APPLY AND OPTIMIZE MACHINE LEARNING ALGORITHMS FOR ESTIMATING BATTERY HEALTH

By

Chin Wai Yee

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Lee Wai Kong who has given me this bright opportunity to engage in a machine learning project. It is my first step to establish a career in machine learning field. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

With the growing demand for energy-efficient and reliable battery-powered systems, accurate estimation of battery State of Charge (SOC), State of Health (SOH), and Remaining Useful Life (RUL) must be monitored to ensure safety, performance, and longevity. Traditional estimation techniques such as Coulomb counting and model-based approaches often suffer from error accumulation, calibration complexity, and poor adaptability to dynamic conditions.

This project investigates machine learning (ML) techniques for estimating the SOC and RUL based on Electrochemical Impedance Spectroscopy (EIS) data. A range of regression and classification models including Random Forest (RF), Support Vector Machines (SVM), Gaussian Process Regression (GPR), and Artificial Neural Networks (ANN) were evaluated on both full-frequency and single-frequency EIS inputs. Results show that full-spectrum EIS features provide superior predictive performance, with Random Forest excelling in regression tasks and ANN achieving the highest classification accuracy. For RUL estimation, ANN and CNN-SAM models demonstrated competitive accuracy compared to baseline Gaussian Process Regression, effectively capturing degradation patterns across different operating temperatures.

To enable deployment on resource-constrained embedded systems, pruning and quantization techniques were employed to compress model size while preserving predictive accuracy. Optimization reduced ANN size from 260 kB to 26 kB and CNN-SAM from 1679 kB to 158 kB, confirming that lightweight yet robust models can be achieved without significant performance loss.

The findings confirm the potential of integrating EIS data with optimized ML models for real-time battery state estimation. This work provides a pathway toward practical, efficient, and intelligent BMS capable of supporting the growing adoption of lithium-ion batteries in diverse applications.

Area of Study: Battery State Estimation, Machine Learning

Keywords: Battery Management System, State of Charge, Remaining Useful Life, Electrochemical Impedance Spectroscopy, Machine Learning, Model Optimization

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BMS	Battery Management System
SOC	State of Charge
SOH	State of Health
RUL	Remaining Useful Life
EOL	End of Life
ML	Machine Learning

Chapter 1 Introduction

1.1 Problem Statement and Motivation

As the use of battery-powered devices, electric vehicles, and renewable energy systems expands, the demand for dependable and efficient Battery Management Systems has significantly increased. The batteries require effective monitoring of their internal state, which are the State of Charge (SOC), which reflects the available capacity at a given moment, the State of Health (SOH), represent the remaining capacity of a battery relative to its initial capacity, and the Remaining Useful Life (RUL), which indicates the number of cycles left before the battery reaches its end of life (EOL) cycle. Accurate estimation of SOC, SOH, and RUL is essential for ensuring safety and ensure optimal performance and longevity [1].

Traditional methods for estimating battery SOC include Coulomb counting, Open Circuit Voltage (OCV) measurement, and model-based approaches like Equivalent Circuit Models (ECMs). In Coulomb counting, SOC is calculated by measuring and integrating current over time. However, it accumulates errors over time due to sensor inaccuracies [2]. OCV is defined as the voltage measured following a relaxation period of several hours. However, this method requires the battery to rest for extended periods to reach equilibrium, making it unsuitable for real-time applications [3]. ECMs, often combined with Kalman filters, simulate the battery's behaviour using electrical components to estimate SOC and SOH. Although more accurate, these models require complex parameter tuning and are sensitive to temperature and aging effects [4], [5].

Machine Learning (ML) has developed as a promising solution to overcome these challenges. Unlike physics-based models, ML methods nonlinear relationships from operational data, enabling robust estimation of SOC and SOH under diverse operating conditions, enhancing adaptability and accuracy. Recent studies [6], [7], [8] have demonstrated the efficiency of various ML techniques, including Support Vector Machines (SVM), Neural Networks (NN), and Ensemble Methods, in estimating SOC and SOH with high accuracy. These models can process large datasets to capture intricate patterns in battery behaviour, offering improved performance over traditional methods. Moreover, combining Electrochemical Impedance Spectroscopy (EIS) with ML models enhances SOH and RUL prediction by capturing the battery's internal

electrochemical dynamics. This combination of data-driven and physics-based approaches leverages the strengths of both methodologies for more reliable predictions.

