less complex to understand and show good efficiency when models that predict battery degradation are created [44]. Data-driven models for battery capacity estimation use real-time data to calculate the battery's remaining capacity [18]. Many methods are used for capacity estimation, including:

1. Neural Network: In neural network-based models, interconnected neurons learn from the training data to estimate the battery capacity. Deep learning was used by Wang et al. [47] to develop a transferrable data-driven capacity prediction framework for lithium-ion batteries. Their framework uses a modular 2D network for increased efficiency and deep neural networks (DNNs) on massive datasets collected in laboratories. Zhang et al. [55] combine artificial neural networks, or ANN, with signal processing techniques. The method uses advanced filtering to create smooth, incremental capacity (IC) curves that maintain important feature values associated with battery aging. Following their extraction from

particular IC curve sections, these features are analyzed to see whether there is any association with battery capacity. The outputs of the ANN models are SOH and RUL, while the input is the essential features collected from the data. Machine learning algorithms are used more frequently to estimate lithium-ion batteries' Remaining Useful Life (RUL). These models use historical lifecycle data to forecast when a battery's performance declines significantly. Support Vector Machine (SVM), Gaussian Process Regression (GPR), Extreme Learning Machine (ELM), Deep Neural Network (DNN), and Recurrent Neural Network (RNN) versions like Long Short-Term Memory (LSTM) are some of the machine learning methods used for this purpose.

2. Support vector machines (SVMs): The flexibility of SVMs to capture complicated relations, overfitting protection, and adaptability to deal with different patterns and data formats make them an invaluable tool for capacity estimation in Li-ion batteries. A technique for simultaneously calculating the capacity and state of charge (SOC) of lithium-ion batteries is presented by Wang et al. They improve real-time and accurate battery state and health assessments by combining SVMs and recurrent neural networks (RNN) [21]. In the study presented by Patil et al., the authors introduce a methodology to predict the RUL of lithium-ion batteries by using SVMs and Support Vector Regression (SVR) [30].

3. LSTM networks: The LSTM recurrent neural network was first proposed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber. This network model is frequently used in speech recognition, audio analysis, and time series prediction applications. Over the years, the LSTM network structure has been improved with several additions, including stacked LSTM, bidirectional LSTM, and CNN-LSTM. The LSTM model is built on the RNN and contains a three-logic gate LSTM cell structure. These gates determine which information must be preserved and which must be erased. Time-series data and complicated systems are ideally suited for DNN and LSTM/RNN models, with LSTM addressing long-term dependencies [15]. Choi et al. gave the capacity estimate framework using FNN, CNN, and LSTM multi-channel machine learning methods, where the most effective learning approach was LSTM, which FNN and followed in that order. If the computing power and accessible datasets can manage its complexity, LSTM is preferred and can be used as the base model for estimating the battery RUL [7]. Using estimated State of Health (SOH) parameters as inputs, LSTM models are frequently used to predict lithium-ion batteries' Remaining Useful Life (RUL). This enables precise degradation prognosis and lifetime estimation [34].

3.2 LSTM cell architecture

In this section, we explain the inner workings of an LSTM network. A cell is a fundamental element of an LSTM network. A cell is characterized by a set of inputs and some input processing to produce a set of outputs, as illustrated in Figure 3.1. Let x_k and y_k denote the input and output of the network at the current time step k. The upper horizontal line in Figure 3.1 denotes the cell state abbreviated as s_k for the current time step k, and s_{k-1} denotes the cell state at the previous time step k-1. The cell state selectively updates, discards, or keeps information based on the information included in the input data and the context from earlier time steps. The lower horizontal line in the Figure shows the hidden layer's output, denoted as h_k at the current time step k and h_{k-1} at the previous time step k-1. It decides which data should be kept and which should be discarded at each time step.

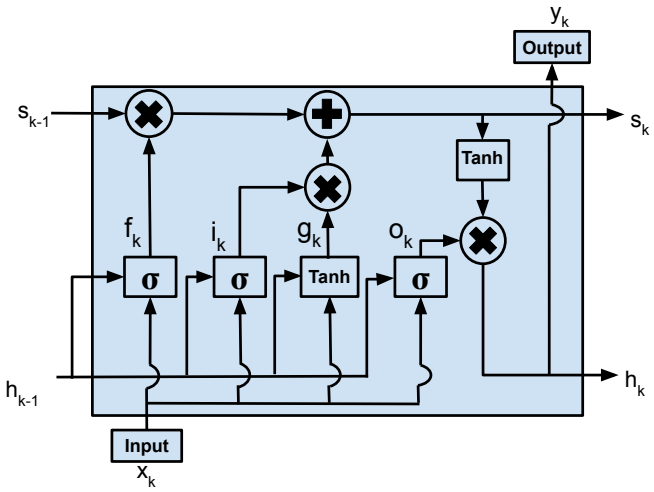

Figure 3.1: The LSTM Cell Structure

An LSTM cell mainly has three gates: the input, output, and forget gate. Equation 3.5 shows the input gate i_k operation, which chooses what new information should be added to the cell state. In the input gate equation, the weight for the gate is denoted as W_{xi}. The value b_i represents a bias value for the respective gate. The Equation 3.6 shows the operation of the output gate. It regulates the information flow from the current cell state to produce the final output at timestep k. The weight for the output gate is denoted as W_{xo}, and b_o represents a bias value for the output gate. For both the input and the output

equations, x_k is the input at time step k and h_{k-1} is the output from the hidden layer at time step $k-1$ in the past [7].

$$i_k = \sigma(W_{xi}x_k + W_{hi}h_{k-1} + b_i). \tag{3.5}$$

$$o_k = \sigma(W_{xo}x_k + W_{ho}h_{k-1} + b_o). \tag{3.6}$$

The Equation 3.7 shows the forget gate operation, abbreviated as f_k and is part of the LSTM cell unit. It determines how much of the present information should be remembered or forgotten using a sigmoid activation function denoted as σ, which assigns a value between 0 (forgetfulness) and 1 (recollection) [16]. In anticipation of the ensuing update of the cell state, a brand-new candidate value g_k is temporarily stored, given in Equation 3.8. The *tanh* is the hyperbolic tangent function.

$$f_k = \sigma(W_{xf}x_k + W_{hf}h_{k-1} + b_f). \tag{3.7}$$

$$g_k = \tanh(W_{xg}x_k + W_{hg}h_{k-1} + b_g). \tag{3.8}$$

The forget gate f_k is combined with the previous cell state s_{k-1} via element-wise product represented by $\odot$ in Equation 3.9. The input gate i_k is combined via element-wise product with the new candidate value g_k. The sum of these two products determines the updated cell state s_k. The updated hidden state h_k is shown in Equation 3.10 [28].

$$s_k = f_k \odot s_{k-1} + i_k \odot g_k. \tag{3.9}$$

$$h_k = o_k \odot \tanh(s_k). \tag{3.10}$$

The final hidden state is obtained through Equation 3.10. The final equation and the values for the weights (W_y and bias b_y in the LSTM network are obtained through a training process discussed in Section 5.5 through which we receive the final weights and bias. Finally, the output y_k is computed by Equation 3.11.

$$y_k = \sigma(W_y h_k + b_y). \tag{3.11}$$

3.3 LSTM for time series data

This section explains how the LSTM model works for time series forecasting. To process time series data to perform predictions, a sequence of LSTM cells is used, as illustrated in Figure 3.2. At any time in step k, the input to the LSTM cell is given as x_k, and after processing, the output is given as y_k. The first block represents an LSTM cell at step 1 with input and output as x_1 and y_1. The input and the previous hidden state are given to the input and output gate inside the LSTM cell to generate some values, which are further used to calculate the cell state and hidden state for the next step. The input to the first LSTM cell is commonly initialized to zero, but random initialization can also be used for the first timestep. The gating mechanisms that control the flow of information into and out of the cell are discussed in the remainder of this section. Each LSTM cell contains hidden nodes or units, which are the number of neurons in the hidden layer of the LSTM cell. They determine the size of the weight matrices and bias vectors used in the forget gate operations. These hidden units contribute to the capacity of the LSTM to capture and represent information from the input sequence. In Figure 3.2, the hidden nodes are depicted as circles inside each LSTM cell and are an essential component of the LSTM architecture. The later LSTM block at time step k receives x_k as input and uses s_{k-1} (previous cell state) and h_{k-1} (previous hidden state) at timestep k to produce h_k (current hidden state) and s_k (current cell state) and the output y_k. The s_k and h_k represent the input for the next LSTM block at time step k+1. Our focus is on univariate time series data for battery capacity and RUL estimation, where the objective is to create a model that can use previous observations to forecast the following number in the sequence precisely. Here, we describe a sophisticated Long Short-Term Memory (LSTM) model for predicting univariate time series. We recommend that you refer to Figure 3.2 for a visual illustration of the mechanism underpinning this model's functioning with time series data to comprehend it fully. It shows the connectivity of LSTM cells during model training.

The LSTM blocks represent the LSTM cells, and we will have the training data as instances of these LSTM cells in the case of forecasting battery capacity using the dataset, with each instance representing a separate battery cycle. For each cycle, the internal mechanisms of the LSTM cells, such as the forget gate, input gate, output gate, and cell state update, will be applied sequentially. The initial LSTM block receives three items as input: s_0 (cell state), h_0 (hidden state), and x_1 (current input at timestep 1), which are supplied into the system. The first block analyses the data and produces two outputs: h_1 (hidden state output) and s_1(updated cell state). The first LSTM block's outputs, s_1 and h_1 are used as inputs for the second LSTM block as training progresses. The next LSTM block in

the series receives these outputs after that. The cell state at time step k+2 (s_2) undergoes an update process that considers the impact of the prior cell state at time step k+1 (s_1), the output of the LSTM from the previous time step (h_1), and the current input (x_2). This iterative process drives the information flow through the LSTM blocks for each training cycle. Until the end of the training cycle, outputs are sequentially relayed across the LSTM blocks. The last LSTM cell is the terminal block at the end of the training, denoted as the nth timestep. The training process is completed by this terminal LSTM cell, which analyses the inputs using precise information from the last cycle and generates the correct outputs. As a result, the LSTM network can efficiently forecast battery capacity across the series of cycles in our training data and capture temporal dependencies [57].

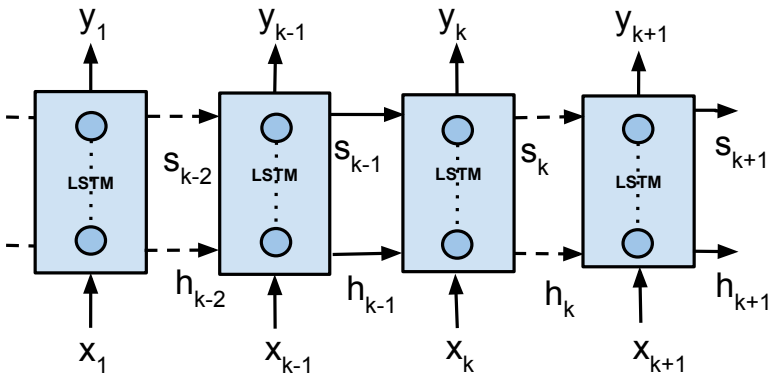

Figure 3.2: LSTM for timesteps

The training process involves iteratively feeding data which is processed and is divided into chunks of window into the LSTM model, allowing the network to learn patterns with the help of gates present in the internal LSTM cell architecture. The temporal dynamics captured by the LSTM's ability to retain and update information over these sequences facilitate the modelling and helps in learning patterns in the battery capacity time series. The window size influence the model's capacity to generalize to future capacity estimates. Windowing technique optimize the LSTM's performance for accurate and robust capacity predictions.

Our methodology adopts a systematic approach to input sequence construction for battery capacity estimation inspired by time series data windowing. An example is illustrated in Figure 3.3, where input configuration involves utilizing six previous values to predict the subsequent value in the sequence, denoting an input width of six and a target variable width of one. In the temporal context of time series data, the input spans from time zero

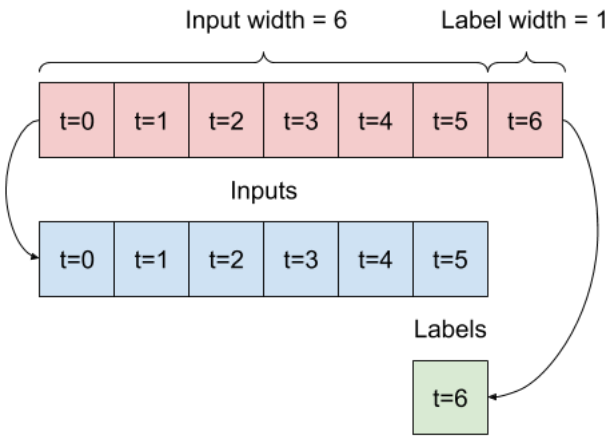

Figure 3.3: LSTM: Input and Output using window [42]

to time 5, culminating in predicting the target variable at time 6. This windowing strategy is foundational in our capacity estimation framework for batteries. The fixed window length is sequentially fed into the LSTM model for each iteration, with the window sliding incrementally by 1. In the example, during the initial iteration, the input comprises values from time zero to time 5, and this window shifts progressively for subsequent iterations. The output for each iteration corresponds to the sequence value at times 6, 7, 8, and so forth. This meticulous input-output configuration optimally leverages the LSTM model's ability to discern temporal patterns, contributing to the accurate and dynamic battery capacity estimation over sequential time points.

3.4 Key LSTM model parameters and metrics

For understanding and optimizing LSTM models in the context of battery capacity estimation, this section describes the critical parameters for this model. This exploration encompasses hidden nodes, a pivotal aspect in processing sequential data, and the window size parameter influencing the temporal context considered by the LSTM network. Moreover, we illustrate the sliding window concept in LSTM. This section also discusses the significance of Root Mean Square Error (RMSE) as a widely used metric for evaluating the accuracy of numerical predictions in capacity forecasting.

Hidden nodes: The hidden nodes are also called hidden units and are essential for processing the information as it moves across the LSTM network. Hidden nodes capture

sequential data from earlier time steps and are also responsible for storing this information.

A memory cell that can retain and store data over several time steps is present in every hidden node. One of the main characteristics of LSTMs is their capacity to remember historical data, which is crucial for identifying long-term dependencies. The information carried over from the previous steps and the information from the current input is processed by hidden nodes. They update and mix this data using various mathematical operations, allowing the network to choose what to output, forget, or recall at each given time step [38].

An LSTM layer's hidden node is a hyperparameter that can be varied in an LSTM model architecture. Expanding the number of hidden nodes improves the network's ability to identify complex patterns. In improper regularisation of these hidden nodes, the model may be overfitted [12].

