DIGITAL TWIN FOR BATTERY LIFE PROGNOSIS:CLOUD BMS

A project Report

submitted by

ROHITH CHIKKALA

EE16B135

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

&

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **Digital Twin For Battery Life Prognosis**, submitted by **Rohith Chikkala (EE16B135)**, to the Indian Institute of Technology, Madras, for the award of the degree of **the degree of Bachelors of Technology and Master of Technology**, is a bonafide record of the project work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Jayaganthan
Project Guide
Professor
Department of Engineering Design
IIT-Madras, 600 036

Prof. Sarathi
Project Co-Guide
Professor
Department of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 18th June 2021

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my project advisor, Prof.Jayaganthan R, and co-guide prof.R.Sarathi for their invaluable guidance and patience.It has been a pleasant learning experience. I sincerely thank them for providing me this opportunity. Finally, I want to thank my family and friends for their constant support and encouragement.

ABSTRACT

With the improvements in electric powered vehicles,grew the want for estimation of State of health for batteries in real-time,consequently numerous strategies had been evolved for estimation of State of health.Methods like coulomb counting, open circuit voltage, kalman filer, etc. had been mentioned withinside the report. Latest traits in AI, have brought about its numerous programs and State of health estimation also can be accomplished the use of deep learning models. State of Health estimation the use of deep artificial neural network models has been mentioned in detail. Various models had been tested and compared. And a very last version has been evolved with minimum loss of 0.268.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

SOH	State of health
BMS	Battery management system
IOT	Internet of things
EC	Electro chemical
LIB	Lithium Ion Batteries

CHAPTER 1

INTRODUCTION

The battery machine has emerged as one of the maximum famous power garage machine in cellular applications. It reduces worldwide carbon emission. However, with out right tracking and controlling of the battery, troubles regarding safety,reliability,sturdiness and price will appear.
Battery Management System(BMS) includes two main modules:

- BMS master
- BMS slave

These two modules can be combined into an integrated system. An accurate health assessment provides important information about the aging rate of the battery, which can be used for maintenance or operational strategies to extend battery life.

Due to the complex non-linear mechanism of battery aging, SOH assessment is more difficult. Although electrochemical impedance spectroscopy is usually used to determine the degree of aging of the battery offline, the strict requirements for the input signal are an obstacle to its online application in BMS.
Various model-based SOH estimation methods have been developed to ensure accurate monitoring of battery conditions. However, with the increase in the number of battery cells and the complexity of the built-in algorithms, BMS is facing problems with processing performance and data storage. Use model-based algorithms to accurately estimate and predict battery status
SOH and strategy optimization is based on historical operation data of the battery cells, so they are difficult to be implemented onboard. They can be achieved by applying cloud computing and the Internet of Things (IoT) technologies.
The digital twin can show the data measured by sensors in battery and also visualize

the internal state of battery with diagnostic algorithms. Further-more, the system reliability increases with wireless IoT communication. Compared with on board BMS, digital twin for battery systems has potential benefits in aspects as follows:

• Continuous and accurate monitoring of the battery state with advanced diagnostic algorithms, supported by high computation power.

• Early detection of glitches in several levels with big data analysis, increasing the system safety and relability.

• Optimization of the system design and operation strategy by evaluating the large data from battery systems with different operations scenarios.

• The technical details of both software and hardware design of the cloud BMS were rarely introduced and therefore the functionalities of the system haven't been validated with field operation.

• Cloud-suited battery diagnostic algorithms are missing, which may improve the diagnostic accuracy and enjoy the high computation power and data storage capability of the cloud BMS.

1.1 Motivation

The adverse effects of the global climate change and greenhouse effect have pushed researchers to develop energy sources and storage devices worldwide. Among these various energy storage devices, Li-ion batteries are considered to be the main source of power for electric vehicles.The SOH is the estimate of the remaining battery capacity and an important parameter for a control strategy. So, an accurate estimation of the SOH helps in protection of the battery, prevention of over-discharge, improving the battery life and making rational control strategies to save energy.

1.2 Digital twin

The term “digital twin” first mentioned in the field of aerospace engineering, the term “digital twin” has evolved into a more general understanding of the concept, one which is notably driven by the trends in digitalization, Internet of Things (IoT)and Industry focus on the usage of digital twins in production processes and state that, depending on the field of application, the term is used and understood in a variety of ways.Digital twin is a multi-discipline simulation of a real-world product, which uses data and sensor information as input to models that mirror and predict the states and behavior over the lifespan of the physical system. In all contributions discussed the digital twin concepts incorporate a collection of data and a set of multi-discipline methods such as models, algorithms or simulations that transform the collected real-world data into knowledge.

1.3 Battery system

Due to electrification, the most costly and complex component in a car is now the energy storage, in most cases a Li-Ion battery. Li-Ion batteries, as perishable goods, show aging effects. Hence, they cannot be used for an unlimited period of time in an automobile . A widely accepted limit for the automotive battery usage is given at SOH of80%.

a) Design Layout:

Nowadays, battery systems consist of a modular structure . The battery system comprises several battery modules, which in turn consist of the actual battery cells. Since temperature is crucial for battery safety and performance, thermal systems are used to keep the cells at a reasonable working temperature. The main task of a battery management system (BMS) is to monitor and control the battery system components. Over- and under voltages as well as high temperatures during car usage and charging processes need to be avoided to keep the battery within safe operating conditions. Furthermore, the BMS controls cell-balancing and estimates the inner states of the battery, e. g.State of Charge (SOC) and SOH.

b) Production:

Producing battery systems is also divided up according to the system’s modular struc-

ture. As a first step, the battery cells are manufactured. In the next layer,several cells with the same performance characteristics are brought together and form a homogeneous module. In the system layer, multiple battery modules are fit into the system case and connected electrically. After integrating the BMS and the thermo system, the HVBS is sealed and closed with alid. After assembly, the entire system undergoes an end-of-line test, where visual checks and technical test routines are performed.

1.4 Objectives

Our goal is to develop a cloud battery management system which can estimate/predict remaining life of a battery. battery life is estimated using a DNN model using historical data of the battery. The prognostic process involves two phases. The first phase of prognostics aims to assess the current health status or state of health (SoH). Terms that are usually used to describe this phase in most of the literature are severity detection and degradation detection, which can also be considered under diagnostics. Classification or clustering techniques can be utilized to perform tasks such as model training ,data cleaning in this phase. The second phase aims to predict the failure time by forecasting the degradation trend,and by identifying the remaining useful life (RUL). Trend projection, tracking techniques, or time series analysis are included in this phase. Most of the academic articles regarding prognostics analysis only consider the first phase . This paper aims to construct and analyze both SoH and RUL, in which focus is made on both the first and second phases of prognostics for the battery system

1.5 literature survey

Generally, there are two existing major approaches for prognostics evaluation; the data-driven model, and physics-based models. Data-driven methods require adequate data or samples from systems that were run until failure, while physics-based methods evaluate the system's failures via the physics of failure progression. Both the data-driven and physics-based model also have different requirements and use cases, and both also have different advantages and drawback as well. There have been many ad-

vancements contributed by researchers from various disciplines to PHM of lithium-ion batteries. Downey et al.proposed a physics-based prognostic approach that considered multiple concurrent degradation mechanisms [6]. Susilo et al. studied the estimation of the lithium-ion battery SoH with the combination of Gaussian distribution data and the least square support vector machines regression approach [7]. Bai et al.developed a generic model-free approach based on ANN and the Kalman filter,to help to improve the health management system of the lithium-ion battery [8]. Other filtering techniques,for example,particle filtering [9] or its variation of the unscented particle filtering technique [10] had been employed in the PHM aspect for lithium-ion batteries.Recently, Li et al.A data-driven model based on the deep learning approach for lithium-ion battery prognosticsis the main focus of this paper. Although various approaches had been proposed to improve the PHM prediction of lithium-ion batteries, the deep learning approach for PHM is still limited. The advancement of computational tools and big data algorithms have largely impacted the development of this approach. The machine learning algorithms,in particular,ANN, have been proven to be able to empirically learn and recognize the more complex patterns of the system's data in many applications. This feature of machine learning algorithms also benefits prognostic analysis modeling as well.This paper presents the preliminary development of a data-driven model using Deep Neural Networks (DNN) to predict the SoH and RUL of lithium-ion batteries.DNN is a deep learning approach that was developed based on Artificial Neural Networks with multiple hidden layers,to analyze more complex data and features. Although some deep learning algorithms, such as Recurrent Neural Network (RNN) and Long Short-Term Memory Network (LSTM),are employed to model prognostic of lithium-ion battery recently, to date,there is no work that has employed a DNN model to perform similar tasks.In addition, there are limited works that have performed a deep learning approach against other data-driven algorithms.The effectiveness of the proposed approach was tested in the lithium-ion battery dataset derived from the NASA Ames Prognostics Center of Excellence (PCoE). A DNN approach was employed to predict the SoH and RUL and the results were compared against other machine learning algorithms such as Linear Regression (LR), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), and ANN. This paper is constructed with the following sections: Section 2 discusses the overview of the Cloud BMS application and the characteristics of the lithium-ion battery used in this paper, Section

3 provides a concise literature review of the proposed approach for DNN analysis and modeling, Section 4 details the experimental results and the comparison of DNN and other machine learning algorithms, and Section 5 concludes the findings and investigates possible future work.