Although ML offers significant advantages in battery state estimation, deploying these models on embedded systems remains challenging because of limited computational power and memory constraints. [9], [10]. High-complexity models may not meet the real-time processing requirements of BMS. Thus, optimization techniques were required to reduce model size and inference time. Model pruning involves eliminating redundant or less important parameters from the neural network. This method reduces the model size without significantly compromising accuracy. Quantization converts the model's weights and activations from high-precision (e.g. 32 bits floating number) to lower-precision (e.g. 8 bits integer) representations, decreasing memory usage and increasing computational efficiency. Applying these optimization techniques enables the deployment of ML models on resource-limited embedded platforms, facilitating real-time SOC and RUL estimation in practical applications.

1.2 Research Objectives

The objectives of this project are outlined as follows:

1. Evaluate and compare the effectiveness of different machine learning algorithms in estimating battery State of Charge (SOC) and Remaining Useful Life (RUL).
2. Apply pruning and quantization techniques to reduce the model size to 200KB or below, ensuring suitability for deployment on embedded systems without significant loss of accuracy.

1.3 Project Scope and Direction

This project focuses on reproducing and evaluating existing machine learning-based SOC prediction models using publicly available battery datasets. In addition, build an RUL prediction model based on EIS spectrum.

1.4 Contributions

This project has the following contributions:

1. Provide a comprehensive evaluation of ML algorithms for battery state estimation

2. Development of optimized ML models suitable for real-time embedded applications

1.5 Report Organization

Chapter 2 discusses literature and related works. Chapter 3 explains the experimental setup and proposed methods. Chapter 4 presents preliminary results and findings. Chapter 5 concludes the report and highlights future work.

Chapter 2 Literature Reviews

2.1 Electrochemical Impedance Spectroscopy (EIS)

EIS is a powerful electroanalytical method for characterizing the electrical properties of electrochemical systems, including batteries. It measures the impedance response to a small sinusoidal perturbation in potential or current over a range of frequencies, providing information on resistance, capacitance, and diffusion processes within the system [11], [12]. Unlike other electrochemical methods, EIS introduces only a minimal disturbance, making it suitable for monitoring under near-operating conditions. This non-destructive nature is particularly advantageous for evaluating battery health and performance evolution over time.

In EIS analysis, the system response is described by the complex impedance [12]:

$$Z = Z_0 e^{i\Phi} = Z_0(\cos\Phi + i\sin\Phi)$$

where Z_0 is the impedance magnitude and ϕ is the phase shift between the applied potential and the resulting current. When plotted, the real component Z_{real} (Re) is typically placed on the X-axis and the imaginary component Z_{imag} (-Im) on the Y-axis to generate a Nyquist plot, in which each point corresponds to a specific frequency. High-frequency data appear on the left of the X-axis, while low-frequency data extend to the right as shown in Figure 2.1.

To model and interpret the EIS spectrum, Equivalent Circuit Models (ECMs) are widely used, which approximate electrochemical processes by electrical components such as capacitors, constant phase elements (CPEs), resistors, and Warburg elements [13]. In lithium-ion batteries, the impedance spectrum can be broadly divided into three frequency regions, each corresponding to different physical/chemical processes inside the cell:

1. High-frequency region (Section 1 in Figure 2.1): Represents very fast, ohmic effects, including resistance in the electrolyte and electrodes and inductive parasitic from wiring, cell leads etc.
2. Mid-frequency region (Section 2 in Figure 2.1): Reflects interfacial phenomena such as charge transfer at electrode/electrolyte boundaries, double-layer

capacitance, and effects from the solid electrolyte interphase (SEI). Modeled using resistor - capacitor or resistor - CPE (constant phase element) networks.

3. Low-frequency region (Section 3 in Figure 2.1): Dominated by diffusion processes including lithium-ion diffusion through electrode materials or porous structure. These are modeled using Warburg or modified Warburg elements, often connected to the mid-frequency elements to reflect realistic diffusion behavior.