Window size: The window size refers to the number of input elements or prior time steps the network considers before predicting the next time step. It controls the amount of previous data the LSTM network considers as a sequence length to predict the next value in the sequence.

The LSTM sliding window technique splits historical data into overlapping windows of a predetermined size. Each window creates a series of previous data points utilized as input to forecast the following value. In Long Short-Term Memory (LSTM) networks, window size refers to the number of successive time steps or input items treated as a single unit. A bigger window size allows the model to consider a broader temporal context, which improves its capacity to identify complex patterns in the data. The window incrementally moves through the data over time, recording fresh input sequences and updating the predicted target output. Using this method, the model may learn temporal patterns and correlations, making it helpful in predicting the capacity of batteries over time and detecting anomalies based on previous trends [3].

The basic idea is shown in Figure 3.4, where we use the sliding window methodology to anticipate future capacity values using historical capacity values. This method uses the neural network's recent temporal steps as input to forecast the next step. Combining input and output data can be called a sliding window. This methodology's flexibility comes from the adjustable parameters: the length of the input sequence and the length of the output sequence. This is flexible for time series forecasting since it enables the window size to be dynamically adjusted.

The figure shows the sliding window's width as the red block, allowing us to concentrate on the value right before the expected capacity, or the (k - 1)th term, to forecast the kth term. The window size is adjustable and represents the number of earlier observations to project future values. The picture also illustrates how we repeatedly use previous values to

anticipate upcoming ones during each subsequent iteration. To predict the 99th value, we first use the values inside the window. Next, we forecast the 100th value using the window size plus the 99th value, and we repeat this iterative procedure until we have predicted future capacity values for all cycles. This method ensures that all temporal dependencies are fully captured for precise capacity forecasts across cycles [3].

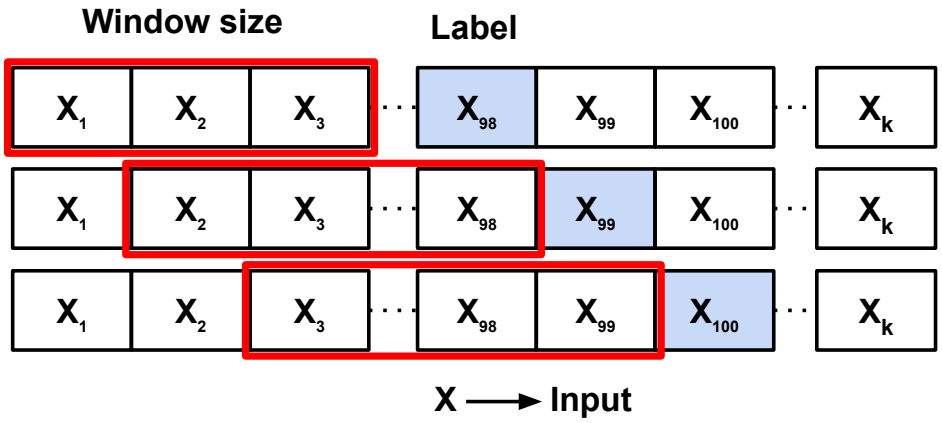

Figure 3.4: Sliding Window Concept in LSTM

Root mean square error: The square root of the mean of the squared errors is a commonly used metric for evaluating the general accuracy of numerical predictions. The formula for RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2} \tag{3.12}$$

In this case, n is the total number of observations used in the analysis, O_i denotes the observed values, and S_i denotes the predicted values for a particular variable.

Since RMSE is a good measure of precision, it is beneficial for comparing and predicting mistakes between different models or configurations related to a specific variable. It's important to remember, though, that the magnitude of the variables affects how effective RMSE is, therefore it shouldn't be used to compare errors between various variables [40].

Chapter 4

Datasets for capacity prediction

Lithium-ion (Li-ion) battery testing is necessary yet expensive and time-consuming. Open battery datasets are an excellent source for investigation and study. Some research groups have made their Lithium-ion battery data available for the public to compare different algorithms and techniques. This section discussed these critical datasets.

4.1 NASA dataset

The NASA Ames Prognostics Centre of Excellence (PCoE) is home to the battery prognostics testbed that provided the information used in this study. Commercially available lithium-ion 18650-sized rechargeable batteries were put through carefully planned trials with a variety of tools as part of the testbed gathering, which submits the batteries to three different operational profiles, including charging, discharging, and electrochemical impedance spectroscopy (EIS) at varying temperatures.

Voltage and Current Dynamics During Charging and Discharging:

The two operational profiles involving charging and discharging at room temperature were applied to four cells, designated 5, 6, 7, and 18. In this study, we used the data from cell 5. A constant current (CC) mode operating at 1.5A was used during the charging phase until the cell voltage reached 4.2V; at this point, a constant voltage (CV) mode was switched on until the charge current was lowered to 20 mA. After that, the cell voltage had to decrease to a certain level of 2.7V for cell five before the discharge phase could begin.

The dataset excerpt shown in Figure 4.1 provides a detailed overview of battery discharge operations by condensing a sample of discharge data. The figures' columns include crucial elements necessary to characterize the discharge process. The charging dataset contains the same columns as the discharging set except for the capacity values.

Each critical column in the battery's discharge dataset provides a unique perspective on how the battery behaves throughout discharge cycles. As the discharge process moves forward, the column that displays the measured voltage helps to know about the battery's terminal voltage. While the temperature variation study offers a critical perspective on the battery's thermal dynamics. It helps to study the effect of temperature change on the performance and safety of batteries. The measured current clarifies the charge flow rate, giving light to the battery's energy supply capacity. The current charge and voltage charge values explore how the battery and load interact, explaining how the battery's energy satisfies the load's requirements. Time is a chronological reference point that provides context for observing changes throughout discharge cycles. The battery's capacity to store energy is given as capacity, including its ability to supply charge up to 2.7 Volts.

Voltage Measured	Current Measured	Temperature Measured	Current Charge	Voltage Charge	Time	Capacity
4.191492	0.004902	24.330034	-0.0006	0.000	0.00	1.856487
4.190749	0.001478	24.325993	-0.0006	4.206	0.28	1.856487
3.974871	2.012528	24.389085	-1.9982	3.062	0.60	1.856487
3.951717	2.013979	24.544752	-1.9982	3.030	0.90	1.856487
3.934352	2.011144	24.731385	-1.9982	3.011	1.20	1.856487

Figure 4.1: Battery 5 Discharging field visualization

The testbench includes equipment like voltmeters, ammeters, and a thermocouple sensor suite to monitor various parameters, as well as a programmable 4-channel DC electronic load and a 4-channel DC power supply for managing discharge operations. The battery impedance behaviour was investigated using specialized electrochemical impedance spectroscopy (EIS) gear. A comprehensive dataset was produced due to the imposition of various operational circumstances in an environmental chamber. A PXI chassis-based data acquisition (DAQ) system managed the complete experimental setup, ensuring accurate control and data collecting. This collection of tools made it possible to cycle the batteries systematically under various conditions, allowing for the analysis of capacity fade and degrading effects.

Voltage and Current Dynamics During Charging: To understand how voltage and current dynamics change throughout the charging cycle, see Figure 4.2. The illustration

shows a pattern in which the current holds steadily until the voltage increases to 4.2 Volts. The voltage then stabilizes at 4.2 Volts while the current decreases gradually and eventually reaches 1.4 Amperes. This visualization clearly represents the complex interaction between voltage and current parameters throughout the charging cycle, revealing notable transitions and variations compellingly.

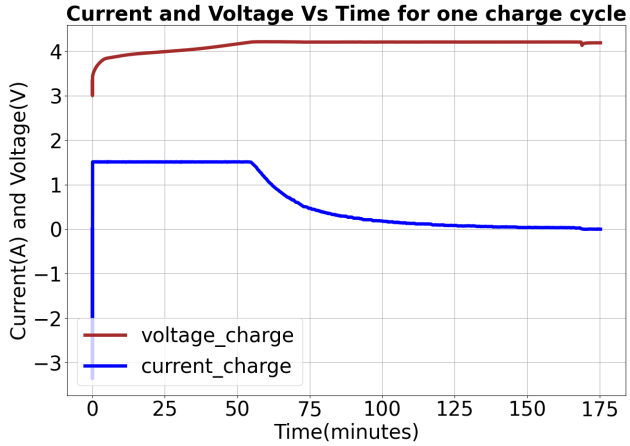

Figure 4.2: Current and voltage for Charging Cycles

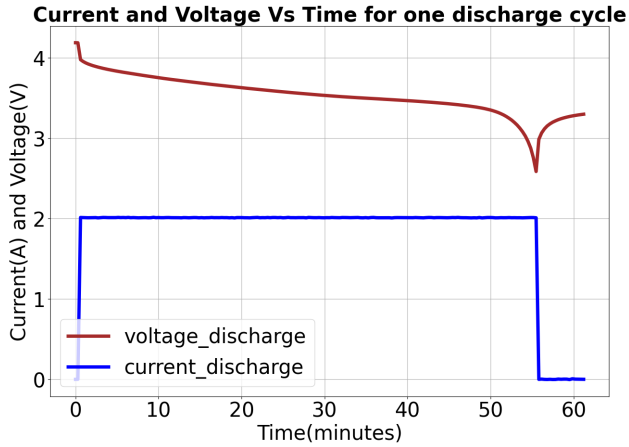

Figure 4.3: Current Vs. time for Discharging Cycles

Voltage and Current Dynamics During Discharging: An illustration of the voltage and current dynamics within the discharge cycle is shown in Figure 4.3. Notably, the graphic reveals a characteristic pattern where the voltage gradually drops until it approaches a

threshold of 2.7 Volts. Interestingly, this voltage decline co-occurs with a constant current reading of 2 Amperes, indicating a continued discharge rate. This illustration explains how voltage and current fluctuations interact throughout the discharge cycle. The figure accurately depicts the critical phases of the cycle, displaying the synchronized changes in voltage and current and illuminating the battery's operation during this time.

4.1.1 Effect of ageing on the NASA battery data

Over time, a natural consequence is the gradual degradation of battery capacity. Figure 4.4 shows the battery capacity degradation of cell 5 with the increase in the number of charging and discharging cycles. Lithium-ion batteries were subjected to discharges at different current load levels until their voltage fell to predetermined thresholds as part of the testbed dataset collection. Some thresholds were purposefully placed below the manufacturer-recommended value of 2.7 V to cause deep discharge aging effects. The battery's increased aging was caused by this charging and discharging cycle. The trials were declared complete when the batteries achieved the end-of-life (EOL) requirement, defined as a 30 percent decline in rated capacity, particularly going from 2 Ah to 1.4 Ah. By carefully cycling the batteries under controlled circumstances, they were able to record the effects of various discharge currents and deep discharge situations on battery deterioration and capacity decline.

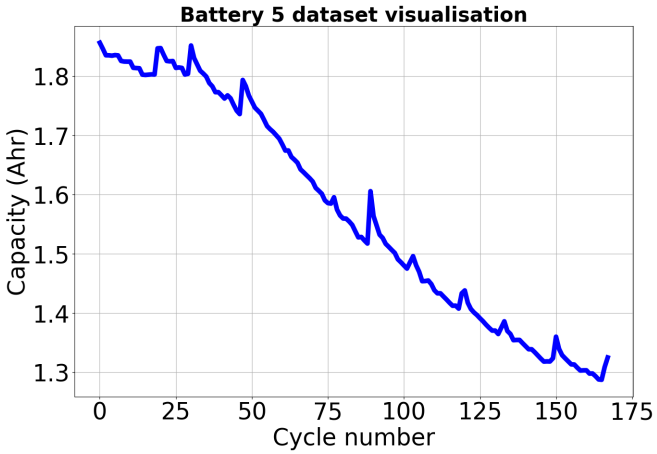

Figure 4.4: Capacity vs Cycles

Effect of aging on Charging: To observe the aging on charging, we'll examine the peculiarities of the changing current, voltage, and temperature charging parameters over

time.

With the help of this study, it is possible to spot trends, outliers, and differences between fresh and aged batteries and learn important details about battery performance, deterioration, and potential failure processes. Through these changing parameters, we will observe the capacity degradation as time passes for the batteries. We will plot the charging cycle's current, voltage, and temperature parameters.

Figure 4.5, 4.6, and 4.7 the graph of changing charging parameters: Current, Voltage, and Temperature concerning time. It shows the voltage, current, and temperature charging profiles for different remaining capacities, including cycles 1, 50, 100, and 150. Notably, the old battery begins to lose current from the constant current phase sooner than the fresh battery and achieves a value of 4.2 V earlier than the fresh battery. In addition, compared to a fresh battery, the older batteries reach a higher maximum temperature faster. These findings suggest that battery aging causes changes in charging behaviour, including earlier voltage peaks, quicker current drop-offs, and faster temperature increases as batteries go through cycles. These findings point to the impacts of battery degradation. The observed variations in charging characteristics, such as the earlier voltage peak, earlier current fall, and quicker temperature rise in old batteries, serve as crystal-clear indicators of the degradation process occurring over time. [29].

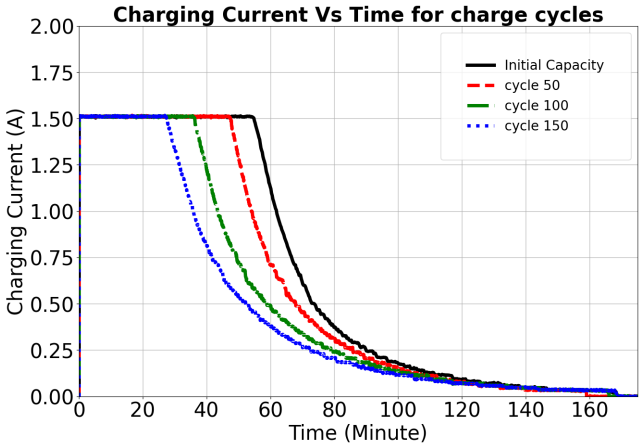

Figure 4.5: Current Vs. time for Charging Cycles

Effect of Ageing on Discharging: The dynamic variations in discharging parameters over time are shown in Figures 4.8, 4.9, and 4.10. Notably, the discharge current and voltage of the battery that is used enough and is almost at the end of its cycle (150th discharge cycle) fall more quickly than those of the battery when it was fresh (1st discharge cycle).