1.5.1 Data Driven

Based on The empirical lifetime data and the use of previous data of the operation of the system Physical understanding of the physical rules of the system, the exact formulas that represent the system Advantages The real behavior of the complex physical system is not required.One of the data-driven model approaches for prognostics and diagnostics mentioned earlier are machine learning approaches, which will be the main discussion topic of this paper.

Advantages :

Data driven models are less complex, easier to employ into a real application The model represents a real system, the model can be observed and judged in a more realistic manner.

Drawbacks:

The data driven models do not represent the actual system, it requires more effort to understand the real system behavior based on the collected data .

1.5.2 Physicial models

Model is based on physical understanding of the physical rules of the system, the exact formulas that represent the system

Advantages:

Higher accuracy because the model is based on an actual (or near-actual) physical system. The model represents a real system, the model can be observed and judged in a more realistic manner.Highly complex, requires extensive computational time/resources, which may not be very suitable for employment in real-world applications

Drawbacks:

Limitations in modeling, especially in cases of large and complex systems with non-

measurable variables

1.5.3 Prognostics and health managment

The PHM of the battery has to be included as part of the condition-based maintenance (CBM) plan of the system. The CBM plan is considered as a preventive strategy, which means that maintenance tasks will be performed only when need arises. This need can be determined by continuously evaluating health status of a particular system's components,or the health state of the system as a whole [11]. CBM has included two major tasks: diagnostics and prognostics. Diagnostics is the process of the identification of faults and part of the current health status of the system, which is described as an SoH, whereas prognostics is the process of forecasting the time to failure. The time left before observing a failure is described as the remaining useful life (RUL) of such a system [12]. To avoid severe negative consequences when systems run until failure, the maintenances must be performed when the system is still up and running. These type of maintenance require early plans and preparation [13]. Thus, CBM must properly be included as part of the system's operation, especially for the critical systems. The prognostic of the system is a crucial factorin CBM.

CHAPTER 2

Cloud battery management system

By connecting the physical and virtual worlds, the digital twin enables the virtual unit to access the battery system while seamlessly transferring data. In terms of hardware, cloud BMS has powerful computing power, huge storage capacity and high system reliability. These functions also support the use of advanced algorithms in the software. The performance of the functions that already exist in the embedded BMS can be further improved by more advanced algorithms. On the other hand, there are new functions such as life-based data prediction and system optimization. What is difficult to implement in an embedded BMS can be deployed in the cloud. This section introduces the principles of cloud BMS and the functions of subsystems.

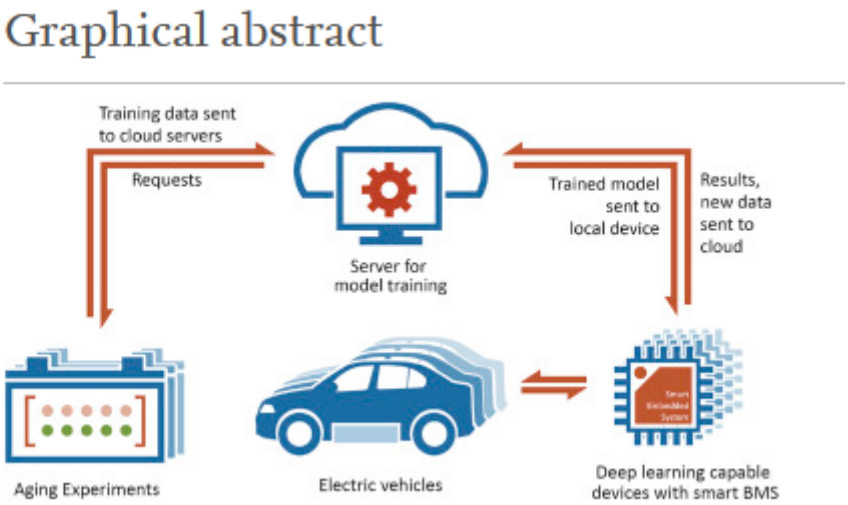


Figure 2.1: Digital twin model

2.1 Cloud

Since IoT devices usually have limited computing and storage capabilities and do not allow complex data processing, cloud computing provides scalable real-time analysis of IoT device data with almost unlimited storage and processing capabilities. It consists of a data logger and a database. The data logger collects a large amount of unstructured or semi-structured data generated by the battery system, and provides a secure gateway and data transmission to a widely certified distributed cloud database. Prevent denial of service (DDoS) and multiple redundant connections. Only the owner and operator of the battery system can access the data; the confidentiality and security of the data are guaranteed.

2.2 BMS Slave

The BMS-Slave performs data acquisition by measuring the voltage, current and temperature of the battery cells with sensors at different sampling rates.While BMS slave is responsible for monitoring battery cells with signal acquisition and filtering, more advanced functions such as battery diagnosis algorithms, which require high computation power, are implemented in BMS-master. However, with increase in number of cells and the scale of battery systems, the reliability and cost of wiring communication are becoming challenging.

The fundamental components withinside the BMS-Slave for information sensing are multi cell battery monitors LTC6804G-2 and LPC11C14F/301. The LTC measures as much as 12 battery cells connected in series with a complete measurement error of much less than 1.2mV, which also can be related in collection to screen high-voltage battery strings. The LPC is a 32-bit ARM Cortex-M0 primarily based totally micro controller, which sends the measurement instructions to the LTC and gets the measured information in hexadecimal values from the LTC.

2.3 IOT Components

The idea behind IOT is to make the devices, mobile battery systems embedded with electronics and connected with the cloud, communicate with others to be monitored. Therefore, a stable internet connection is important for a stable real-time data transfer between the battery system and the digital twin.

Functions that require historical operating data can only be performed in the cloud. The digital twin communicates with the battery system and updates model parameters to improve system performance and slow down battery degradation. Taking into account the advantages of the Raspberry Pi as a single-board computer, the price and weight are relatively low, and it is responsible for collecting battery data and communicating with battery systems such as cloud platforms. The measurement data from the battery cell is sent to the IoT component. Based on BMS slave controller. Local Area Network Protocol (CAN). The software is designed and implemented using Python in IoT components to convert CAN signals into physical values. In addition, the IoT component is also responsible for sending the generated data to the cloud through TCP/IP and Message Queuing Telemetry Transport Protocol (MQTT), and ensuring security and data protection.

CHAPTER 3

SOH Estimation

The SOH of a battery could be expressed as the ratio of maximum available capacity in the present condition to the nominal capacity of the battery in a fresh condition. This can be stated as

$$SOH = \frac{Q_{Present}}{Q_{Fresh}} \cdot 100\%$$

Figure 3.1: Soh formula

where $Q_{Present}$ denotes present available capacity of the battery and Q_{Fresh} indicates the capacity of the battery in the fresh condition.Below 80% of its original capacity, the battery is considered unusable Used for application purposes. This is due to exponential degradation of the battery capacity exhibited below 80% cut-off. There are numerous inner and outside factors the health of the LIB and its overall performance degradation over a length of time. Some of the inner elements consists of battery material, calendar ageing and increase in inner resistance. The outside factors are operating temperature, uncertain using circumstance and overcharging/discharging. Due to many unknown and unpredictable factors influencing the health of the battery, estimating battery SOH will become pretty challenging. These factors are liable for the unpredictable battery ageing process. The fundamental source of ageing mechanisms consists of chemical, mechanical, thermal and enormously relies upon on electrode compositions. aging Occurs in two types :

- Calendar ageing
- Cycle ageing

3.1 Calendar ageing

Due to Storage situations the calendar ageing quickens ageing of battery and it will increase inner resistance and self-discharge rate. This form of ageing is based totally on outside environmental elements which includes temperature beneathneath storage condition.