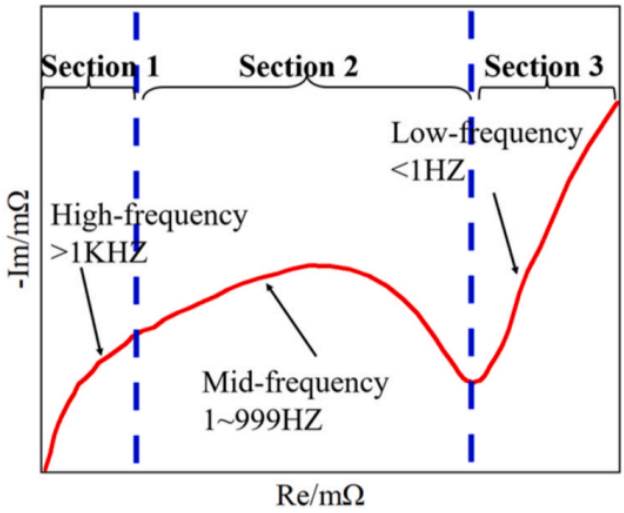


Figure 2.1 EIS impedance spectrum, adapted from [14]

2.2 Previous works on SOC prediction

Several recent works [6] have been done on SOC prediction using machine learning regression algorithms, including Neural Network, Support Vector Machine, and Gaussian Process Regression. The study in [6] has done a comparative assessment of the models, using input parameters such as humidity, pack voltage, pack current, pack cell temperature, motor temperature, FET temperature, and ambient temperature. The key result of the experiment was that the Gaussian Process Regression (GPR) model demonstrated the best SOC prediction accuracy compared to the other machine learning models.

On the other hand, [15] first done the feature sensitivity analysis with Pearson correlation coefficient on impedances values across 54 frequencies ranging from 1 mHz to 6 kHz. The study found that EIS data in high and mid-frequency regions are strongly correlated with SOC. Then, two machine learning models were trained, a linear

regression and a Gaussian Process Regression (GPR) to map the EIS features to SOC. Importantly, the GPR model achieved a SOC prediction error under 3.8%.

In other studies, [16] proposed a novel method for estimating the state of charge (SOC) of lithium-ion batteries by combining electrochemical impedance spectroscopy (EIS) measurements with convolutional neural networks (CNNs). Their approach leverages the distinctive visual patterns of impedance curves to classify SOC levels with high accuracy, achieving up to 80% top-2 classification accuracy on previously unseen batteries, and up to 100% top-2 accuracy on known batteries. However, the model's performance significantly declines when estimating SOC for unseen battery, demonstrating only 62% accuracy. Thus, more datasets may be needed to build a better model.

In addition, [17] has done a comprehensive analysis of using EIS for battery SOC estimation in Lithium-Ion batteries, which compare different machine learning methods and found that utilizing the full EIS frequency range of impedance as input yields better performance than individual frequency-impedance pairs. The LSTM model excelled in regression analysis with low error metrics and a high R^2 value, while ANN, DA, LSTM, and TM demonstrated superior performance in the classification scenario. In addition, [18] also explored ML based SOC estimation using EIS as classification problem with the same dataset. The study transforms the original EIS data using an equivalent circuit model and successfully demonstrated that this transformation enhances the accuracy of SOC classification. As an output, the study trained a SOC classification model which achieves an accuracy of above 93%

Furthermore, similar work done in [19] explores the application of tiny machine learning (TinyML) for estimating the SOC of electric vehicle batteries, emphasizing hardware-accelerated implementation on microcontrollers and microprocessors. It contrasts traditional model-based SOC estimation techniques, which struggle with the non-linear dynamics of batteries and require complex modelling. The study evaluated various tiny neural network architectures, including Feedforward Neural Networks (FNN), Long Short-Term Memory (LSTM), Temporal Convolutional Network (TCN), Gated Recurrent Unit (GRU), and Legendre Memory Units (LMU), using a dataset from a BMW i3. As a result, the Parallel TCN architecture achieved the highest accuracy, while the stateless LSTM offered a balance between accuracy and

computational demands, highlighting the potential of TinyML for efficient and decentralized BMS in electric vehicles, especially when coupled with hardware acceleration.