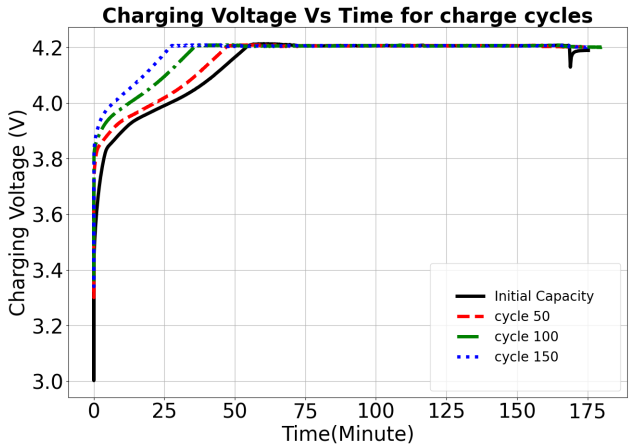

Figure 4.6: Voltage Vs. time for Charging Cycles

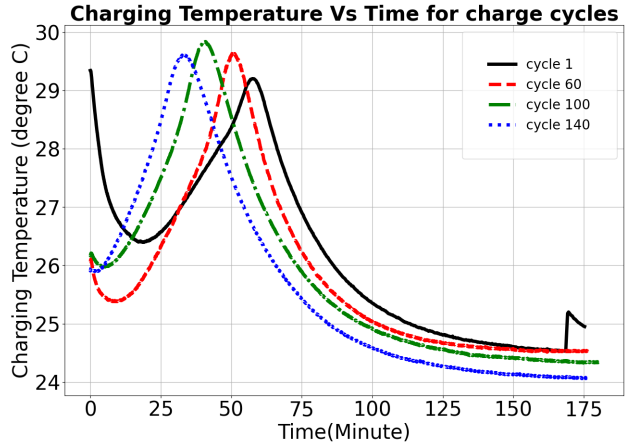

Figure 4.7: Temperature Vs. time for Charging Cycles

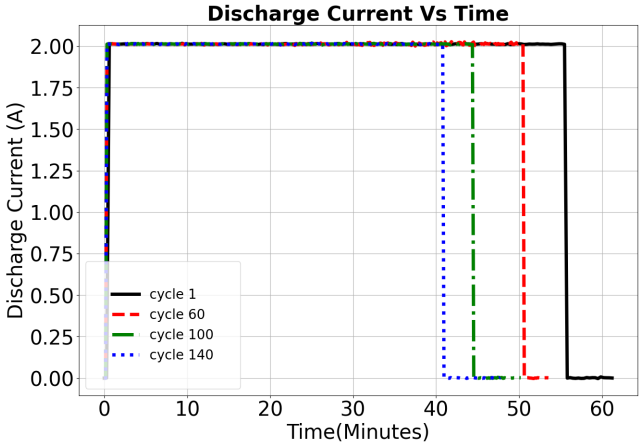

Figure 4.8: Current Vs. time for Discharging Cycles

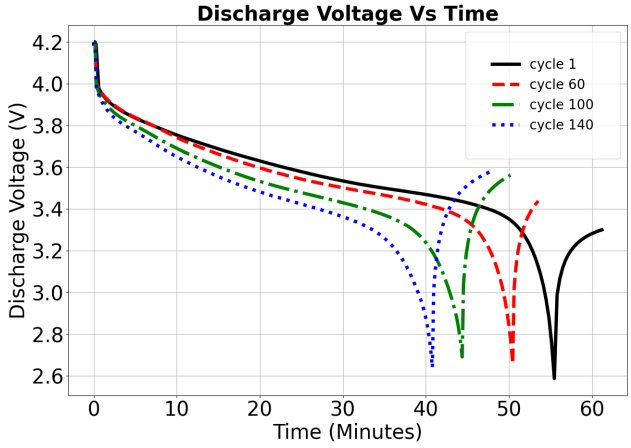

Figure 4.9: Voltage Vs. time for Discharging Cycles

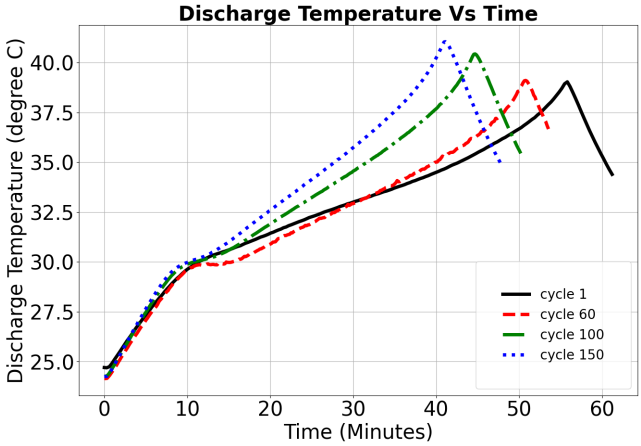

Figure 4.10: Temperature Vs. time for Discharging Cycles

In addition, the older battery shows a quicker temperature rise than the newer battery. These patterns show how battery deterioration manifests over time through accelerated changes in thermal behaviour and discharge performance.

The constant charging and draining process accelerates battery aging. The batteries were evaluated in the experiments until they reached the end-of-life (EOL) standards. This paper's primary objective is to offer accurate estimates of the Remaining Useful Life (RUL) for these batteries, which is essential for comprehending their operation and making the most of their use.

4.1.2 Data Patterns and Variability in the NASA Dataset

Randomness in a dataset describes the existence of abnormalities or some observed variations in the data. This can be because of measurement errors, variability or environmental factors. Within the NASA dataset, randomness can be seen if data points diverge from anticipated trends. Good quality instruments, reproducible experimental configurations, and careful data gathering all work together to reduce the amount of randomness in the NASA dataset. Therefore randomness is given minimal weight in the modelling process in this study with the majority of the attention given to systematic trends and degradation mechanisms found in the dataset.

From Figure 4.4 one can observe some spikes in capacity measurements. This is because when successive cycles are started, the battery can experience a partial recovery of capacity.

This is called relaxation effect in lithium ion batteries. This process is explained by the internal strains in the battery relaxing and the components rearranging themselves during the rest period, which allows the battery to partially restore its lost capacity.

4.2 CALCE dataset

Research on lithium-ion batteries can benefit significantly from the battery dataset provided by the CALCE (Centre for Advanced Life Cycle Engineering). Research groups worldwide are welcome to collaborate on this dataset. It contains experimental test data on lithium-ion batteries, covering battery degradation and performance data. This dataset is helpful for battery state estimation, end-of-life and remaining usable life prediction, accelerated battery degradation modelling, and battery capacity estimation [46].

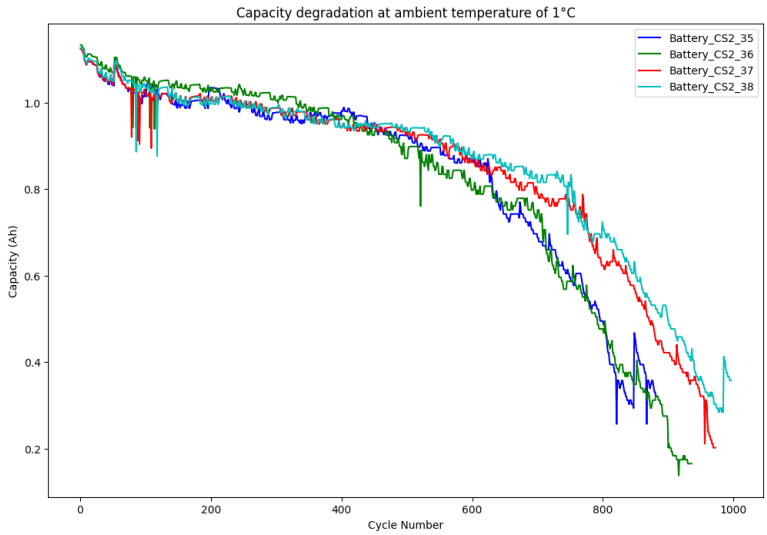

Figure 4.11: CALCE dataset

The CALCE battery dataset's characteristics are:

1. Different types of batteries included: Dataset includes a variety of battery form factors, such as prismatic, pouch, and cylindrical.

2. Diverse range of Chemistry: Includes Nickel Manganese Cobalt Oxide (NMC), Lithium Iron Phosphate (LFP), and Lithium Cobalt Oxide (LCO), which enables researchers to investigate the behaviour of distinct lithium-ion batteries.

3. Testing data profile: A wide range of test data, such as continuous full and partial cycling, storage, dynamic driving profiles, measurements of open circuit voltage, and measurements of impedance, are provided by the dataset.

5. Reliability Testing: Data from ongoing reliability testing and qualification testing are included in the dataset to evaluate how samples from various production lots perform in comparison.

The CS2 batteries are one of the types included in the CALCE dataset. These batteries went through constant current and constant voltage charging. There was a continuous current till the voltage reached 4.2 V. The cutoff voltage for these batteries is 2.7 V. The dataset contains details about the features of charging and discharging, cycling at various current rates, and other particular testing scenarios. Visualization of the capacity degradation data of CS2 cells from the CALCE dataset is given in figure 4.11. Its capacity degrades as the battery goes through continuous charge and discharge cycles. The figure contains the capacity values of four cells. One cell named $CS2_{35}$ is chosen for training and testing in this study. This cell's total number of charge and discharge cycles is 882. The maximum initial capacity for this cell is observed to be 1.1263, which gives the EOL value as 0.784 by keeping the critical percentage θ as 70 in Equation (2.5). Figure 4.12 shows the capacity degradation for $CS2_{35}$ with charge and discharge cycles, along with the EOL line.

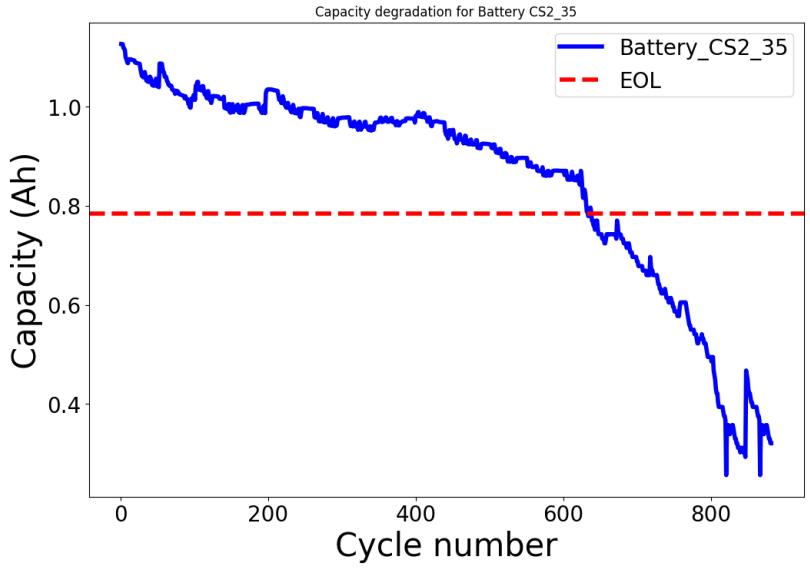

Figure 4.12: CALCE dataset: $CS2_{35}$ cell

Chapter 5

Proposed LSTM models for RUL prediction

This section introduces three models and provides information on the LSTM architecture. This section explains the inputs and outputs that each LSTM model requires for its training and operation, along with the training process. It also discusses the algorithm and pseudo code for each model. It discusses the input, output, and preprocessing steps and the training process used by LSTM networks and briefly explains the basic and proposed LSTM model using algorithms and pseudo codes.

5.1 Baseline LSTM model: LSTM 1

This section provides a detailed exploration of the basic LSTM model, LSTM-1. The discussion thoroughly explains each model's intricacies, complemented by an in-depth examination of the corresponding algorithms and pseudo code. The LSTM 1 model makes precise time series predictions using LSTM networks. This model takes the input data and the historical capacity values and feeds these values to the LSTM network. In Figure 5.1, if we have N, samples of the dataset are taken in the first layer. The data are then preprocessed and normalized to ensure that the model receives consistent input. The network gets the data and organizes it into sequences with a particular window size. Then, the network learns to forecast future values based on prior observations using the structured sequences as input. As a result, the model can comprehend the underlying dependencies and patterns in the time series data. These forecasted capacity values are then stored and

used to calculate the RMSE. The model is trained to reduce the discrepancy between its predictions and the actual data, as shown in Figure 5.1. The Root Mean Squared Error (RMSE) quantifies the accuracy of the model's predictions versus the actual data and evaluates the model's performance.

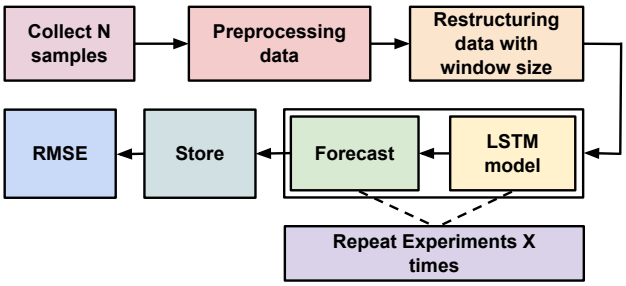

Figure 5.1: LSTM-1 flow diagram

LSTM 1 Model Implementation:

1. Preprocessing and Loading Data: The code loads a dataset first. In this instance, battery capacity data—typically gathered from lithium-ion batteries—is loaded using the 'load data' function. The data is preprocessed and normalized to guarantee that the model receives consistent input.

2. Data Splitting: - Training and testing sets are created from the dataset. Forty percent of the data is used for testing, and sixty percent is used for training.

3. Transformation of Data: The method to create a dataset in the code is responsible for transforming the dataset into a format appropriate for time series prediction. It generates pairs of input-output, where the output is the upcoming capacity value to be predicted, and the input is a series of past capacity values.

4. Model Architecture: TensorFlow's Keras API generates an LSTM model. The model comprises units called neurons, which are present in the LSTM layer. There is a single unit-dense layer. The Adam optimizer and the mean squared error loss function compile the model. With this configuration, the model is ready for training.

5. Training the Model: The parameter number of epochs refers to how many times a machine learning model goes through the whole training dataset in one iteration. The parameter batch size controls how many training instances are processed in a single iteration, which impacts training speed and memory usage. The model is trained on the training data, and during training, the model weights are updated using the loss function.

6. Creating Predictions: After it has been trained, the model is used to create predictions on training and testing data. The model generates predicted capacity values using a predict function observed from the pseudo codes.

7. Inverse Transformation: Using the Min-Max scaling earlier in the code, the predicted values are inversely transformed to their original scale. This step ensures that the projections and the original data have the same units.

5.2 Proposed LSTM models: LSTM-2 and LSTM-3

In this section, the inputs and outputs that each LSTM model requires for its training and operation are explained along with the training process. This section also discusses the algorithm and the pseudo code for each model.