3.2 Cycle ageing

Cycle ageing takes place during charging or discharging conditions. It is mostly influenced by battery-charging methods and discharge rate. In addition, cycle ageing may result cycle-life loss in the LIB due to irreversible damage caused to electrodes under improper battery operation.

3.3 Different methods to estimate battery SOH

Most of the prevailing strategies make use of a capacity fading and electrochemical (EC) model to measure SOH due to the fact, as battery age capacity decreases, EC parameters are modified. However, a majority of these strategies estimate SOH below sturdy assumptions and static cycling condition. These conditions aren't appropriate for real world EV batteries due to the fact this requires real-time SOH computation and charging/discharging takes place in a dynamic manner. Therefore, we will adapt machine learning algorithms for estimation of battery SOH.

3.3.1 Direct methods

- Coulomb counting
- Open circuit voltage
- Electro chemical impedance

Direct measurement is a widely known approach used for estimation of battery SOH through at once measuring parameters like internal resistance. Internal

resistance is inversely proportional to SOH of a battery/capability of a battery. Direct strategies are mostly primarily based totally on measuring internal resistance of a battery the use of ohms law, additionally through measuring electro-chemical impedance value as function of frequency SOH of a battery may be estimated. And through tracking fee and discharge of current(Ah counting) it may be estimated. Direct strategies are commonly completed in labs with lot of checking out equipment and those are carried in static conditions/non-dynamic situations

3.3.2 MODEL BASED

- Electro chemical model
- Equivalent circuit model

In this approach SOH of a battery is measured/estimated by developing equivalent circuit models(ECM) and developing ECHM models using basic electrical components such as resistor, capacitor and voltage source. The models are used to determine some parameters such as voltage ,temperature,current. These experiments generate huge data set, they need high computation cost and these tests are mostly not reliable

3.3.3 ADAPTIVE FILTERS

- Kalman filter
- Particle filter
- Least square

In this method battery parameters are monitored continuously which are sensitive to aging of battery. kalman filter process consists of two process. First, prediction stage filters are designed to estimate current battery parameters then update stage estimation is estimated with current battery parameters for precise output.

3.3.4 DATA DRIVEN

- Fuzzy logic
- Artificial neural networks
- Support vector machine

3.4 Data driven SOH estimation

EV batteries discharges dynamically based on driving behavior, traffic conditions and other external factors. To eliminate such limitations, this research work is a data-driven technique to predict SOH based on vital data such as voltage (V), Capacity (C)and temperature (T) collected from LIBs under variable load conditions. An artificial neural network(ANN) framework is developed to establish the non-linear relationship among battery parameters(V, C, T) and SOH.

•A new approach is adopted to identify and extract a set of input features associated with the battery degradation process.

•A data-driven based SOH estimation using an independently recurrent neural network (RNN)is developed with the help of vital battery parameters collected from LIBs.

3.5 Developing model

This model is developed on google-collab platform with 12GB gpu access for training machine learning models using python language with following libraries:

- Tensorflow 2.0
- Numpy
- Pandas
- Scipy
- Sci-kit learn
- Matplot

3.6 DATA-SET

To train the Deep Learning model proposed, it is necessary to collect the large amount data related to the discharge of the battery, the data used for developing this model is collected from " NASA Ames Prognostics Data Repository" . To extract this data from the ".mat" file and store it in pandas DataFrames for later access, we have used the load_data function mentioned below. After loading the dataset, a description of the data is made using panda functions to verify if the data loading was correct.

```
def load_data(battery):
 mat = loadmat('/content/drive/MyDrive/data/' + battery + '.mat')
 print('Total data in dataset: ', len(mat[battery][0, 0]['cycle']
 counter = 0
 dataset = []
 capacity_data = []
 for i in range(len(mat[battery][0, 0]['cycle'][0])):
   row = mat[battery][0, 0]['cycle'][0, i]
   if row['type'][0] == 'discharge':
     ambient_temperature = row['ambient_temperature'][0][0]
     date_time = datetime.datetime(int(row['time'][0][0]),
                                   int(row['time'][0][1]),
                                   int(row['time'][0][2]),
                                   int(row['time'][0][3]),
                                   int(row['time'][0][4])) + datetime.
     data = row['data']
     capacity = data[0][0]['Capacity'][0][0]
     for j in range(len(data[0][0]['Voltage_measured'][0])):
       voltage_measured = data[0][0]['Voltage_measured'][0][j]
       current_measured = data[0][0]['Current_measured'][0][j]
       temperature_measured = data[0][0]['Temperature_measured'][
       current_load = data[0][0]['Current_load'][0][j]
       voltage_l oad = data[0][0]['Voltage_load'][0][j]
       time = data[0][0]['Time'][0][j]
       dataset.append([counter + 1, ambient_temperature, date_tim
                       voltage_measured, current_measured,
                       temperature_measured, current_load,
                       voltage_load, time])
     capacity_data.append([counter + 1, ambient_temperature, date
     counter = counter + 1
 print(dataset[0])
 return [pd.DataFrame(data=dataset,
                      columns=['cycle', 'ambient_temperature', 'd
                               'capacity', 'voltage_measured',
                               'current_measured', 'temperature_m
                               'current_load', 'voltage_load', 't
         pd.DataFrame(data=capacity_data,
                      columns=['cycle', 'ambient_temperature', 'd
                               'capacity'])]
```

3.7 Pre-processing the data

The data-set contains following attributes:
voltage, current,battery, temperature, vector time: the date and time of the start of the cycle, capacity, current load,voltage load.
Above data is used as input for the deep network model. This data is reduced to 3 inputs which are time, capacity, temperature and 1 output value which is SOH value for the given inputs.combining all cycles drive together , the whole data has more than 5,00,000 data points which can be used in training and testing model. This data has been normalized between a range of 0-1.

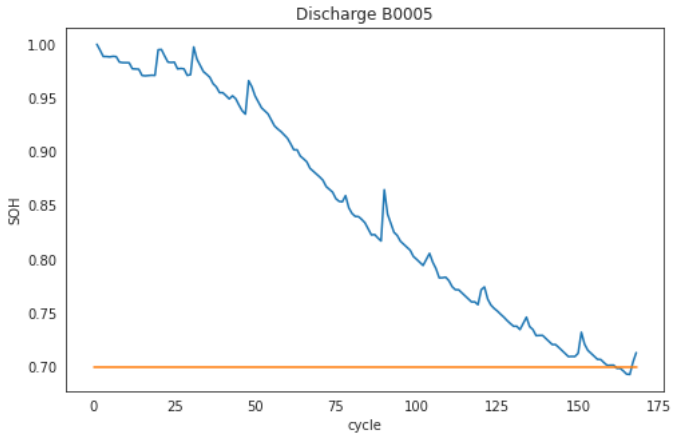


Figure 3.2: actualSOH value

Total data in dataset: 616

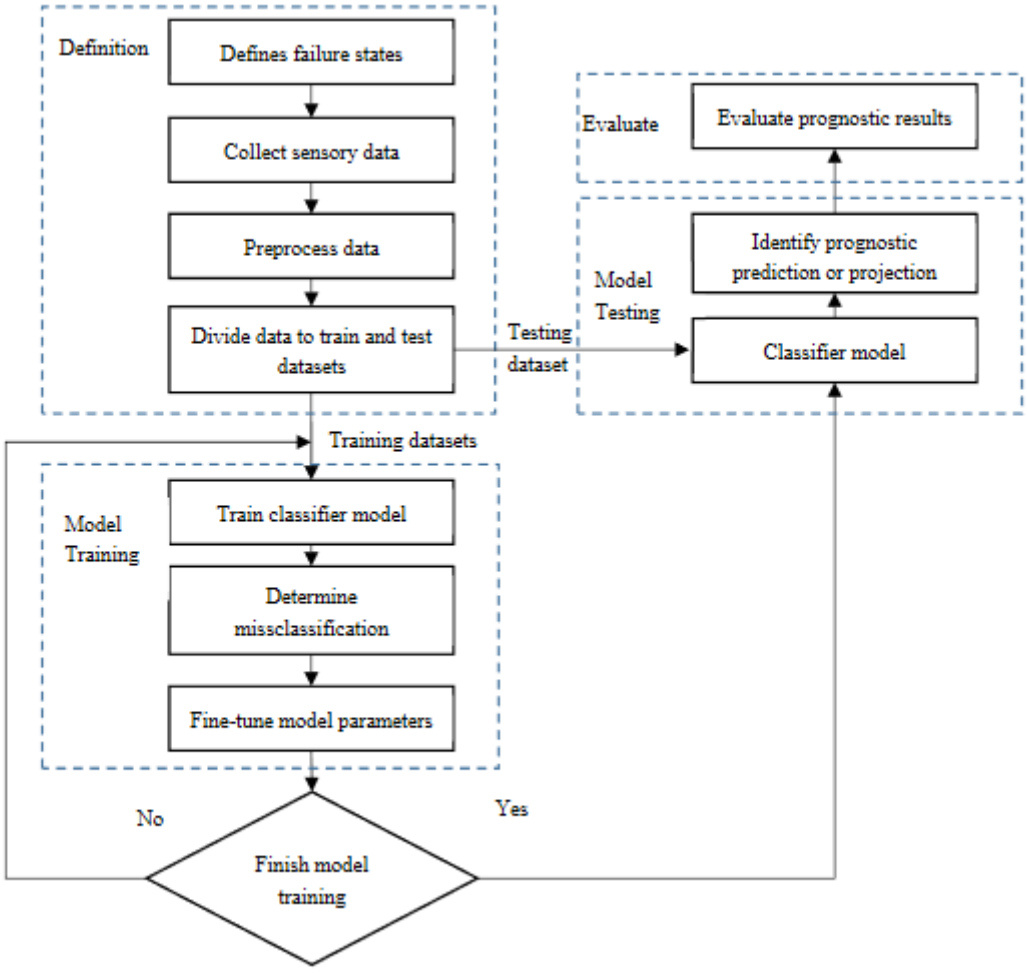
	cycle	ambient_temperature	datetime	capacity	voltage_measured	\
0	1	24	2008-04-02 15:25:41	1.856487	4.191492	
1	1	24	2008-04-02 15:25:41	1.856487	4.190749	
2	1	24	2008-04-02 15:25:41	1.856487	3.974871	
3	1	24	2008-04-02 15:25:41	1.856487	3.951717	
4	1	24	2008-04-02 15:25:41	1.856487	3.934352	

	current_measured	temperature_measured	current_load	voltage_load	time
0	-0.004902	24.330034	-0.0006	0.000	0.000
1	-0.001478	24.325993	-0.0006	4.206	16.781
2	-2.012528	24.389085	-1.9982	3.062	35.703
3	-2.013979	24.544752	-1.9982	3.030	53.781
4	-2.011144	24.731385	-1.9982	3.011	71.922

Figure 3.3: data extracted from dataset

3.8 DNN model

RNN is a category of deep neural community, which has precise capabilities called inner cell state or memory to observe input capabilities and time dependencies from sequential enter data to predict the estimate output. Therefore, the output from every neuron now no longer relies upon on current input however additionally the records of previous hidden state outputs. Depending upon the wide variety of time steps, RNN can efficaciously preserve information about the past. In addition, RNN can be greater appropriate for time-collection forecasting packages like SOH estimation, which are expecting their present output primarily based totally on a records of preceding output and present input over a time. But, RNN and its version LSTM is without problems inclined to long-variety dependency problem, which vanishes gradient in the course of again propagation operation. In version preparation 3 dense layers and one dropout layer(to prevent network from over-fitting) are used. As an optimizer, Adam optimizer is capable of lowering the mistake rate variation among real and expected value for each iteration,consequently it's miles used as an optimizer. Adam is in reality efficient while we're operating with huge dataset



1.Definition states phase. This phase specifically focuses on defining the failure of the system, identifying the prognostic problem, and evaluating system health states.
2.Pre-processing phase. In this phase, sensory data are collected according to the predefined health state,in order to build a raw dataset for the experiment. The raw datasets are pre-processed and normalized, and then divided into a training and a testing dataset.
3.Training phase. In this phase, initial parameters are developed,and the classification model is trained by the training dataset,based on deep learning theory. It is particularly important to fine-tune the classification model through misclassification errors (such as RSME).
4.Testing phase. In this phase, the testing dataset is put into the trained classification model to identify prognostic predictions or projection results.
5.Evaluating phase. This phase mainly finishes with computing the accuracy, reportin gon, and evaluating the diagnosis results from the final model.
The prognostic model of the battery data using DNN was developed based on the aforementioned framework. The experiment with the data was constructed by varying the number of dense layers in DNN. In this experiment, the hidden layer was varied to analyze the SoH of battery data until it delivered the best RMSE results. In addition, the dropout layer was also applied as the last layer before the output layer,to prevent the over-fitting problem. The dropout layer applied to the last layer of DNN,to randomly drop neurons during the model training. Each neuron is retained with a fixed probability,p which is independent of other neurons. The neural network after being sampled, the so-called “thinned” network, will contain only the surviving neurons . By training a neural network with some dropouts, the whole network can be trained more often than training regular networks without dropout,because the network is thinned so that it can be trained at less frequency. The network then becomes less sensitive to some specific weights. This results in the network being better at generalization. In this work,a p-value of 0.25 is applied to the network,as suggested,to be the optimal dropout rate for the network to avoid over-fitting,but to still maintain the best prediction accuracy

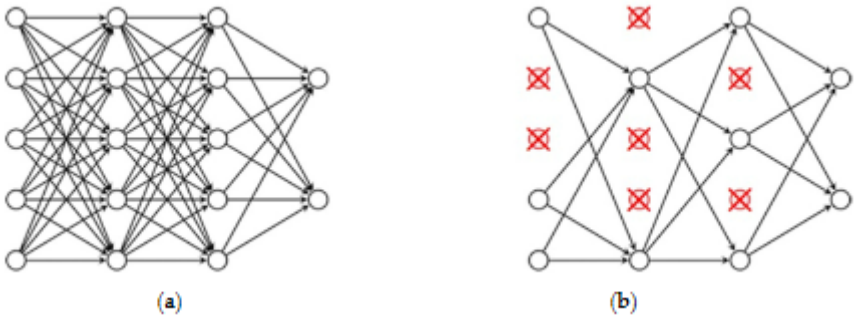


Figure 3.4: a)network without dropout layer b) network with dropout layer

no.of hidden layers	Rmse
2	0.046
3	0.037
4	0.041

Table 3.1: RMSE alue wrt layers

3.9 Model Training

To train the model 5028 samples and 50 epochs(iterations) are used. After 50 epochs the training would be completed and the model starts to predict/estimate SOH of battery

```
   cycle            datetime  capacity       SoH
0      1 2008-04-02 15:25:41  1.856487  1.000000
1      2 2008-04-02 19:43:48  1.846327  0.994527
2      3 2008-04-03 00:01:06  1.835349  0.988614
3      4 2008-04-03 04:16:37  1.835263  0.988567
4      5 2008-04-03 08:33:25  1.834646  0.988235
(50285, 7)
(50285, 1)
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 8)                 64
_________________________________________________________________
dense_11 (Dense)             (None, 8)                 72
_________________________________________________________________
dense_12 (Dense)             (None, 8)                 72
_________________________________________________________________
dropout_10 (Dropout)         (None, 8)                 0
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 9
=================================================================
Total params: 217
Trainable params: 217
Non-trainable params: 0
```

Figure 3.5: model training

3.10 Model Testing

For training, 70% of data is used and rest of the data is used for testing the model. The accuracy of the model decreases with increase in epochs, this is due to over fitting of model, if we have high training accuracy then the validation accuracy will decrease because model will not be able to generalize well. To overcome these problems we are adding a dropout layer

```
   cycle            datetime  capacity       SoH
0      1 2008-04-02 15:25:41  2.035338  1.000000
1      2 2008-04-02 19:43:48  2.025140  0.994990
2      3 2008-04-03 00:01:06  2.013326  0.989185
3      4 2008-04-03 04:16:37  2.013285  0.989165
4      5 2008-04-03 08:33:25  2.000528  0.982898
/usr/local/lib/python3.7/dist-packages/tensorflow/py
  warnings.warn('`Model.state_updates` will be remov
(50285, 1)
   cycle       SoH    NewSoH
0      1  1.000000  0.960147
1      2  0.994990  0.957485
2      3  0.989185  0.954390
3      4  0.989165  0.954376
4      5  0.982898  0.951027
5      6  0.989467  0.954535
6      7  0.989075  0.954328
7      8  0.967304  0.942716
8      9  0.966997  0.942549
9     10  0.961625  0.939679
Root Mean Square Error:  0.0921229098432579
Total data in dataset:  616
[1, 24, datetime.datetime(2008, 4, 2, 15, 25, 41), 1
Model: "sequential_1"
```

Figure 3.6: Testing Data

CHAPTER 4

RESULTS

4.1 Predicted SOH vs Actual SOH

This plot shows the comparison between actual SOH value and predicted SOH value. Because of the model training, there would be delay in SOH prediction. The red line indicates the predicted value and blue line indicates actual SOH derived from the data set and the dotted line indicates threshold value after which battery is completely degraded.