5.2.1 LSTM-2 model

LSTM-2 uses a dynamic retraining approach that entails adding every predicted test pattern to the training dataset and then improving the model for two additional training epochs before the subsequent forecast. Through this iterative process, the LSTM model is guaranteed to catch initial patterns and adapt to changing trends, improving its accuracy and propensity for prediction. Figure 5.2 shows the flow diagram of this approach. The unique aspect of this method is its retraining mechanism, which allows the LSTM model to adjust to changing patterns and maintain its accuracy over time. After each cycle, the most recent data point is added to the training set. The model can wait for more cycles, too, which is discussed in the next section. For this case, we assume the Z value is 1. Predictions are made for the test dataset using the trained model, and then these forecasts are shrunk to the original data range. In the model's flow diagram, a fundamental block shows how patient the model is as it waits for a predetermined amount of samples given as Z before smoothly integrating them into the training set for retraining. This retraining technique helps the model become more accurate and flexible over time. The root mean squared error (RMSE), which measures the discrepancy between the predicted and actual values, is used to evaluate the model's performance. The entire procedure is repeated for a predetermined number of cycles, with the performance results being kept and compiled to ensure resilience. This iterative approach uses the LSTM's capacity to record temporal relationships and the continual retraining mechanism to improve capacity estimation accuracy in dynamic scenarios, leading to a more in-depth comprehension of system capabilities in the given context.

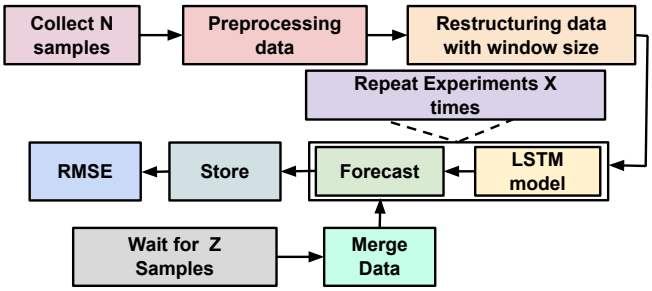

Figure 5.2: LSTM-2 flow diagram

LSTM-2 Implementation:

By adding each predicted test pattern to the training dataset and dynamically retraining the LSTM model over two training epochs before the next forecast, LSTM 2 employs an incremental approach with retraining to increase accuracy over time. The LSTM model's accuracy and prediction abilities are improved by this iterative process, which also helps the model identify early patterns and adjust to shifting trends. The root mean squared error (RMSE) is used to assess model performance when the procedure is performed a predefined number of times. The implementation steps are very similar to those of LSTM 1, but the retraining part is extra.

1. Dataset and preprocessing: After loading the dataset, carry out any required data preprocessing. To build input-output pairs, transform the dataset into a supervised learning format. Then, the information is stored in test and train sets. We are scaling the data to a desired range, from -1 to 1.

2. Incremental Training: To evaluate the model's performance, do an iterative experiment using the original data training set. Then, update the model using the new data point from the test set for each time step in the test dataset and utilize the LSTM model, creating a forecast in one step. To get the prediction within the initial data range, reverse the scaling and differencing procedures. For retraining, append the new data point from the test set to the training copy. The training copy, a duplicate of the original training dataset, is appended with the new data point from the test set. The model can now learn from the latest data thanks to this modification. For the whole test dataset, the procedure is repeated.

The difference with LSTM-1:

1. The LSTM-2 model employs an incremental technique that adds new data to the training dataset, continuously enhancing the model's capacity estimation. In contrast,

LSTM 1 trains the model from scratch only once. Each predicted test pattern is added to the training dataset to retrain the LSTM-2 model. It goes through more training epochs after adding fresh data before generating the next forecast. As a result, the LSTM model can adjust to changing trends and patterns in the data.

2. LSTM-2 uses incremental training to seamlessly incorporate fresh data into the training set so that the model does not suddenly forget previously learned patterns. It offers a continual learning process, particularly helpful in dynamic situations.

5.2.2 LSTM-3 model :

This experimental model used a unique wait-and-retrain strategy to improve the LSTM model's prediction performance for the Remaining Useful Life (RUL) estimate. In this method, the model's predictions were based on data gathered from ten additional operating cycles before the retraining procedure was started. The model obtained enough operational data to capture recent battery behaviour patterns by purposefully delaying. The LSTM model underwent retraining utilizing the newly received data from the prior cycles after the ten-cycle waiting time. The most recent operational insights were easily incorporated during this retraining phase, enabling the model to adjust to any changing dynamics in battery performance. The model's ability to predict outcomes could be improved by including a larger and more recent dataset, considering the shifts and trends noticed throughout the waiting period. The main goal of the wait-and-retrain strategy was to balance information assimilation with promptness. While waiting ten cycles would cause forecasts to be delayed, it made sure that the model was aware of recent operational changes, perhaps resulting in more precise RUL estimates. In Figure 5.2, we can set the parameter Z, which is the number of samples that the network waits for is 10. This strategy attempted to overcome the drawbacks of non-retraining and incremental retraining after every cycle by allowing the model to wait for more data for retraining. It provides a potential path for increased predicting accuracy in dynamic battery operational circumstances and decreases the computational requirements.

LSTM-3 Implementation:

To enhance the model's prediction performance for Remaining Useful Life (RUL) estimation, LSTM 3 presents a novel "wait-and-retrain" technique. Here's a summary of the main stages involved in implementation:

1. Repeats and Updates: To ensure statistical robustness, the experiment is programmed to run for a predetermined number of "repeats," or periods during which the entire procedure will be repeated. The' updates' parameter determines the number of

updates or retraining epochs the model experiences following the waiting time. The parameter 'update interval' indicates the number of data points at which the model will be updated. In this case, it is set to 10.

2. Waiting and Retraining: The wait and retrain approach is used in LSTM-3 model. The model produces a one-step forecast for each repeat and time step in the test dataset. The model begins retraining after a set waiting period which is indicated as update interval in this study. Retraining allows the model to adjust to the latest operational insights using the current data from the recent historic cycles. For better forecasts, the model must be able to capture changes and trends noticed throughout the waiting period.

3. Performance Evaluation: The root mean squared error (RMSE), a measure of prediction accuracy, is used to assess the model's performance following each repetition. To evaluate the model's overall performance, the RMSE scores are gathered for every repeat. The goal of the wait-and-retrain approach is to achieve a balance between promptness and information absorption. It purposefully postpones retraining by waiting for ten more operational cycles to record recent battery behaviour trends. This the method may resulis t in more precise RUL estimations when dynamic battery functioning.

5.3 PICE-LSTM

The PICE-LSTM model distinguishes itself from LSTM-1, LSTM-2, and LSTM-3 through two fundamental assumptions. Firstly, it acknowledges that accurately estimating the RUL requires a multi-step prediction approach. Instead of relying on a single-step prediction, the model employs a method that does not depend on test data during the inference phase. It iteratively uses the predicted values as input for subsequent predictions, enabling the forecast of multiple capacity values into the future, ultimately yielding the estimated RUL.

Secondly, the model adopts a pragmatic approach by recognizing that utilizing the entire historical dataset, from cycle one to the present, may not be necessary. The model selectively employs a smaller batch of data to streamline training and overcome data volume challenges. This batch, typically consisting of the data preceding the prediction initiation point (e.g., 25 points before cycle 100), serves as a more focused training set. This strategic use of recent data ensures the model remains relevant and practical, mainly when large historical datasets are impractical or provide diminishing returns.

This model's efficacy becomes apparent when real-time predictions are essential, and data availability is limited. By enabling multi-step predictions and optimizing data utilization, PICE-LSTM is a practical solution for electric vehicle battery capacity estimation,

offering both accuracy and efficiency in dynamic, real-world applications. Figure 5.3 shows the architecture of this model. The first step involves collecting and processing the data. The next step consists of defining a window with all the past points that a model will use each time for a forecast. The defined training data is used to train the LSTM model. After the training, the trained LSTM model does a one-step forecast. This value is stored in the window, and this new updated window is used to forecast the next value in sequence. This process repeats until we notice that the forecasted capacity values are below the threshold, indicating the battery's EOL.

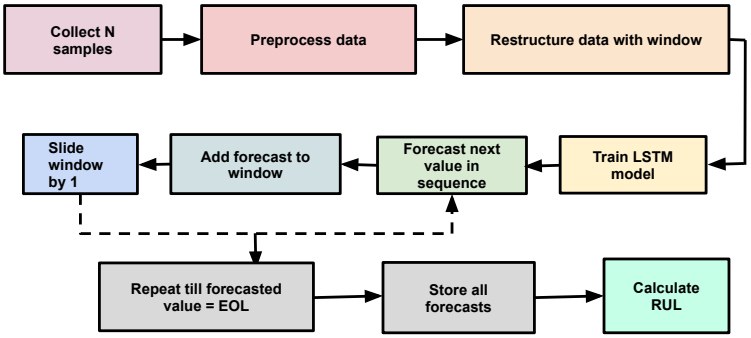

Figure 5.3: PICE-LSTM flow diagram

5.4 Pseudo code for proposed LSTM models

In this section, the inputs and outputs that each of the LSTM models requires for its training and its operation are explained. The variables used in the model are given in Table 5.1.

Consider an input sequence denoted by X and given as $[x_1, x_2, x_3, \ldots, x_k]$, where k represents the number of samples in the dataset. The first step is to normalize the data using min-max normalization, as shown in Equation (5.1), where X' represents the normalized data for X, X is the original data, $\min(X)$ is the minimum value of X, and $\max(X)$ is the maximum value of X.

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{5.1}$$

The next step involves dividing the sequence X' into equally spaced time series segments $\hat{X}$ by using a sliding window of length w. Sliding window methodology is used to

Table 5.1: Variables and Their Meanings in Proposed Model

Variable	**Meaning**
X	Original time series data
X'	Min-max normalized time series data
$\hat{X}$	Sliding window output
d	Number of hidden nodes
B	Batch size
e	Number of epochs
q	Train-test split percentage
α	Learning rate of optimizer
ϵ	Number of updates
Z	Update interval
w	Window size
b	Final Bias term in the algorithm
W	Final weight in the algorithm
Y	Predicted time series

anticipate future values using historical values. The Figure 3.4 shows the sliding window width as the red block, allowing us to concentrate on the value right before the expected output. The window size is adjustable and represents the total number of earlier observations used to predict future values. It can be seen as slicing data into small lists of sizes to pass on to the LSTM model at each step [3] [14].

We start with a normalized sequence $X' = [x'_1, x'_2, x'_3, \ldots, x'_k]$. This sequence is then divided into smaller slices, each of length w, with an overlap of 1, forming a vector $\hat{X} = [\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_k]$, where each $\hat{x}_i$ is a vector of w elements. The first vector, $\hat{x}_1$ is given as $\hat{x}_1 = [x'_1, x'_2, x'_3, \ldots, x'_w]$. Subsequently, each following vector, like $\hat{x}_2$, is obtained by shifting the window by one element, including the next element in the sequence (x'_{w+1}). This process continues, and in general, the k-th vector $\hat{x}_k$ represents a window of w consecutive elements by shifting the window by one element at each time step: $\hat{x}_k = [x'_{k-w+1}, x'_{k-w+2}, \ldots, x'_k]$. This sliding window approach captures sequential patterns in time series data, enabling the creation of training examples for a model based on overlapping windows of the original sequence. Each vector represents a window of consecutive elements and the window shifts by one element for each subsequent vector. The process can be written as an Equation represented as Equation (5.2).

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_k \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & \dots & x'_w \\ x'_2 & x'_3 & \dots & x'_{w+1} \\ \vdots & \vdots & \vdots & \vdots \\ x'_{k-w+1} & x'_{k-w+2} & \dots & x'_k \end{bmatrix} \tag{5.2}$$

The next crucial step involves splitting the data into train and test sets. A certain percentage of the available data, X, is designated for training the model, while the remaining portion is set aside for validation or testing. This separation ensures that the model can learn patterns from the training data.

LSTM-1: The proposed algorithm, LSTM-1, is designed for time series prediction. The model requires input values presented as a normalized and window-sliced time series for training. This implies that for prediction made for the time step k, the model looks back at the series of values from $k - w + 1$ to k. The input provided to the model will be $\hat{x}_k$, with the corresponding output denoted as y_k. The relationship between input and output is captured by Equation (5.3) [50].

$$y_k = p(\hat{x}_k) \tag{5.3}$$

At cycle k, w represents the length of the sliding window, and p denotes the nonlinear prediction function derived from the LSTM model. The inputs to the LSTM model are given in Algorithm 1 along with the intermediate steps required to generate the output from the input. The algorithm needs as input a time series dataset X along with parameters such as time window size w, hidden nodes d, batch size B, epochs e, and learning rate α. The algorithm comprises two main phases. In the first phase (Training), the input time series is normalized, and a sliding window is applied to create subsequences ($\hat{X}$). A training set ($Train\_data$) is formed for each subsequence, and an LSTM model is trained using specified parameters. The Adam optimizer is employed for optimization. In the second phase (Inference), the final network parameters obtained from training are used to predict the output (Y) using the sliding window approach. The goal is to get the expected time series $(y_{n+1}, y_{n+2}, \dots, y_k)$.

In Figure 5.4, we also provide a pseudo-code consisting of three main functions. The first function, named scale, is for data scaling, consisting of Min-Max scaling of the original dataset X. Second is LSTM_model, which is for building and training the LSTM-1 model.

The third function window is for creating time series segments using a sliding window. The primary function, including all these three functions, is called Predict, which reads

Algorithm 1 LSTM-1: Time Series Prediction

Require: X: Time series data, w: Window size, d: Hidden nodes, B: Batch size, e: Epochs, α: Learning rate

1: **First Phase (Training):**
2: Min-Max normalize time series X to get X'. (Refer to Equation (5.1))
3: **for** $i = 1$ to k **do**
4: Divide sequence X' into equally spaced time series segment $\hat{X}$ using a sliding window of length w (Refer to Equation (5.2) and Figure Figure 3.3)
5: $Train\_data \leftarrow q\% \times \hat{X} = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n\}$
6: $Train\_model \leftarrow$ LSTM_model$(Train_data, d, B, e)$ {Refer to Equations **??**-3.11 for LSTM cell working}
7: $Optimiser \leftarrow tf.train.AdamOptimizer(\alpha)$
8: **end for**
9: **Second Phase (Inference):**
10: Let W and b be the final network parameters (weights and bias) obtained in the first phase.
11: **Output:** $Y = W \times X + b$ (Refer to Equation (3.11))
12: **Target:** Get predicted time series $Y = \{y_{n+1}, y_{n+2}, \ldots, y_k\}$.

a battery dataset, scales it, generates time series segments, trains an LSTM model, and predicts on test data. The algorithm uses the Min-Max scaling method, constructs an LSTM model with specified architecture, and utilizes a sliding window approach for time series segmentation.