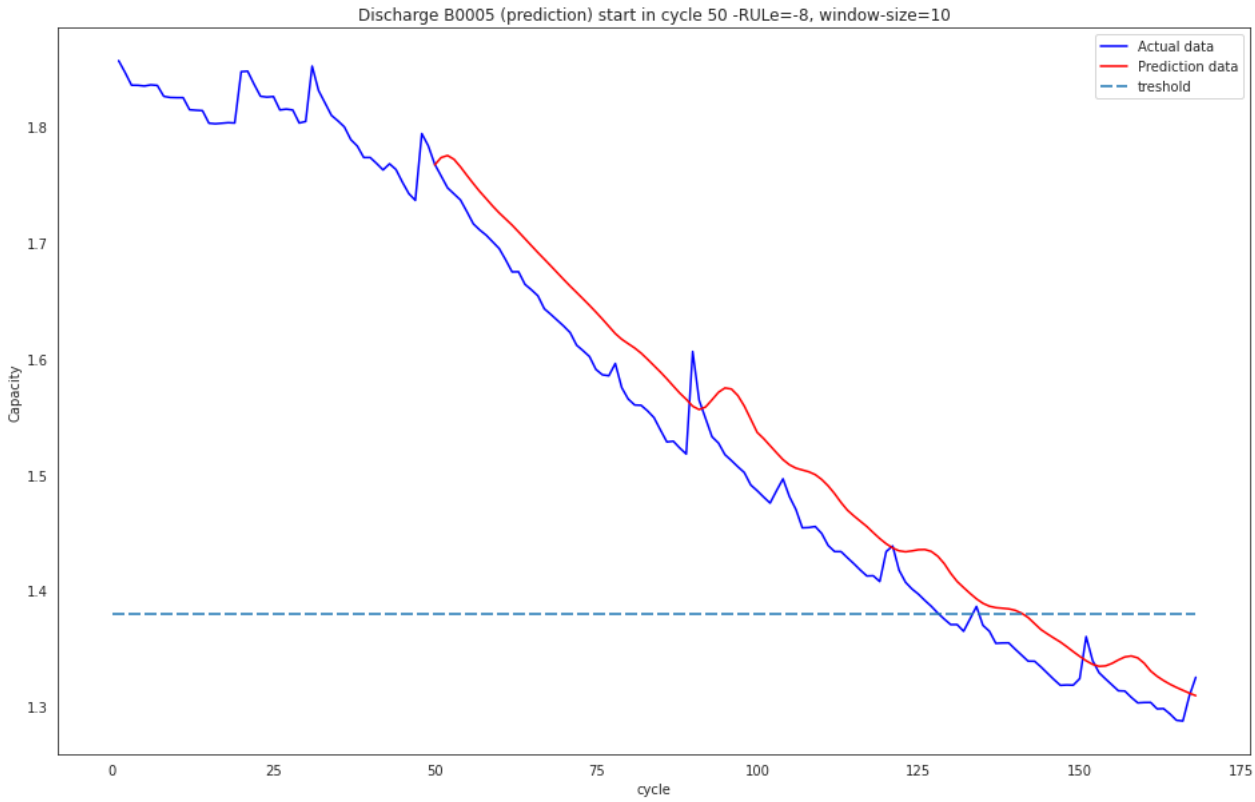


Figure 4.1: SOH Prediction

After training, the model is tested using 200 epochs. Our model predicts/estimates SOH with an error of 13cycles.

```
Test RMSE: 0.037
The Actual fail at cycle number: 128
The prediction fail at cycle number: 141
The error of RUL= 13 Cycle(s)
```

Figure 4.2: Prediction error

To check the model and validate the accuracy we will use different dataset. Estimation of B0006 model:

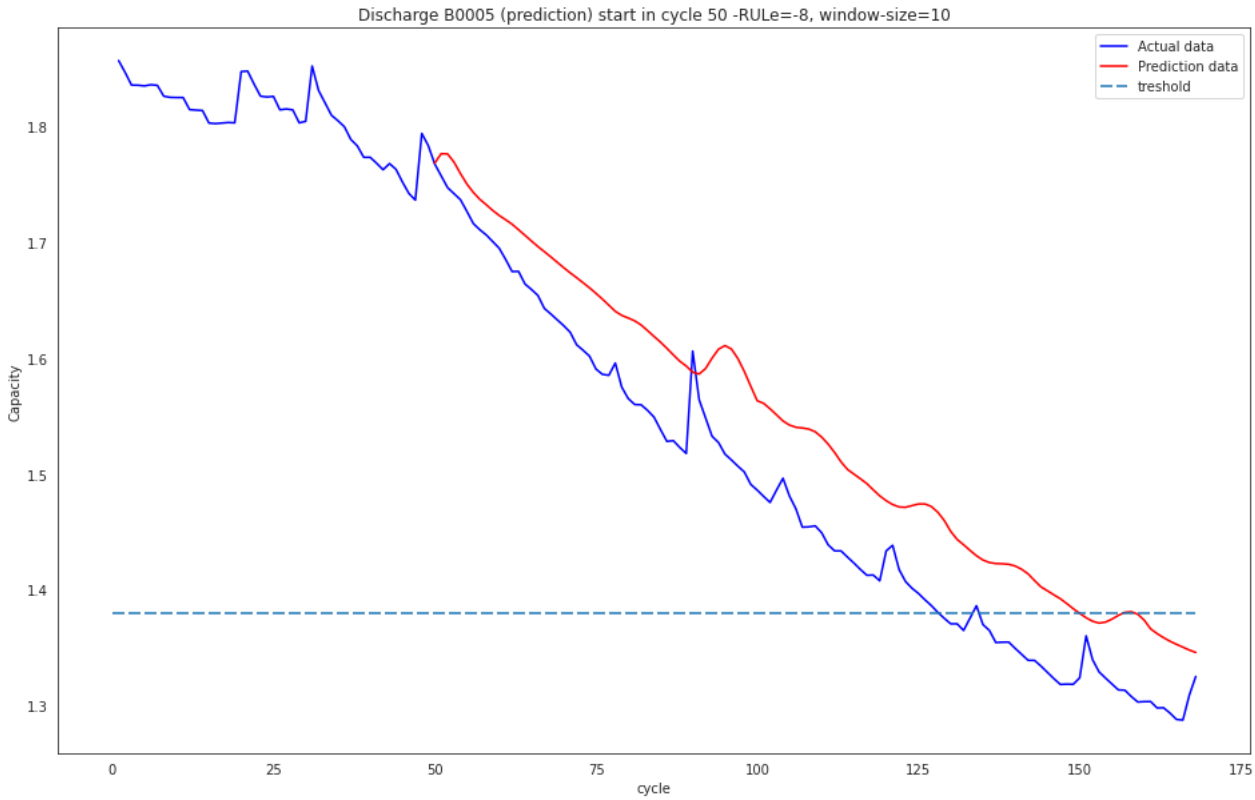


Figure 4.3: RMSE value

For the given dataset, RMSE value is 0.064 with an error of 21 cycles

```
Test RMSE: 0.064
The Actual fail at cycle number: 128
The prediction fail at cycle number: 149
The error of RUL= 21 Cycle(s)
```

Figure 4.4: SOH Estimation

This model works for different data-sets, B0005 data-set have lesser RMSE value which indicates prediction/estimation was more accurate. Final specs of the model:

ADAM optimization

Model: Sequential

Loss : Mean Squared Error

ReLu activation.