LSTM-2: The LSTM-2 model is proposed with an identical training phase as LSTM-1. The key distinction lies in the inference phase model, which is retrained to predict future time steps better. The algorithm 2, LSTM-2, is designed for incremental time series prediction. The first phase (Training) normalizes the time series data, divides it into segments, and trains the model using a sliding window approach. The second phase (Inference) involves updating the model for subsequent data points. The final network parameters are used to predict the time series. The target is to obtain the expected time series Y with specific indices. The algorithm incorporates critical elements such as window size, hidden nodes, batch size, epochs, learning rate, updates, and update interval to optimize the training and updating process.

The presented algorithm is a two-phase process for forecasting time series data using an LSTM neural network. In the first phase, referred to as the training phase, the processing is similar to the LSTM-1 model. In the second phase, known as the inference phase, the trained model is updated (Update_model) for $k-n$ cycles with specified update parameters (ϵ and Z). The final network parameters obtained are used to predict the output Y by applying them to the sliding window approach. The goal is to obtain the predicted time series $Y = \{y_{n+1}, y_{n+2}, \ldots, y_k\}$ using the obtained weights (W) and bias (b) in the output calculation.

The pseudo-code in Figure 5.5 represents LSTM-2, tailored for time series prediction with incremental updates. The Predict_update function loads the battery data, scales it, and prepares the training and test data. It then trains an LSTM model and updates it incrementally. The scale function normalizes the dataset, the window function creates time series segments, and the LSTM_model function sets up the LSTM model. The update_model function is designed to refine the LSTM model (Train_model) during the inference phase. It iterates through the provided time series data (Test_data) and, at specified intervals (update_interval), which is set to 1 for LSTM-1, which gives Z=1, performs a set number of updates (updates) to the existing model. A new LSTM model is trained using the augmented training data (Train_data) for each update. The predictions (predict) are then generated based on this updated model. Subsequently, the function extends the training data by incorporating the current time step of the test data. This iterative process enhances the adaptability of the LSTM model, allowing it to evolve and improve its

Algorithm 2 LSTM-2: Incremental Time Series Prediction

Require: X: Time series data, w: Window size, d: Hidden nodes, B: Batch size, e: Epochs, α: Learning rate, ϵ: Updates, Z: Update interval

1: **First Phase (Training):**
2: Min-Max normalize time series X to get X'. (Refer to Equation (5.1))
3: **for** $i = 1$ to k **do**
4: Divide sequence X' into equally spaced time series segment $\hat{X}$ using a sliding window of length w (Refer to Equation (5.2) and Figure 3.3);
5: $Train\_data \leftarrow q\% \times \hat{X} = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n\}$
6: $Train\_model \leftarrow$ LSTM_model$(Train_data, d, B, e)$ {Refer to Equations (**??**)-(3.9) for LSTM cell working}
7: $Optimiser \leftarrow tf.train.AdamOptimizer(\alpha)$
8: **end for**
9: **Second Phase (Inference):**
10: **for** $i = n + 1$ to k **do**
11: Update_model$(Train_model, \epsilon, Z)$
12: **end for**
13: Let W and b be the final network parameters (weights and bias) obtained in the second phase.
14: **Output:** $Y = W \times X + b$ (Refer to Equation (3.11))
15: **Target:** Get predicted time series $Y = \{y_{n+1}, y_{n+2}, \ldots, y_k\}$

predictions in response to changing patterns in the time series data. The final output of the function is the set of forecasts Y.

LSTM-3: This model addresses computational challenges by adopting a wait-and-retrain strategy. This approach aims to mitigate the computational load associated with retraining the model at each time step. The algorithm and code for LSTM-3 with Z=10 remain identical to those of LSTM-2, Z=1, with the only distinction being the value assigned to the update interval. In LSTM-2 at Z=1, we sought to update the model at every time step, and with LSTM-2, Z=10, the model is updated after every ten samples. Essentially, this model can be viewed as a variation of LSTM-2, where the Z value is adjusted to 10, introducing a delay before model updates. This strategy enhances computational efficiency while maintaining the essence of the LSTM-2 methodology.

PICE-LSTM: The PICE-LSTM model has the same training process as the other models. However, to make it easier to use in the real world, we are using limited values of capacity cycle data for training the model, as real-world batteries have extensive lifespans. To predict the RUL at a point, there is no compulsion to use the complete historical data; instead, the last previous samples to train the model can be used. For the inference phase, instead of passing the model with a test set, we construct a custom set that leverages predicted values to do forecasting; at each time step, the one-step forecast is saved in the window that was initially created for predicting. The window gets shifted by each point each time for the next prediction. The program runs till all the forecasts have been done.

The algorithm 3 explains how the model works for calculating the RUL. In the first training phase, we consider a time series of data that undergoes normalization and then is divided into space-time series using a window. Some part of this data is used for training using a percentage q of $\hat{X}$. In the next inference phase, instead of passing the test set to the model, the predicted values to perform the forecast are passed on to the model at each time step. These predicted values are saved in a list, which is shifted by one point after each step for the next prediction.

Figure 5.6 gives the pseudocode for the PICE-LSTM model. The training phase includes pre-processing the data and performing window operations. The next step is the training process using the LSTM networks. In the inference phase, the model does not utilize a conventional test set but instead uses predicted values for forecasting. Each forecast is saved in a window that changes with each prediction using a one-step forecasting process.

Algorithm 3 PICE-LSTM

Require: X: Time series data, w: Window size, d: Hidden nodes, B: Batch size, e: Epochs, α: Learning rate, ϵ: Updates, Z: Update interval

First Phase (Training):
Min-Max normalize time series X to get X'. (Refer to Equation (5.1))
for $i = 1$ to k **do**
    Divide sequence X' into equally spaced time series segment $\hat{X}$ using a sliding window of length w (Refer to Equation (5.2) and Figure 3.3);
    $Train\_data \leftarrow q\% \times \hat{X} = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots, \hat{x}_n\}$
    $Train\_model \leftarrow$ LSTM_model$(Train_data, d, B, e)$ {Refer to Equations (**??**)-(3.9) for LSTM cell working}
    $Optimiser \leftarrow tf.train.AdamOptimizer(\alpha)$
end for
Second Phase (Inference):
Initialize an empty list $Forecast\_Window$ {To store one-step forecasts}
for $i = n + 1$ to k **do**
    $Xi \leftarrow$ Take_W_Values$(\hat{X})$ {Take w values from $\hat{X}$, named Xi}
    $Forecast \leftarrow$ One_Step_Forecast$(Train_model, Xi)$
    Append $Forecast$ to Xi
    Shift Xi by 1 position {Window shifting by 1 for each forecast}
end for
Let W and b be the final network parameters (weights and bias) obtained in the second phase.
Output: $Y = W \times X + b$ (Refer to Equation (3.11))
Target: Get predicted time series $Y = \{y_{n+1}, y_{n+2}, \ldots, y_k\}$

5.5 Training process for LSTM networks

LSTM networks update their weights and biases at every time during training. The updating of weights and biases is part of the backpropagation through time (BPTT) algorithm, a variant of the standard backpropagation algorithm used in training recurrent neural networks [4]. Here's a brief overview of how the weight and bias updates occur in an LSTM during the training process:

- *Forward Pass:* The LSTM processes each input time step k sequentially during the forward pass. The input normalized sequence X' and the previous hidden state and cell state are fed into the LSTM cell. The LSTM cell performs computations using its parameters (weights and biases) discussed in section 3.2 to produce an output and update the current hidden state and cell state.

- *Loss Calculation:* The output of LSTM at each time step is compared to the target value. Loss is calculated by reflecting the difference between predicted and actual values.

- *Backward Pass (Backpropagation)* Loss is backpropagated through time using the BPTT algorithm. At each time step, loss gradients with respect to parameters (weights and biases) are calculated. Gradients indicate contributions to the error and are used for parameter updates.

- *Weight and Bias Updates* Weights and biases are updated using an optimizer. The optimizer adjusts the parameters in the direction that minimizes the loss. The learning rate determines the size of the step during each update. The understanding of the optimizer and learning rate is elucidated through their application in the subsequent algorithmic section.

- *Repeat for Multiple Epochs:* X' is processed through the LSTM for multiple epochs e. With each e, LSTM refines parameters to improve accuracy on the X' [**?**].

Function Predict_update():

1. X = read_csv('NASA battery dataset') Load battery data
2. X' = scale(X)
3. $\hat{X}$ = window(X', w)
4. $Train\_data$ = length($\hat{X}$) $\times$ q%
5. $Test\_data = \hat{X} - Train\_data$
6. $Train\_model$ = LSTM_model($Train\_data$, Hidden nodes, Batch size, Epochs)
7. update_model($Train\_model$, Train_data, Batch size, updates, update_interval)

Function scale(dataset):

1. $scaler$ = MinMaxScaler(feature_range = $(-1, 1)$)
2. $scaler$ = scaler.fit($dataset$)
3. $scaled$ = scaler.transform($dataset$)
4. **return** $scaled$

Function window(dataset, w):

1. $dataX$ = empty_list
2. **for** i **in range**(length(dataset) - w - 1):
    1. a = dataset$[i : (i + w), 0]$
    2. append a to $dataX$
3. **return** $dataX$

Function LSTM_model(Train_data, Hidden nodes, Batch size, Epochs):

1. $model$ = Sequential()
2. $model$.add(LSTM($neurons$, Batch size)) Refer to Equations (3.6)-(3.11) for LSTM cell working
3. $model$.add(Dense(1))
4. $model$.Compile(loss, optimizer, learning rate)
5. $model$.fit($Train\_data$, epochs, Batch size)
6. **return** $model$

Function update_model(Train_model, updates, update_interval):

1. **for** i **in range**(len($Test\_data$)):
    1. **if** $i > 0$ **and** i mod update_interval $= Z$:
        1. **for** j **in range**(updates):
            1. $model$ = LSTM_model($Train\_data$, Hidden nodes, Batch size, Epochs)
            2. $predict = model.predict(Test\_data)$
            3. $Train\_data$ = concatenate($Train\_data, Test\_data[i, :]$)
2. **return** $predict$

Figure 5.5: LSTM-2 Pseudo Code

Function Predict():

1. X = read_csv('NASA battery dataset') # Load battery data
2. X' = scale(X)
3. $\hat{X}$ = window(X', w)
4. $Train\_data$ = length($\hat{X}$) $\times$ q%
5. $model$ = LSTM_model($Train\_data$, Hidden nodes, Batch size, Epochs)
6. $pred\_list$ = []
7. $batch$ = Train_data[$-n\_input$:]
8. **for** i **in range**(length(Train_data)):
    1. $pred\_list$.append(model.predict(batch)[0])
    2. $batch$ = np.append(batch[:, 1:, :], [[pred_list[i]]], axis = 1)
9. **return** $pred\_list$

Function scale(dataset):

1. $scaler$ = MinMaxScaler(feature_range = $(-1, 1)$)
2. $scaler$ = scaler.fit($dataset$)
3. $scaled$ = scaler.transform($dataset$)
4. **return** $scaled$

Function LSTM_model(Train_data, Hidden nodes, Batch size, Epochs):

1. $model$ = Sequential()
2. $model$.add(LSTM($neurons$, Batch size)) # Refer to Equations (3.6)-(3.11) for LSTM cell working
3. $model$.add(Dense(1))
4. $model$.compile(loss, optimizer, learning rate))
5. $model$.fit($Train\_data$, epochs, Batch size)
6. **return** $model$

Function window(dataset, w):

1. $dataX$ = empty_list
2. **for** i **in range**(length(dataset) - w - 1):
    1. a = dataset[$i : (i + w), 0$]
    2. append a to $dataX$
3. **return** $dataX$

Figure 5.6: Pseudo Code for PICE-LSTM

Function Predict():

1. X = read_csv('NASA battery dataset') # Load battery data
2. X' = scale(X)
3. $\hat{X}$ = window(X', w)
4. $Train\_data$ = length($\hat{X}$) $\times$ q%
5. $Test\_data = \hat{X} - Train\_data$
6. $model$ = LSTM_model($Train\_data$,
   Hidden nodes, Batch size, Epochs) # Train model
7. $predict$ = model.predict($Test\_data$) # Predict on test data

Function scale(dataset):

1. $scaler$ = MinMaxScaler(feature_range = $(-1, 1)$)
2. $scaler$ = scaler.fit($dataset$)
3. $scaled$ = scaler.transform($dataset$)
4. **return** $scaled$

Function LSTM_model(Train_data, Hidden nodes, Batch size, Epochs):

1. $model$ = Sequential()
2. $model$.add(LSTM($neurons$, Batch size)) # Refer to Equations (3.6)-(3.11) for LSTM cell working
3. $model$.add(Dense(1))
4. $model$.compile(loss, optimizer, learning rate))
5. $model$.fit($Train\_data$, epochs, Batch size)
6. **return** $model$

Function window(dataset, w):

1. $dataX$ = empty_list
2. **for** i **in range**(length(dataset) - w - 1):
   (a) a = dataset$[i : (i + w), 0]$
   (b) append a to $dataX$
3. **return** $dataX$

return $predict$

Figure 5.4: Pseudo Code for LSTM-1

Chapter 6

Experiment for comparing LSTM models and their parameters

This section will detail the particular experiments conducted, including how altering hidden nodes and window sizes affected the performance of the traditional LSTM model. The traditional LSTM approach will also be covered in this part, along with a detailed explanation of how the model was trained and how root mean squared error (RMSE) was used to assess its performance. Also, a thorough examination of the incremental LSTM technique, with a distinction between retraining and non-retraining models and an explanation of their impact on runtime and RMSE, is discussed.

6.1 Effect of changing hidden nodes

For the first experiment, we will look into the effects of changing the amount of hidden nodes inside the LSTM architecture to improve model performance. Similar to the neurons of the LSTM, these nodes control the strength of the network, and more of them provide the ability to recognize complex patterns and relationships. However, because of the larger parameter field, additional potency comes at the expense of longer training times. We will set the hidden node counts at 4, 50, 100, and 150. We aimed to comprehend the interaction between prediction accuracy and computing efficiency by methodically analyzing each hidden node count's RMSE values and runtime. The LSTM architecture was investigated utilizing different numbers of LSTM cells, ranging from 1 to 200. With a batch size of 1, 25 epochs were used to represent the number of times.

The value of variable d in the LSTM layer argument can be changed to change the number of hidden nodes in the model architecture. The LSTM layer can be defined with any number of nodes. The loop that iterates over various values for the number of concealed nodes is the critical component of the code. The LSTM design and the precise number of hidden nodes being studied are specified for every iteration. These variations in the number of hidden nodes provide a chance to comprehend how this hyperparameter modification affects the network's ability to recognize patterns.