Complete python code has been attached at the end of this paper for reference

4.2 Testing model with different data

We will train the model with data-set(B0005) and test the model with data-set(B0006). By doing this process, we can confirm that model works with real time value(feeding the trained model with realtime data).

temperature_measured = data[0][0]['Temperature_measu voltage_load = data[0][0]['Voltage_load'][0][j]red'][0][j]

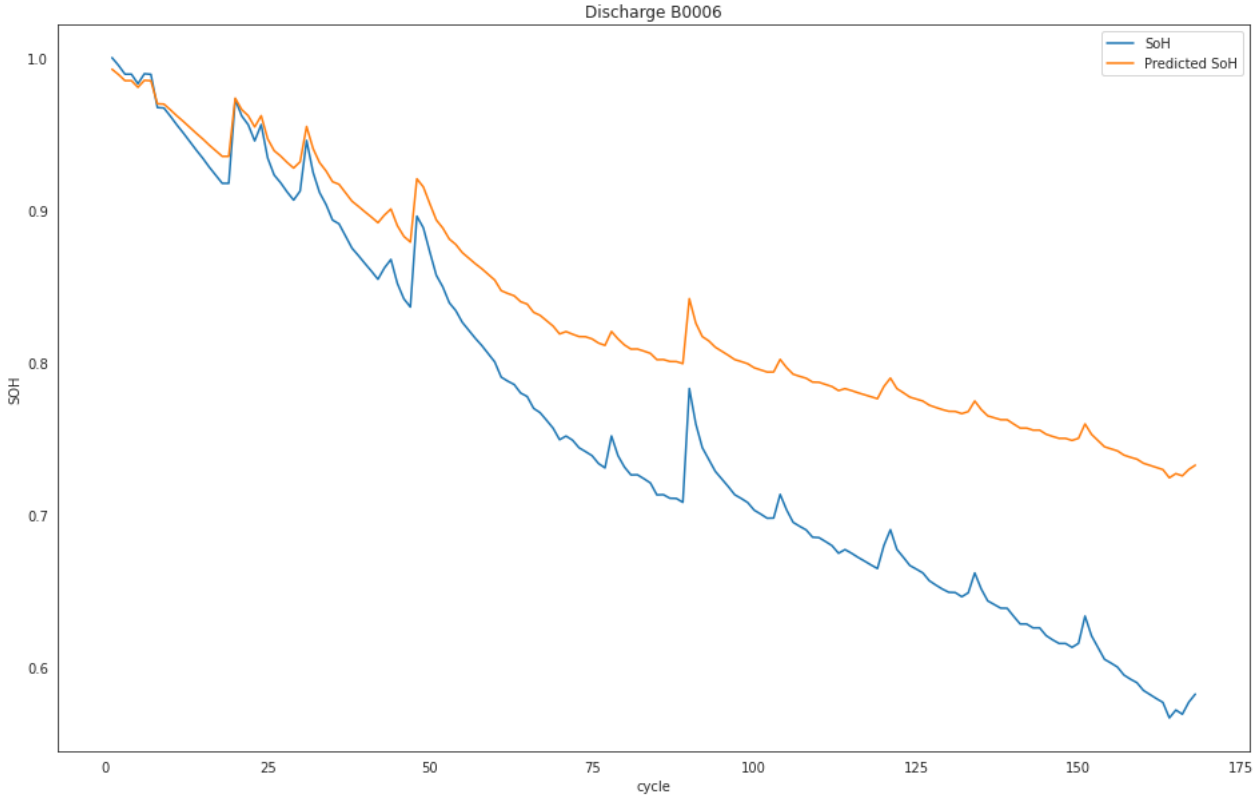


Figure 4.5: B6 SOH estimation with B5 trained model

The above graph shows the SOH estimation of B6 data set with B5 trained model

4.3 Test result with sgd optimier

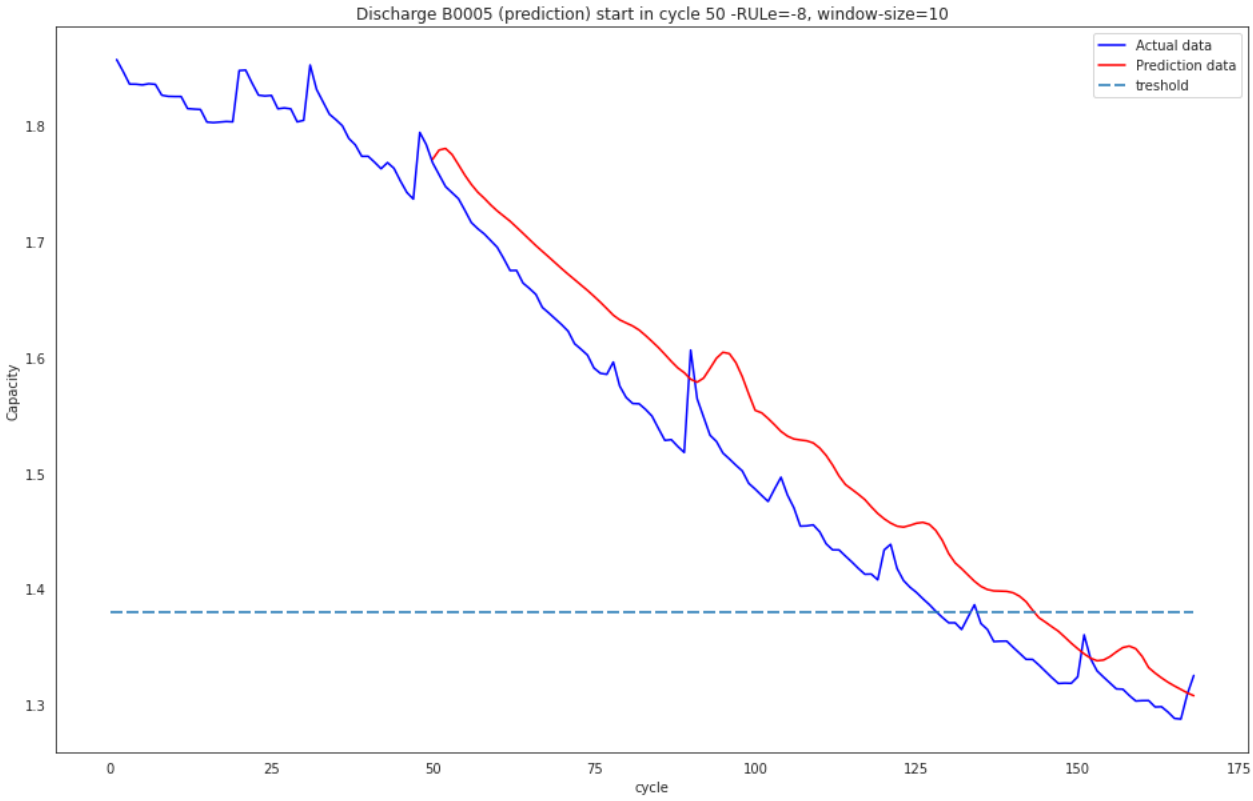


Figure 4.6: SGD optimizer estimation

```
Test RMSE: 0.051
The Actual fail at cycle number: 128
The prediction fail at cycle number: 143
The error of RUL= 15 Cycle(s)
/usr/local/lib/python3.7/dist-packages/ipykernel_laur
A value is trying to be set on a copy of a slice fror
```