The LSTM model is built up, trained, and used to make predictions on the training and testing datasets throughout each iteration. The important thing is to keep an eye on how the number of hidden nodes affects the model's predicted performance. To optimize the LSTM model for a given job, conducting experiments with the number of hidden nodes is crucial. Following the training and prediction stages, a plot visually represents the data and shows how well the LSTM can predict battery capacity. This illustrates how changing the quantity of hidden nodes might affect the LSTM model's accuracy and performance.

6.2 Effect of changing window size

Extensive research was done in the study to determine how different window sizes affected model performance. In the experiment, the window size parameter was gradually changed from 1 to 20 while the associated error values and runtimes were tracked. The effect of various window sizes on the functionality of the LSTM model was examined in this experimental investigation. The window size was gradually increased from 1 to 20. The dataset was loaded, preprocessed, and normalized for each window size. After that, the LSTM model was built and trained using the given window size. The root mean squared error (RMSE) was calculated for the training and test sets, providing information about the model's precision. Based on their lowest RMSE values, three of the examined window widths were chosen, and their runtimes were observed.

Implementation: To observe the impact of changing the window size in LSTM architecture, we use Python to create the LSTM model. Inside the code, we introduce the term "look back," which is the parameter we change to examine how changing the window size affects time series prediction in the context of LSTM network modelling. This manages the recollection of past information that the model uses to form predictions. The main goal is to assess how various "look back" values affect the model's forecasting ability.

To guarantee the reproducibility of findings, the code starts by initializing random TensorFlow and NumPy seeds. This phase is crucial when working on machine learning projects since it ensures that the same random processes produce consistent results

throughout multiple runs. Next, The algorithm loads the NASA dataset and takes the essential capacity values for the research. After that, it performs data preprocessing, normalizes the data using Min-Max scaling, and transforms the dataset into NumPy arrays. In machine learning, normalization is a frequent technique to bring data into a predefined range, usually between 0 and 1, so the model can learn more efficiently.

The model uses 65%(q) of the data for training and 35 % for testing, which makes n_1= 107 and n= 166. The model uses B as 1 batch size and e as 25 epochs. This division ensures that the model is tested on one subset of the data and trained on another, which aids in determining how well the model generalizes. The RMSE values for every look-back iteration are recorded in distinct lists for the training and testing predictions. These RMSE values are helpful measures for evaluating the LSTM model's performance with various window size configurations. For the proposed LSTM-2 model, retraining is done with two epochs using newly available training data. One experiment is done by keeping Z=1, LSTM-2, Z=1, and the other by keeping Z=10 for model LSTM-3. Each experiment is repeated ten times to calculate RMSE and runtime values. The Table 6.1 gives the values assigned for the experiments.

This code allows for a systematic investigation of how past data affects the prediction accuracy of the LSTM model. It provides insight into the ideal window size value for the particular forecasting task at hand and highlights how crucial model evaluation and hyperparameter adjustment are to the machine learning process.

6.3 Experimental configurations for LSTM models

This section presents a comprehensive overview of our experimental methodology. It focuses on parameter tuning for the LSTM-1 model and provides detailed configurations for the LSTM-1, LSTM-2, and LSTM-3 models in the context of RUL estimation.

6.3.1 Experiment with LSTM-1:

The algorithm starts by loading battery data and then goes through it to extract the necessary features. The capacity data is then preprocessed by applying Min-Max scaling to normalise it between 0 and 1. The sets contain 100 training samples and 66 testing samples. The architecture of the LSTM model is defined as one LSTM layer with four units, followed by a dense layer with one output unit. The model's construction uses the mean squared error loss function and the Adam optimizer. It is trained using the training

data with a batch size of 1 and 25 iterations. The model forecasts battery capacity for both the training and testing sets after training. The predictions are then inverted using the scaler to obtain capacity values in their original scale. For both the training and testing sets of data, the Root Mean Squared Error (RMSE) is generated to assess the model's performance. Finally, the cycle number is displayed against the original and forecasted capacity values to demonstrate graphically how well the model performs capacity estimate.

6.3.2 Experiment with LSTM-2 and LSTM-3:

The incremental approach without retraining was developed using the former. It converted the dataset into a supervised learning format and split it into training and test sets. The training set has 100 cycles, and the testing set has 66 cycles. Two epochs of retraining were carried out in the incremental approach with retraining, utilizing the newly available data from the training subset following each prediction iteration. The RMSE and runtime for this model were calculated and compared with the model without retraining and the original LSTM model.

In the incremental LSTM technique, the update interval is set to 10 cycles. The prediction method included this update period, enabling the LSTM model to be adjusted frequently. An LSTM model's initial training with predetermined hyperparameters occurs before incremental learning begins. The number of training epochs, batch size, and hidden node count are a few examples of these hyperparameters.

Incremental learning emulates the situation in which fresh data is always coming in. When fresh data becomes available, the LSTM model in this configuration is updated gradually. The code uses an experiment function to evaluate the predictive ability of the LSTM model. The procedure is repeated a number of times, as shown by repetitions. By differentiating successive data points, the data is first turned into a stationary state. After that, the information is transformed into a format for supervised learning that can be used to train and evaluate machine learning models. The data are separated into training and test sets. A MinMaxScaler is used to scale the training data to a specified range.

Rerunning the Experiment is when the LSTM model is repeatedly trained and assessed by the code. An entirely new LSTM model is trained for every repetition. The model forecasts each observation in the test dataset and changes its predictions with each new piece of information. To facilitate comparisons, the predicted values are reversed to reflect their original scale. The update mechanism adheres to a structured sequence to do this. From the incoming data point, it initially extracts the input features and matching target

values. After that, these extracted data are modified to meet the model's input specifications. The model is updated by fitting the new data for a single epoch and preserving the model's state.

6.3.3 Experiment with PICE-LSTM:

For the experiment with the PICE-LSTM model, we use the same data processing and training methods. The advantage of this model is that the model does not have to use all the historical data for training but instead can use some portion of historical data for training the model. This parameter is set as a train window of 50 for this experiment. The model in the inference phase saves the predicted values in a list as observed in the predict function in pseudo-code 5.6. These values are used to predict RUL values. The model uses a window of size 14 and is trained for 25 epochs. We define a test position parameter as the point at which the RUL is predicted. The values of the test position taken for this experiment are 60, 80, 100 and 125. Two experiments were conducted to test this model. The first is to compare the PICE-LSTM results with those obtained for LSTM1-1, LSTM-2 and LSTM-3 models. This experiment was performed with the same testing parameters; the training was done with the first 107 samples, and the rest were used for testing. The prediction position is fixed at 107. The second experiment takes 50 data points past the test position for training the model, and the test position is varied and is taken as 60, 80, 100 and 125.

6.4 Parameter selection for RUL estimation

In this section, we calculated the NASA dataset's RUL, taking the capacity values as input, denoted as X. Initially, we determined the EOL threshold using Equation (2.5), considering θ as 75%. The observed maximum battery capacity (C_i) was 1.856. This led to an EOL of $1.856 \times 0.75 = 1.3923$. Identifying the cycle where charging began, with a capacity of 1.392 Ah, we found the actual EOL (N_{EOL}) to be 128 cycles using Equation (2.6).

Next, RUL was calculated for the original dataset with N_{EOL} set at 128 cycles. Additionally, we determined the N_{ECL} as n with a value of 107. This yielded a RUL at this cycle of 21 cycles, calculated using Equation (2.7). The exact process was applied to the predicted series Y to compute the predicted EOL and RUL. The predicted N_{EOL} and RUL values are discussed in later Sections.

Table 6.1: Experiment Parameters for LSTM Model

Variable Notation	**Assigned Value**
n	107
k	166
Batch Size (B)	1
e (Number of epochs)	25
q (Train-test split)	65%
Z	1 (for LSTM-2, $Z = 1$)
Z	10 (for LSTM-2, $Z = 10$)
ϵ (Number of updates)	2
Repeated experiments	10

6.5 Transfer learning across similar cells in NASA battery dataset

In this study, we explore the viability of transfer learning for battery capacity estimation by training our model on cell 5 data and subsequently testing its predictive capabilities on chemically analogous cells (cell 6, 7, and 18) within the NASA battery dataset. Initially trained on 100 values of capacity from cell 5, the model is challenged to generalize its knowledge to different cells with identical chemical compositions. By evaluating its performance beyond the training cell, this experiment aims to validate the model's adaptability and assess its predictive accuracy on unseen but similar battery cells, providing crucial insights into the efficacy of transfer learning in battery health estimation.

6.6 Cross-dataset validation: assessing LSTM models on CALCE dataset

In this experiment, the LSTM models are applied to the CALCE dataset to test their performance. One cell from the CALCE dataset, the $CS2_{35}$ cell, is selected to train and test the LSTM models. Since the total cycles are 882, the first 200 samples are used for training for a good amount of exercise and the rest are used for testing. The number of epochs used for training is 25. The hidden nodes and window size are kept the same to train the model with the NASA dataset.

Chapter 7

Experimental results

In this section, we present the results and discussions from our experimentation with the LSTM-1 model, focusing on the effects of parameter tuning in Section A. It includes the impact of varying hidden nodes, d and window size, w on RMSE and runtime. Section B compares different LSTM models, including LSTM-1, LSTM-2, Z=1, and LSTM-2, Z=10 regarding runtime and RMSE values.

7.1 Effect of parameter tuning

The results of changing the hidden nodes and window size regarding RMSE and runtime are in table 7.1.

Table 7.1: RMSE and Runtime for Different Parameters

	Hidden Nodes	**Window Size**
	4, 50, 100, 150	**11, 13, 14**
RMSE	0.2, 0.07, 0.05, 0.03	0.0137, 0.0116, 0.0126
Runtime (sec)	10.35, 12.38, 16.8, 24.56	12.48, 12.6, 12.87

In Figure 7.1, capacity prediction varies with hidden nodes. RMSE decreases from 0.2 to 0.03 as hidden nodes increase from 4 to 150 before the model gets overfitted. Training and testing duration increases with more nodes, ranging from 10.35 seconds (4 hidden nodes) to 24.56 seconds (150 hidden nodes).

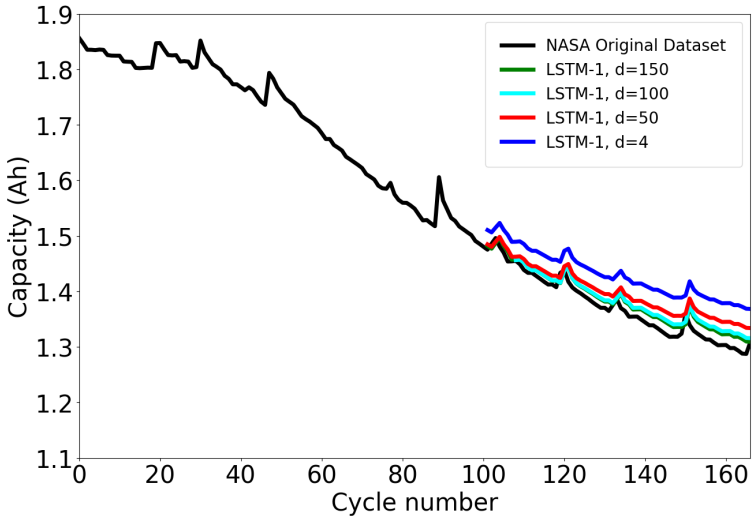

Figure 7.1: Comparing Lithium-Ion Battery Capacity Prediction with Varied Hidden Nodes

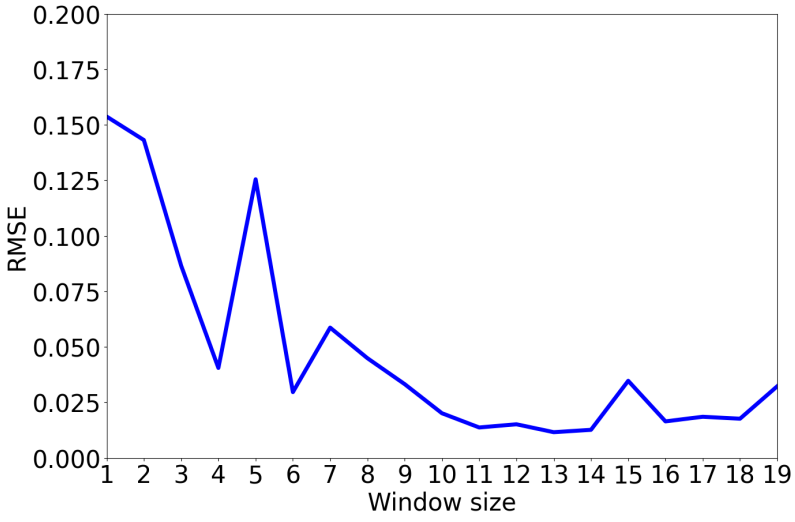

Figure 7.2: RMSE vs Window Size

Varying the window size from 1 to 20 revealed a decline in error values, particularly notable in the range of 10 to 14. The error minima at window sizes 11, 13, and 14 were remarkable, all close to 0.01 (Figure 7.2). The experiment demonstrated improved performance with more oversized input windows, up to a maximum of 14, providing the model access to a broader range of input patterns. Smaller window sizes (1 to 7) struggled to generate meaningful value functions. Notably, a window size 14 resulted in the lowest RMSE of 0.0126, while 11 yielded the most negligible runtime at 12.48 seconds.

7.2 Comparision of LSTM-1, LSTM-2 and LSTM-3

This section compares the three proposed LSTM models, LSTM-1, LSTM-2, and LSTM-3, based on the RMSE and runtime values obtained for estimating the RUL.

The results demonstrate that LSTM-2 outperforms the baseline LSTM-1 model. This can be observed in Figure 7.3. Predictions from LSTM-2 and LSTM-3 align more closely with the original dataset than those from LSTM-1. Table 7.3 details the RMSE, runtimes and predicted RUL for each model, with the proposed model exhibiting lower RMSE values (0.00982 and 0.0104) compared to LSTM-1 (0.01193), indicating improved predictive performance. LSTM-3 achieves efficiency comparable to LSTM-2 but with a shorter runtime of 6 min 82 s, attributed to fewer retraining rounds. As explained in the experiment section, the actual N_{EOL} value stands at 128, coinciding with the point where the discharge capacity reaches 1.39 Ah (EOL) in cycle 128. It is evident that the proposed LSTM-2 and LSTM-3 models closely approximate the N_{EOL} with predicted values of 129 and 130, respectively. The RUL is calculated in terms of remaining charge and discharge cycles. As the actual N_{EOL} stands at 107 cycle number, the predicted RUL by LSTM-2 and LSTM-3 is 22 and 23 charge and discharge cycles. This indicates that the battery has a remaining life of the calculated cycle numbers and will no longer be useful after these many charge and discharge cycles.