4.4 Test result with RMSProp optimizer

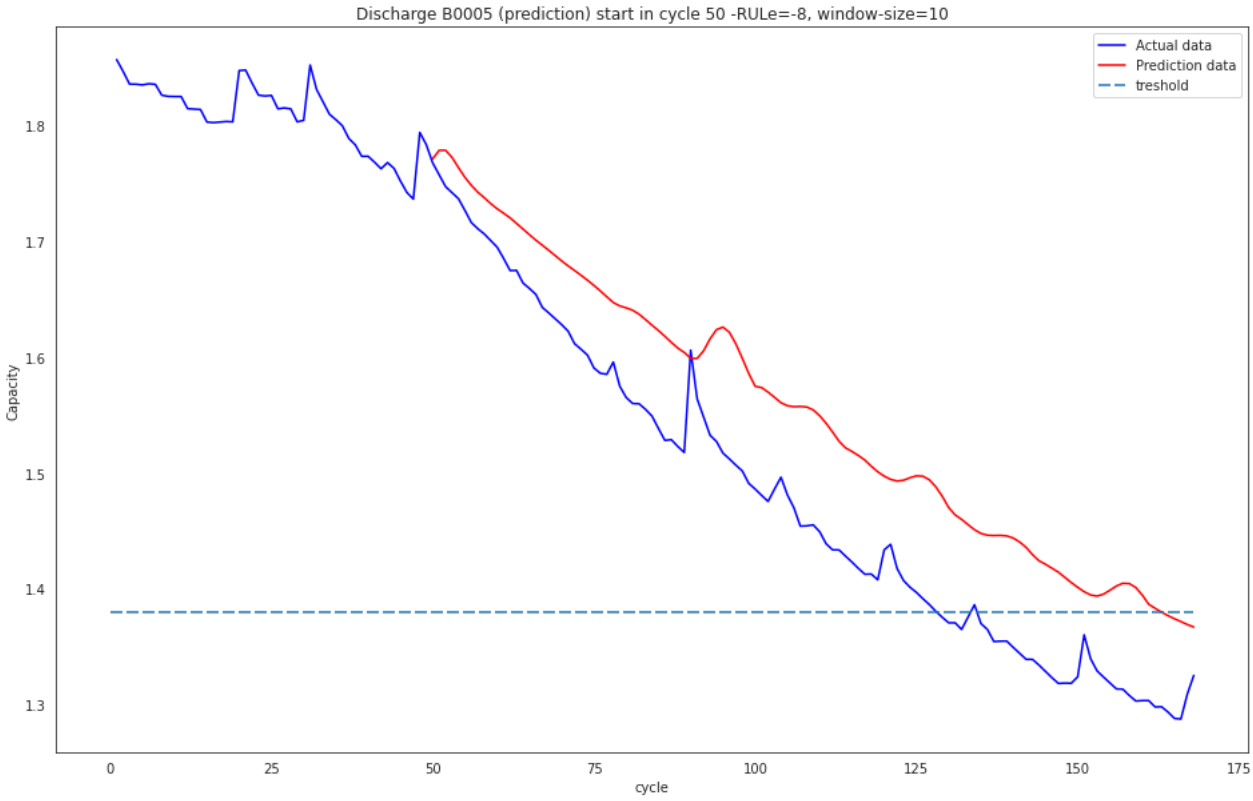


Figure 4.7: soh estimation using rmsprop optimizer

```
0.0063003697448390465
Test RMSE: 0.079
The Actual fail at cycle number: 128
The prediction fail at cycle number: 162
The error of RUL= 34 Cycle(s)
```

4.5 Comparative analysis with other research papers

SOH method	Approach used	SOH error%	Reference
OCV	Parameter varying approach (PVA)	0.5% to 3%	[11]
cc	Linear function andoperations	1.08%	[12]
KF	Forgetting factorrecursive least square	1.52%	[14]
LS	Ordinary leastsquares and total leastsquares	<5%	[15]
SVM	Grid search method,Radial basis function	<2%	[16]
ANN	Multi layer perceptron(MLP)	<1.5%	[17]
ECM	Incremental capacity analysis based model	NASA dataset:–0.57	[13]

Table 4.1: Caption

our model has lesser error percentage when compared with other papers.When model is compared with other models which have same parameters but different optimizer(sgd,rmsprop,adam) adam optimizer gives us best results. This model can used to predict SOH in our proposed cloud bms system.

Author Contributions: Conceptualization; methodology; software,; validation; formal analysis; investigation; resources; data curation; writing—original draft preparation; writing—review and editing;

CHAPTER 5

Observations and Conclusion

From the above simulations with different number of hidden layers and different optimizer we have made an conclusion that an model with three hidden layers and one dropout layer each with 8 nodes gives us best rmse value when compared with other models with 2 and layers.Among sgd , rmsprop , adam optimizer adam optimizer gives us less error percentage This model can be used in Cloud BMS to predict/estimate RUL of battery

5.1 Possible Research

Digital twin technology can revolutionize automotive industry, with this methodology we can develop full diagnostics devices for automobile. we can diagnose every part of automobile and this helps us for maintenance,operational strategies and helps to increase mileage and reduces maintenance charges of vehicle. Digital twin technology can be incorporated in many other Fields and helps us monitor in real time with less cost and wiring communications.

CHAPTER 6

Python code

```
import datetime
import numpy as np
import pandas as pd
from scipy.io import loadmat
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
#Define loading data method
def load_data(battery):
 mat = loadmat('/content/' + battery + '.mat')
 print('Total data in dataset: ', len(mat[battery][0, 0]['cycle'][0]))
 counter = 0
 dataset = []
 capacity_data = []

 for i in range(len(mat[battery][0, 0]['cycle'][0])):
  row = mat[battery][0, 0]['cycle'][0, i]
   if row['type'][0] == 'discharge':
    ambient_temperature = row['ambient_temperature'][0][0]  date_time
                          int(row['time'][0][1]),
                          int(row['time'][0][2]),
                          int(row['time'][0][3]),
                          int(row['time'][0][4])) + datetime.timedelt
     data = row['data']
     capacity = data[0][0]['Capacity'][0][0]
     for j in range(len(data[0][0]['Voltage_measured'][0])):
```

```
            voltage_measured = data[0][0]['Voltage_measured'][0][j]
            current_measured = data[0][0]['Current_measured'][0][j]
            temperature_measured = data[0][0]['Temperature_measu
            voltage_load = data[0][0]['Voltage_load'][0][j]red'][0][j]
            current_load = data[0][0]['Current_load'][0][j]
            time = data[0][0]['Time'][0][j]
            dataset.append([counter + 1, ambient_temperature, date_time, capac
                            voltage_measured, current_measured,
                            temperature_measured, current_load,
                            voltage_load, time])
             capacity_data.append([counter + 1, ambient_temperature, date_time
             counter = counter + 1
        print(dataset[0])
        return [pd.DataFrame(data=dataset,
                             columns=['cycle', 'ambient_temperature', 'datet
                                      'capacity', 'voltage_measured',
                                      'current_measured', 'temperature_measu
                                      'current_load', 'voltage_load', 'time'
                pd.DataFrame(data=capacity_data,
                             columns=['cycle', 'ambient_temperature', 'datet
                                      'capacity'])]

#Import data set from custom function 05
dataset, capacity = load_data('B0005')
pd.set_option('display.max_columns', 10)
print(dataset.head())
dataset.describe()

plot_df = capacity.loc[(capacity['cycle']>=1),['cycle','capacity']]
sns.set_style("darkgrid")
plt.figure(figsize=(12, 8))
plt.plot(plot_df['cycle'], plot_df['capacity'])
#Draw threshold
```

```
plt.plot([0.,len(capacity)], [1.4, 1.4])
plt.ylabel('Capacity')
# make x-axis ticks legible
adf = plt.gca().get_xaxis().get_major_formatter()
plt.xlabel('cycle')
plt.title('Discharge B0005')


attrib=['cycle', 'datetime', 'capacity']
dis_ele = capacity[attrib]
C = dis_ele['capacity'][0]
for i in range(len(dis_ele)):
    dis_ele['SoH']=(dis_ele['capacity'])/C
print(dis_ele.head(5))

plot_df = dis_ele.loc[(dis_ele['cycle']>=1),['cycle','SoH']]
sns.set_style("white")
plt.figure(figsize=(8, 5))
plt.plot(plot_df['cycle'], plot_df['SoH'])
#Draw threshold
plt.plot([0.,len(capacity)], [0.70, 0.70])
plt.ylabel('SOH')
# make x-axis ticks legible
adf = plt.gca().get_xaxis().get_major_formatter()
plt.xlabel('cycle')
plt.title('Discharge B0005')

C = dataset['capacity'][0]
soh = []
for i in range(len(dataset)):
  soh.append([dataset['capacity'][i] / C])
soh = pd.DataFrame(data=soh, columns=['SoH'])
```

```
attribs=['capacity', 'voltage_measured', 'current_measured',
         'temperature_measured', 'current_load', 'voltage_load', 'time
train_dataset = dataset[attribs]
sc = MinMaxScaler(feature_range=(0,1))
train_dataset = sc.fit_transform(train_dataset)
print(train_dataset.shape)
print(soh.shape)

##Model Making
#Use 3 layers of dense, and use these parameters according to: 3 dense

import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
#import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import LSTM
from tensorflow.keras.optimizers import Adam
#Set GPU resource allocation avoid#Set GPU resource allocation avoid B
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
#Settings keras
from tensorflow.python.keras.backend import set_session
config = tf.ConfigProto()
config.gpu_options.allocator_type = 'BFC'
 #A "Best-fit with coalescing" algorithm, simplified from a version of
config.gpu_options.per_process_gpu_memory_fraction = 0.8
config.gpu_options.allow_growth =True

set_session(tf.Session(config=config))
```

```
model = Sequential()
model.add(Dense(8, activation='relu', input_dim=train_dataset.shape[1]
model.add(Dense(8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(1))
model.summary()
model.compile(optimizer=Adam(beta_1=0.9, beta_2=0.999, epsilon=1e-08),
####
model.fit(x=train_dataset, y=soh.values, batch_size=25, epochs=50)
####


##In order to verify the correctness of the model, load the 06 data se
dataset_val, capacity_val = load_data('B0006')
attrib=['cycle', 'datetime', 'capacity']
dis_ele = capacity_val[attrib]
C = dis_ele['capacity'][0]
for i in range(len(dis_ele)):
    dis_ele['SoH']=(dis_ele['capacity']) / C
print(dataset_val.head(5))
print(dis_ele.head(5))

attrib=['capacity', 'voltage_measured', 'current_measured',
        'temperature_measured', 'current_load', 'voltage_load', 'time'
soh_pred = model.predict(sc.fit_transform(dataset_val[attrib]))
print(soh_pred.shape)

C = dataset_val['capacity'][0]
soh = []
for i in range(len(dataset_val)):
  soh.append(dataset_val['capacity'][i] / C)
new_soh = dataset_val.loc[(dataset_val['cycle'] >= 1), ['cycle']]
```

```
new_soh['SoH'] =  soh
new_soh['NewSoH'] = soh_pred
new_soh = new_soh.groupby(['cycle']).mean().reset_index()
print(new_soh.head(10))
rms = np.sqrt(mean_squared_error(new_soh['SoH'], new_soh['NewSoH']))
print('Root Mean Square Error: ', rms)

plot_df = new_soh.loc[(new_soh['cycle']>=1),['cycle','SoH', 'NewSoH']]
sns.set_style("white")
plt.figure(figsize=(16, 10))
plt.plot(plot_df['cycle'], plot_df['SoH'], label='SoH')
plt.plot(plot_df['cycle'], plot_df['NewSoH'], label='Predicted SoH')
#Draw threshold
#plt.plot([0.,len(capacity)], [0.70, 0.70], label='Threshold')
plt.ylabel('SOH')
# make x-axis ticks legible
adf = plt.gca().get_xaxis().get_major_formatter()
plt.xlabel('cycle')
plt.legend()
plt.title('Discharge B0006')

###Use the first data of the first 50 cycles to use the battery capaci
dataset_val, capacity_val = load_data('B0005')
attrib=['cycle', 'datetime', 'capacity']
dis_ele = capacity_val[attrib]
rows=['cycle','capacity']
dataset=dis_ele[rows]
data_train=dataset[(dataset['cycle']<50)]
data_set_train=data_train.iloc[:,1:2].values
data_test=dataset[(dataset['cycle']>=50)]
data_set_test=data_test.iloc[:,1:2].values

sc=MinMaxScaler(feature_range=(0,1))
```

```
data_set_train=sc.fit_transform(data_set_train)
data_set_test=sc.transform(data_set_test)

X_train=[]
y_train=[]
#take the last 10t to predict 10t+1
for i in range(10,49):
    X_train.append(data_set_train[i-10:i,0])
    y_train.append(data_set_train[i,0])
X_train,y_train=np.array(X_train),np.array(y_train)

X_train=np.reshape(X_train,(X_train.shape[0],X_train.shape[1],1))



regress = Sequential()
regress.add(LSTM(units=200, return_sequences=True, input_shape=(X_trai
regress.add(Dropout(0.3))
regress.add(LSTM(units=200, return_sequences=True))
regress.add(Dropout(0.3))
regress.add(LSTM(units=200, return_sequences=True))
regress.add(Dropout(0.3))
regress.add(LSTM(units=200))
regress.add(Dropout(0.3))
regress.add(Dense(units=1))
regress.compile(optimizer='adam',loss='mean_squared_error')
regress.summary()


regress.fit(X_train,y_train,epochs=200,batch_size=25)

print(len(data_test))
data_total=pd.concat((data_train['capacity'], data_test['capacity']),a
```

```
inputs=data_total[len(data_total)-len(data_test)-10:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)


X_test=[]
for i in range(10,129):
    X_test.append(inputs[i-10:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
pred=regress.predict(X_test)
print(pred.shape)
pred=sc.inverse_transform(pred)
pred=pred[:,0]
tests=data_test.iloc[:,1:2]
rmse = np.sqrt(mean_squared_error(tests, pred))
print('Test RMSE: %.3f' % rmse)
metrics.r2_score(tests,pred)

###Theoretically, it should be seen that the average value of RMSE is
ln = len(data_train)
data_test['pre']=pred
plot_df = dataset.loc[(dataset['cycle']>=1),['cycle','capacity']]
plot_per = data_test.loc[(data_test['cycle']>=ln),['cycle','pre']]
plt.figure(figsize=(16, 10))
plt.plot(plot_df['cycle'], plot_df['capacity'], label="Actual data", c
plt.plot(plot_per['cycle'],plot_per['pre'],label="Prediction data", co
#Draw threshold
plt.plot([0.,168], [1.38, 1.38],dashes=[6, 2], label="treshold")
plt.ylabel('Capacity')
# make x-axis ticks legible
adf = plt.gca().get_xaxis().get_major_formatter()
plt.xlabel('cycle')
```

```
plt.legend()
plt.title('Discharge B0005 (prediction) start in cycle 50 -RULe=-8, wi

pred=0
Afil=0
Pfil=0
a=data_test['capacity'].values
b=data_test['pre'].values
j=0
k=0
for i in range(len(a)):
    actual=a[i]

    if actual<=1.38:
        j=i
        Afil=j
        break
for i in range(len(a)):
    pred=b[i]
    if pred< 1.38:
        k=i
        Pfil=k
        break
print("The Actual fail at cycle number: "+ str(Afil+ln))
print("The prediction fail at cycle number: "+ str(Pfil+ln))
RULerror=Pfil-Afil
print("The error of RUL= "+ str(RULerror)+ " Cycle(s)")
```