Table 7.2: Results of LSTM models with NASA dataset

Model Type	**RMSE**	**Runtime**	$\boldsymbol{N_{EOL}}$	**RUL (cycle num)**
LSTM-1	0.01193	2 min 48 s	132	25
LSTM-2	0.00982	11 min 41 s	129	22
LSTM-3	0.0104	6 min 82 s	130	23

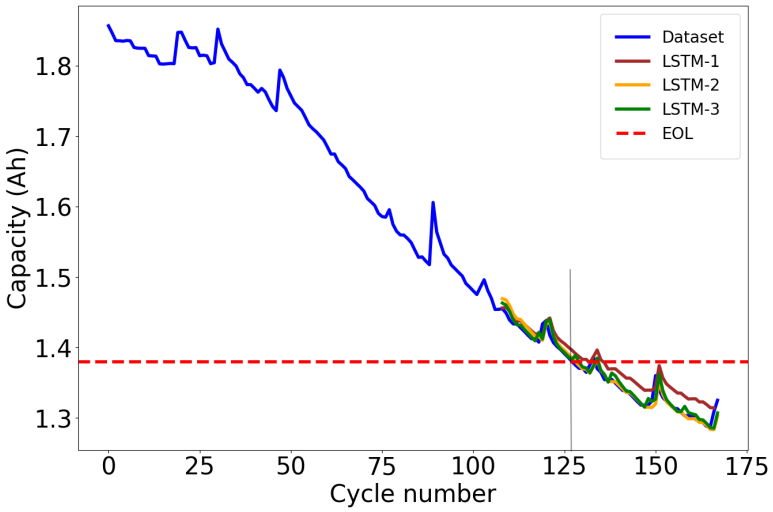

Figure 7.3: Comparision of LSTM-1, LSTM-2 and LSTM-3 with NASA dataset

7.2.1 Results with CALCE dataset

The LSTM-1, LSTM-2 and LSTM-3 models are also applied to another dataset to validate the generalizability and robustness of these models across different domains. The results are shown in Figure 7.4. The actual N_{EOL} value is cycle 630. The predicted values of N_{EOL} by LSTM-1, LSTM-2 and LSTM-3 are 776, 645 and 647 respectively. Since the starting point for prediction is cycle 200, the predicted RUL values are given accordingly in the table in terms of charge and discharge cycles. The table shows the RMSE values as well that shows that LSTM-2 and LSTM-3 have low error rate and perform better than LSTM-1.

Table 7.3: Results for LSTM models with CALCE dataset

Model Type	**RMSE**	**Runtime**	$\boldsymbol{N_{EOL}}$	**RUL(cycle number)**
LSTM-1	0.184	3 min 54 s	776	576
LSTM-2	0.0462	19 min 63 s	645	445
LSTM-3	0.0729	11 min 48 s	647	447

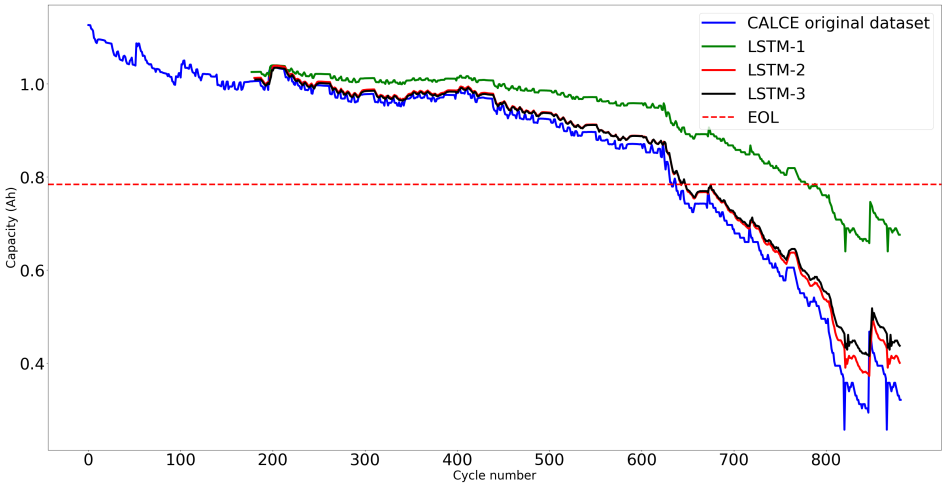

Figure 7.4: Comparision of LSTM-1, LSTM-2 and LSTM-3 with CALCE dataset

7.3 Results for transfer learning

Following the training of the proposed LSTM-3 model, incorporating retraining at regular intervals on NASA's cell 5 data, we conducted an insightful evaluation of cells 6, 7, and 18. The resulting graphs, depicting the model's predictive performance, reveal a notable success in extending its capabilities through transfer learning. The model demonstrates proficiency in predicting RUL values for cells with similar chemical compositions, showcasing the efficacy of the proposed approach. These findings underscore the adaptability and generalization potential of the LSTM-3 model, affirming its utility in predicting battery health across diverse but chemically akin cells within the NASA dataset.

The results for transfer learning are shown in Figure 7.5, 7.6, and 7.7, where the initially trained LSTM-3 model is tested on new data, which is the capacity data of cell 5, 6 and 18. This shows that the LSTM-3 model can be used to predict the RUL of similar cells.

7.4 Results for PICE-LSTM

The LSTM-1, LSTM-2 and LSTMThe PICE-LSTM model is tested with two experiments. The first experiment had 107 data points for training, and the test position was 107. The model is well-trained and can learn patterns across the training. In the first experiment, the model can predict the RUL as shown in Figure 7.8 with the actual EOL being 128 and

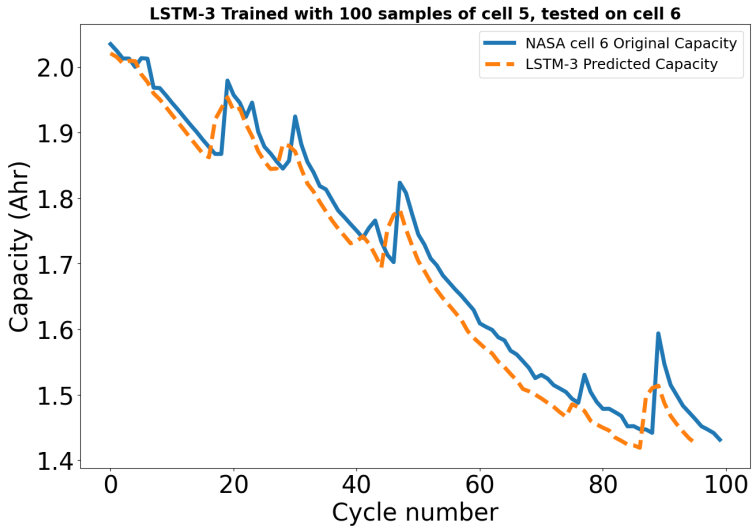

Figure 7.5: Transfer learning result on cell 6

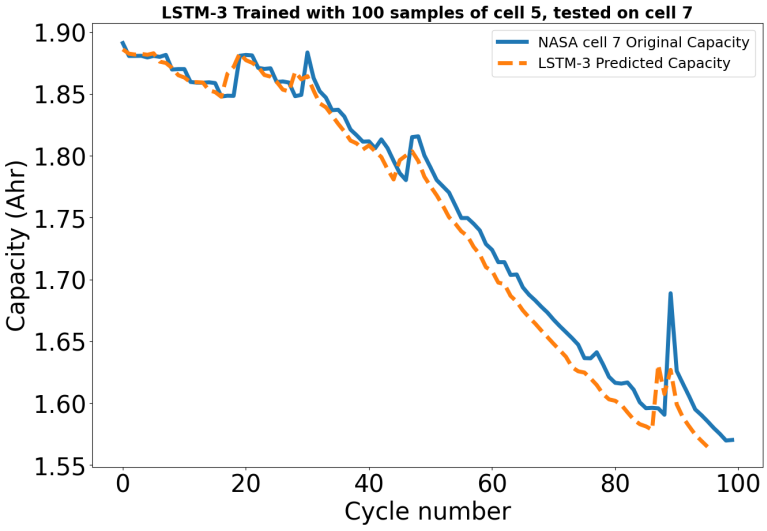

Figure 7.6: Transfer learning result on cell 6

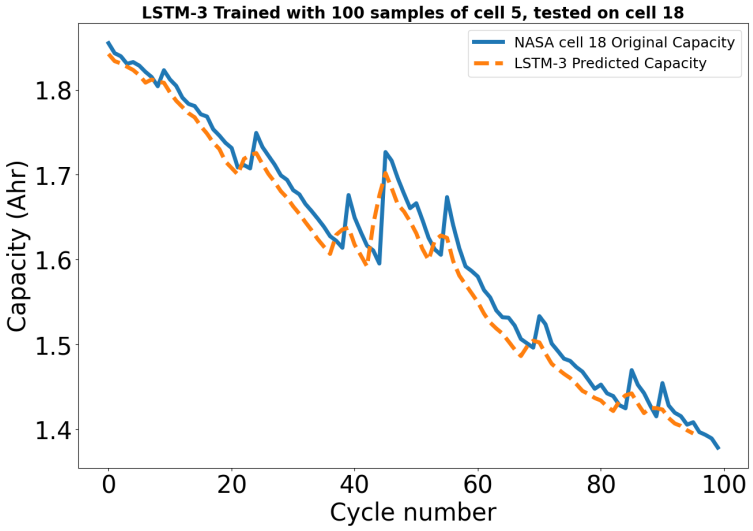

Figure 7.7: Transfer learning result on cell 6

the predicted EOL being 146. The RMSE value obtained after the prediction is 0.1685. The second experiment takes 50 values past the test position, and the predicted RUL values can be visualized through Figure 7.9. This experiment deals with various test positions, the values being 60, 80, 100, and 125. At 60, the historical data from the last 50 points, from cycles 10 to 60, are taken for training. For 80, the values from cycles 30 to 80 are taken. The results show that the results improve as we move closer to the prediction test position, and the model can predict closer to the EOL line.

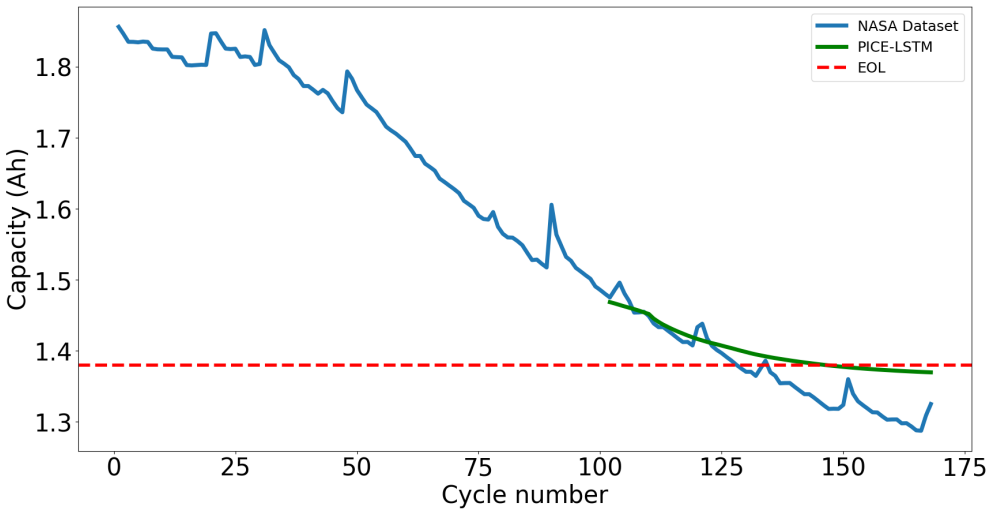

Figure 7.8: PICE-LSTM Experiment 1

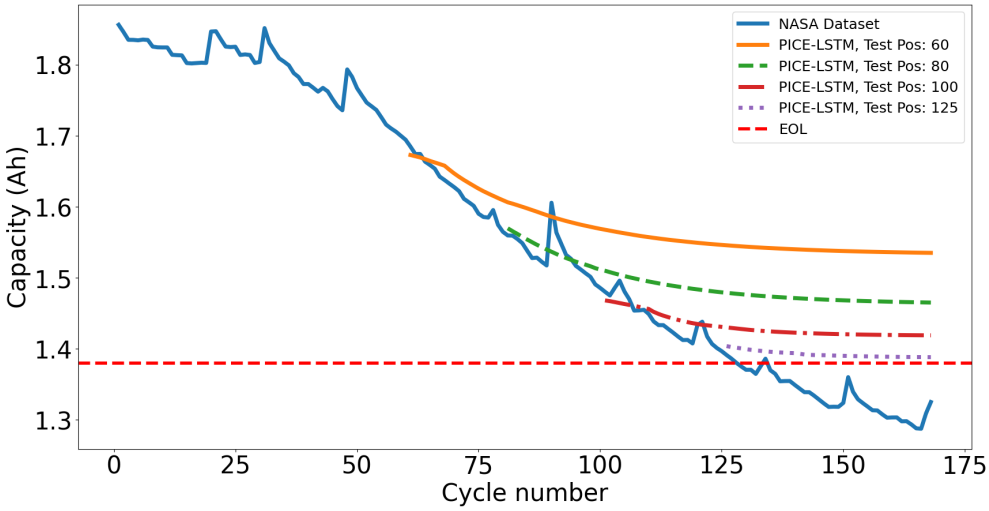

Figure 7.9: PICE-LSTM Experiment 2

Chapter 8

Conclusions

This section summarizes the results of our study on LSTM models. It sheds light on how the different proposed LSTM models perform in predicting the RUL, the efficiency the model provides, and the importance of parameter optimization. This section sheds light on how transfer learning and the proposed model, PICE-LSTM, help in RUL estimation in real-life scenarios.

First, we conclude the impact of varying parameters, such as hidden nodes and window size, on LSTM model performance. Optimal results were achieved with a window size of 14 and 150 hidden nodes, striking a balance between runtime efficiency and accuracy. Careful parameter selection is crucial for maximizing model capabilities. It is important to experiment to find the correct number of hidden nodes for practical training and the window size that provides the highest efficiency in LSTM models.

The proposed LSTM-2 and LSTM-3 models provide precise estimates of battery capacity and exhibit accuracy in predicting RUL on NASA's lithium-ion battery dataset. Notably, LSTM-2 achieves an RMSE value of 0.00982, while LSTM-3 closely follows with an RMSE of 0.0104. Both models outshine LSTM-1, which records an RMSE of 0.01193. Furthermore, LSTM-3 showcases a runtime of 6 minutes and 82 seconds, surpassing LSTM-2's 11 minutes and 41 seconds runtime. This underscores LSTM-3's effective resource utilization, making it a better choice for real-time applications and scenarios with computational constraints. PICE-LSTM focuses on multi-step prediction where the model assumes that the user can start predicting at any point of the capacity degradation cycles and have no data available at every moment. PICE-LSTM achieves an RMSE value of 0.1685 but effectively predicts the RUL with a small set of training points and requires no real-time test data for future predictions.

The outcomes of our experiments with transfer learning have demonstrated significant success, suggesting that the models trained in one battery context can be effectively tested on similar batteries, yielding commendable results.

In conclusion, this study's findings underline the importance of parameter selection and focus on accurately predicting the RUL for lithium-ion batteries for different applications using LSTM networks.

Chapter 9

Future Work

Our findings bring up several exciting directions for further investigation. One of the directions to improve the LSTM model prediction accuracy is to take into account external elements, namely charging and discharging voltage, temperature, and current values, which may improve the ability of the model to capture a more comprehensive understanding of the dynamic behaviour of battery capacity degradation. This can help better predict the RUL in batteries. This concept of multivariate LSTM can be applied to consider these multiple variables that influence the prediction.

Investigating the combination of LSTM models with other machine learning methods, such as transformer topologies, may result in hybrid models with even more improved performance. Especially for the proposed PICE-LSTM model, the prediction can be improved by adding a hybrid model consisting of neural networks, linear regression or mathematical models. The mathematical model discussed in section 3.1 can give the direction of degradation in capacity, and the LSTM network can help learn the degradation pattern. Combining these two models can help better predict future capacity values and accurate RUL.

LSTM models can be trained on massive data containing the life cycle of batteries with different cell chemistries. These types of models are generalized to many batteries. It leads to better performance when the model is asked to predict the RUL for unseen batteries whose previous historical data is not used in the training dataset.

References

[1] Khaled Sidahmed Sidahmed Alamin, Yukai Chen, Enrico Macii, Massimo Poncino, and Sara Vinco. A machine learning-based digital twin for electric vehicle battery modeling. In *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2022.

[2] Sabri Baazouzi, Niklas Feistel, Johannes Wanner, Inga Landwehr, Alexander Fill, and Kai Peter Birke. Design, properties, and manufacturing of cylindrical li-ion battery cells—a generic overview. *Batteries*, 9(6):309, 2023.

[3] Jiahao Bian, Lei Wang, Rafał Scherer, Marcin Woźniak, Pengchao Zhang, and Wei Wei. Abnormal detection of electricity consumption of user based on particle swarm optimization and long short term memory with the attention mechanism. *IEEE Access*, 9:47252–47265, 2021.

[4] George Bird and Maxim E Polivoda. Backpropagation through time for networks with long-term dependencies. *arXiv preprint arXiv:2103.15589*, 2021.

[5] Weifeng Chen, Baojia Wang, and Lorenz T Biegler. Parameter estimation with improved model prediction for over-parametrized nonlinear systems. *Computers & Chemical Engineering*, 157:107601, 2022.

[6] Hyeonwoo Cho, Changbeom Hong, Daeki Hong, Se-Kyu Oh, and Yeonsoo Kim. Thermal equivalent circuit model and parameter estimation for high-capacity li-ion cell. *Journal of The Electrochemical Society*, 170(8):080520, 2023.

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

[8] Yingzhi Cui, Jie Yang, Chunyu Du, Pengjian Zuo, Yunzhi Gao, Xinqun Cheng, Yulin Ma, and Geping Yin. Prediction model and principle of end-of-life threshold for lithium ion batteries based on open circuit voltage drifts. *Electrochimica Acta*, 255:83–91, 2017.

[9] Da Deng. Li-ion batteries: basics, progress, and challenges. *Energy Science & Engineering*, 3(5):385–418, 2015.

[10] Panagiotis Eleftheriadis, Spyridon Giazitzis, Sonia Leva, and Emanuele Ogliari. Data-driven methods for the state of charge estimation of lithium-ion batteries: An overview. *Forecasting*, 5(3):576–599, 2023.

[11] Carlos Ferreira and Gil Gonçalves. Remaining useful life prediction and challenges: A literature review on the use of machine learning methods. *Journal of Manufacturing Systems*, 63:550–562, 2022.

[12] Jian Gao, Hong Xu, Qiu-jie Li, Xiao-hai Feng, and Sha Li. Optimization of medium for one-step fermentation of inulin extract from jerusalem artichoke tubers using paenibacillus polymyxa zj-9 to produce r, r-2, 3-butanediol. *Bioresource technology*, 101(18):7076–7082, 2010.

[13] Kaidi Gao, Jingyun Xu, Zuxin Li, Zhiduan Cai, Dongming Jiang, and Aigang Zeng. A novel remaining useful life prediction method for capacity diving lithium-ion batteries. *ACS omega*, 7(30):26701–26714, 2022.

[14] Jiaojiao Hu, Xiaofeng Wang, Ying Zhang, Depeng Zhang, Meng Zhang, and Jianru Xue. Time series prediction method based on variant lstm recurrent neural network. *Neural Processing Letters*, 52:1485–1500, 2020.

[15] Siyu Jin, Xin Sui, Xinrong Huang, Shunli Wang, Remus Teodorescu, and Daniel-Ioan Stroe. Overview of machine learning methods for lithium-ion battery remaining useful lifetime prediction. *Electronics*, 10(24):3126, 2021.

[16] Pritam Khan, Priyesh Ranjan, and Sudhir Kumar. Data heterogeneity mitigation in healthcare robotic systems leveraging the nelder–mead method. In *Artificial Intelligence for Future Generation Robotics*, pages 71–82. Elsevier, 2021.

[17] Phattara Khumprom and Nita Yodo. A data-driven predictive prognostic model for lithium-ion batteries based on a deep learning algorithm. *Energies*, 12(4):660, 2019.

[18] Xin Lai, Wei Yi, Yifan Cui, Chao Qin, Xuebing Han, Tao Sun, Long Zhou, and Yuejiu Zheng. Capacity estimation of lithium-ion cells by combining model-based and data-driven methods based on a sequential extended kalman filter. *Energy*, 216:119233, 2021.

[19] Izaro Laresgoiti, Stefan Käbitz, Madeleine Ecker, and Dirk Uwe Sauer. Modeling mechanical degradation in lithium ion batteries during cycling: Solid electrolyte interphase fracture. *Journal of Power Sources*, 300:112–122, 2015.

[20] Lyu Li, Yu Peng, Yuchen Song, and Datong Liu. Lithium-ion battery remaining useful life prognostics using data-driven deep learning algorithm. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 1094–1100. IEEE, 2018.

[21] Xiaoyu Li, Zhenpo Wang, and Lei Zhang. Co-estimation of capacity and state-of-charge for lithium-ion batteries in electric vehicles. *Energy*, 174:33–44, 2019.

[22] Xin Liu, Changbo Yang, Yanmei Meng, Jihong Zhu, and Yijian Duan. Capacity estimation of li-ion battery based on transformer-adversarial discriminative domain adaptation. *AIP Advances*, 13(7), 2023.

[23] Yiwei Liu, Jing Sun, Yunlong Shang, Xiaodong Zhang, Song Ren, and Diantao Wang. A novel remaining useful life prediction method for lithium-ion battery based on long short-term memory network optimized by improved sparrow search algorithm. *Journal of Energy Storage*, 61:106645, 2023.

[24] Saifullah Mahmud, Mostafizur Rahman, Md Kamruzzaman, Md Osman Ali, Md Shariful Alam Emon, Hazera Khatun, and Md Ramjan Ali. Recent advances in lithium-ion battery materials for improved electrochemical performance: A review. *Results in Engineering*, 15:100472, 2022.

[25] Arumugam Manthiram. An outlook on lithium ion battery technology. *ACS central science*, 3(10):1063–1069, 2017.

[26] Man-Fai Ng, Jin Zhao, Qingyu Yan, Gareth J Conduit, and Zhi Wei Seh. Predicting the state of charge and health of batteries using data-driven machine learning. *Nature Machine Intelligence*, 2(3):161–170, 2020.

[27] Naoki Nitta, Feixiang Wu, Jung Tae Lee, and Gleb Yushin. Li-ion battery materials: present and future. *Materials today*, 18(5):252–264, 2015.

[28] Şaban Öztürk. *Convolutional Neural Networks for Medical Image Processing Applications*. CRC Press, 2022.

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

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

[31] T PMoseley and J Garche. Electrochemical energy storage for renewable sources and grid balancing, 2014.

[32] Cheng Qian, Binghui Xu, Liang Chang, Bo Sun, Qiang Feng, Dezhen Yang, Yi Ren, and Zili Wang. Convolutional neural network based capacity estimation using random segments of the charging curves for lithium-ion batteries. *Energy*, 227:120333, 2021.

[33] M Siva Ramkumar, C Reddy, Agenya Ramakrishnan, K Raja, S Pushpa, S Jose, Mani Jayakumar, et al. Review on li-ion battery with battery management system in electrical vehicle. *Advances in Materials Science and Engineering*, 2022, 2022.

[34] Huzaifa Rauf, Muhammad Khalid, and Naveed Arshad. Machine learning in state of health and remaining useful life estimation: Theoretical and technological development in battery degradation modelling. *Renewable and Sustainable Energy Reviews*, 156:111903, 2022.

[35] Lei Ren, Jiabao Dong, Xiaokang Wang, Zihao Meng, Li Zhao, and M Jamal Deen. A data-driven auto-cnn-lstm prediction model for lithium-ion battery remaining useful life. *IEEE Transactions on Industrial Informatics*, 17(5):3478–3487, 2020.

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

[37] Maria Kayra Saskia and Evvy Kartini. Current state of lithium ion battery components and their development. In *IOP Conference Series: Materials Science and Engineering*, volume 553, page 012058. IOP Publishing, 2019.

[38] Shipra Saxena. What is lstm? introduction to long short-term memory. *Analytics Vidhya*, January 2024. https://www.analyticsvidhya.com/blog/2024/01/what-is-lstm-introduction-to-long-short-term-memory/.

[39] Khaled Sidahmed Sidahmed Alamin, Yukai Chen, Enrico Macii, Massimo Poncino, and Sara Vinco. A machine learning-based digital twin for electric vehicle battery modeling. *arXiv e-prints*, pages arXiv–2206, 2022.

[40] Xiangbao Song, Fangfang Yang, Dong Wang, and Kwok-Leung Tsui. Combined cnn-lstm network for state-of-charge estimation of lithium-ion batteries. *Ieee Access*, 7:88894–88902, 2019.

[41] Muhammad Osama Tarar, Ijaz Haider Naqvi, Zubair Khalid, and Michal Pecht. Accurate prediction of remaining useful life for lithium-ion battery using deep neural networks with memory features. *Frontiers in Energy Research*, 11:1059701, 2023.

[42] TensorFlow. Time series forecasting with tensorflow - a comprehensive tutorial. https://www.tensorflow.org/tutorials/structured_data/time_series.

[43] Ye Tian, Chen Lu, Zili Wang, Laifa Tao, et al. Artificial fish swarm algorithm-based particle filter for li-ion battery life prediction. *Mathematical Problems in Engineering*, 2014, 2014.

[44] Yukai Tian, Jie Wen, Yanru Yang, Yuanhao Shi, and Jianchao Zeng. State-of-health prediction of lithium-ion batteries based on cnn-bilstm-am. *Batteries*, 8(10):155, 2022.

[45] Manh-Kien Tran, Manoj Mathew, Stefan Janhunen, Satyam Panchal, Kaamran Raahemifar, Roydon Fraser, and Michael Fowler. A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters. *Journal of Energy Storage*, 43:103252, 2021.

[46] University of Maryland, A. James Clark School of Engineering, CALCE. Calce battery dataset. https://calce.umd.edu/battery-data, 2024.

[47] Qiao Wang, Min Ye, Xue Cai, Dirk Uwe Sauer, and Weihan Li. Transferable data-driven capacity estimation for lithium-ion batteries with deep learning: A case study from laboratory to field applications. *Applied Energy*, 350:121747, 2023.

[48] Shunli Wang, Siyu Jin, Dan Deng, and Carlos Fernandez. A critical review of online battery remaining useful lifetime prediction methods. *Frontiers in Mechanical Engineering*, 7:719718, 2021.

[49] Zhuqing Wang, Ning Liu, Chilian Chen, and Yangming Guo. Adaptive self-attention lstm for rul prediction of lithium-ion batteries. *Information Sciences*, 635:398–413, 2023.

[50] Zhuqing Wang, Ning Liu, and Yangming Guo. Adaptive sliding window lstm nn based rul prediction for lithium-ion batteries integrating ltsa feature reconstruction. *Neurocomputing*, 466:178–189, 2021.

[51] Bolun Xu, Alexandre Oudalov, Andreas Ulbig, Göran Andersson, and Daniel S Kirschen. Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140, 2016.

[52] Asri Rizki Yuliani, Ade Ramdan, Vicky Zilvan, Ahmad Afif Supianto, Dikdik Krisnandi, Raden Sandra Yuwana, Dicky Prajitno, and Hilman Pardede. Remaining useful life prediction of lithium-ion battery based on lstm and gru. In *Proceedings of the 2021 International Conference on Computer, Control, Informatics and Its Applications*, pages 21–25, 2021.

[53] Lijun Zhang, Tuo Ji, Shihao Yu, and Guanchen Liu. Accurate prediction approach of soh for lithium-ion batteries based on lstm method. *Batteries*, 9(3):177, 2023.

[54] Shuxin Zhang, Zhitao Liu, and Hongye Su. State of health estimation for lithium-ion batteries on few-shot learning. *Energy*, 268:126726, 2023.

[55] Shuzhi Zhang, Baoyu Zhai, Xu Guo, Kaike Wang, Nian Peng, and Xiongwen Zhang. Synchronous estimation of state of health and remaining useful lifetime for lithium-ion battery using the incremental capacity and artificial neural networks. *Journal of Energy Storage*, 26:100951, 2019.

[56] Shaishai Zhao, Chaolong Zhang, and Yuanzhi Wang. Lithium-ion battery capacity and remaining useful life prediction using board learning system and long short-term memory neural network. *Journal of Energy Storage*, 52:104901, 2022.

[57] Hangxia Zhou, Yujin Zhang, Lingfan Yang, Qian Liu, Ke Yan, and Yang Du. Short-term photovoltaic power forecasting based on long short term memory neural network and attention mechanism. *Ieee Access*, 7:78063–78074, 2019.

[58] Xingyu Zhou, Xuebing Han, Yanan Wang, Languang Lu, and Minggao Ouyang. A data-driven lifepo4 battery capacity estimation method based on cloud charging data from electric vehicles. *Batteries*, 9(3):181, 2023.