REFERENCES

1. https://www.researchgate.net/publication/260722928_Relaxation_model_of_the_open-circuit_voltage_for_state-of-charge_estimation_in_lithium-ion_batteries
2. https://www.researchgate.net/publication/317307864_Lithium-ion_Battery_State_of_ChargeState_of_Health_Estimation_Using_SMO_for_EVs
3. https://www.sciencedirect.com/science/article/pii/S037877530500083via%3Dihub
4. https://www.sciencedirect.com/science/article/pii/S037877532031167sec5
5. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repo
6. Downey, A.; Lui, Y.H.; Hu, C.; Laflamme, S.; Hu, S. Physics-Based Prognostics of Lithium-Ion Battery Using Non-linear Least Squares with Dynamic Bounds. Reliab. Eng. Syst. Saf. 2019, 182, 1–12.
7. Susilo, D.D.; Widodo, A.; Prahasto, T.; Nizam, M. State of Health Estimation of Lithium-Ion Batteries Based on Combination of Gaussian Distribution Data and Least Squares Support Vector Machines Regression. In Materials Science Forum; Trans Tech Publications: Princeton, NJ, USA, 2018; Volume 929, pp. 93–102.
8. Bai, G.; Wang, P.; Hu, C.; Pecht, M. A generic model-free approach for lithium-ion battery health management. Appl. Energy 2014, 135, 247–260.
9. Saha, B.; Goebel, K. Modeling Li-ion battery capacitydepletion in a particle filtering framework. In Proceedings of the Annual Conference of the Prognostics and Health Management Society, San Diego, CA, USA, 27 September 27–1 October 2009; pp. 2909–2924.
10. Miao, Q.; Xie, L.; Cui, H.; Liang, W.; Pecht, M. Remaining useful life prediction of lithium-ion battery with unscented particle filter technique. Microelectron. Reliab. 2013, 53, 805–810
11. Tong, S.; Klein, M.P.; Park, J.W. On-line optimization of battery open circuit voltage for improvedstate-of-charge and state-of-health estimation.J. Power Sources2015,293, 416–428.
12. Weng, C.; Sun, J.; Peng, H. A unified open-circuit-voltage model of lithium-ion batteries for state-of-charge estimation and state-of-health monitoring.J. Power Sources2014,258, 228–237.
13. 20.Ng, K.S.; Moo, C.-S.; Chen, Y.-P.; Hsieh, Y.-C. Enhanced coulomb counting method for estimatingstate-of-charge and state-of-health of lithium-ion batteries.Appl. Energy2009,86, 1506–1511.

14. Fang, L.; Li, J.; Peng, B. Online Estimation and Error Analysis of both SOC and SOH of Lithium-ion Battery based on DEKF Method.Energy Procedia2019,158, 3008–3013

15. Bakas, E.; Rosca, B.; Wilkins, S.; Donkers, T. Least-Squares-Based Capacity Estimation for Lithium-ion BatteryCells. In Proceedings of the EEVC 2017—The European Battery, Hybrid Fuel Cell Electric Vehicle Congress,Geneva, Switzerland, 14–17 March 2017

16. Nuhic, A.; Terzimehic, T.; Soczka-Guth, T.; Buchholz, M.; Dietmayer, K. Health diagnosis and remaininguseful life prognostics of lithium-ion batteries using data-driven methods.J. Power Sources2013,239, 680–688.

17. Kim, J.; Yu, J.; Kim, M.; Kim, K.; Han, S. Estimation of Li-ion Battery State of Health based on MultilayerPerceptron: as an EV Application.IFAC-PapersOnLine2018,51, 392–397.