2.3 Previous works on RUL prediction

One of the foundational works in the field utilized a Gaussian Process Regression (GPR) model to estimate capacity and predict RUL [20]. GPR is a probabilistic and non-parametric machine learning model that is well-suited for regression tasks. This research introduced a crucial concept: using the entire EIS spectrum as a direct input to the model, rather than relying on an expert to manually select features from it. A key aspect of this GPR model was its use of an Automatic Relevance Determination (ARD) kernel, which functions as an automatic feature selection mechanism. The ARD kernel assigns importance weights to each input feature, effectively down-weighting and pruning irrelevant frequencies from the high-dimensional EIS spectrum. By analyzing the weights assigned by the model, researchers could identify which spectral features were most salient for predicting battery degradation. The GPR model achieved an R^2 value of 0.96 for RUL prediction for batteries cycled at 25°C. The model's analysis of the EIS spectrum for this dataset revealed that frequencies 17.80 Hz and 2.16 Hz were sufficient to estimate capacity and predict RUL.

Another early approach [21] used a linear Lasso regression algorithm to predict RUL based on a minimal number of features. Unlike the GPR model that automatically selected features, this method relied on a pre-selected set of inputs from the first 20 cycles of a battery's life. The chosen features included the negative out-of-phase impedance response (-Im(Z)) at 20 kHz and 8.8 Hz, in combination with temperature data. This simplified model demonstrated remarkable accuracy, achieving RUL predictions with an R^2 greater than 0.96. The research further showed that the model could be simplified even more, to require only the 20 kHz impedance response and temperature, with the model still maintaining an R^2 greater than 0.90.

More recent research has moved beyond foundational regression techniques to leverage the power of advanced deep learning architectures. The CNN-BiLSTM model is a sophisticated hybrid architecture that combines a 1d Convolutional Neural Network (CNN) with a Bidirectional LSTM (BiLSTM) network [22]. The CNN component acts as a feature extractor, automatically processing the high-dimensional EIS spectrum to

identify critical degradation patterns. The BiLSTM component then analyzes the temporal sequence of these extracted features across successive cycles, enabling the model to learn and predict the long-term degradation trend. The model demonstrated superior performance in SOH estimation and RUL prediction compared to the GPR model and other standard neural networks like LSTM and CNN-LSTM. The model's reported R^2 values for RUL prediction were 0.85 at 25℃, 0.86 at 35℃, and 0.94 at 45℃, surpassing the performance of earlier models on the same multi-temperature dataset. A key strength of this approach is its ability to forecast the entire capacity degradation trajectory from a limited number of early cycles, such as predicting 300 cycles of degradation from only the first 50 cycles.

Another hybrid model, the Conv1D-SAM [14], which combine CNN and self-attention mechanism was developed to predict RUL using the negative imaginary part of the EIS impedance as input. This study demonstrated that the negative imaginary component of the impedance spectrum provides a strong representation for RUL prediction. This feature was found to be more robust for RUL prediction as its change with aging was less dependent on temperature compared to the real part of the impedance. The Conv1d-SAM model was specifically engineered to mine the detailed degradation information within the negative imaginary impedance data. It utilizes the self-attention mechanism to weight the importance of different temporal features, improving the model's ability to adapt to the inherent uncertainty in battery aging. The model demonstrated a significant improvement over the "latest published method," with its Mean Absolute Error (MAE) being improved by 72% on average across nine test cells.1 This suggests that a focus on a specific, physically relevant subset of the EIS data can yield exceptional results.

A third advanced hybrid model, the CNN-Transformer network [23], was also proposed for SOH estimation and RUL prediction. This model initially uses a CNN to process the EIS data, which is then passed to a Transformer network. The Transformer's self-attention mechanism effectively captures long-range dependencies in sequential data, making it well-suited for time-series tasks such as battery degradation prediction. In the data pre-processing stage, this research used the Pearson correlation coefficient (PCC) to select the most informative EIS features, confirming that EIS data from a fully rested state (State V) had the highest correlation with battery capacity data. The model was also tested on a low-performance platform without GPU support, demonstrating its

practicality for real-world applications. The CNN-Transformer model achieved an exceptional R^2 value of 0.9872 for SOH estimation.

2.4 Pruning

Pruning is a method used to reduce the number of parameters in a machine learning model by eliminating weights or entire components that contribute little to the model's performance. This results in a smaller model, which can lower memory usage, speed up inference, and improve energy efficiency. Pruning is generally classified into structured and unstructured types.

2.4.1 Structured pruning

Structured pruning involves removing entire structures from the neural network, such as neurons, filters, or channels [24]. In contrast with unstructured pruning, structured pruning results in more regular and compact models that can be efficiently processed by standard hardware structured sparsity and does not rely on special hardware or software specification [25]. Previous work of structured pruning includes [26], [27], [28], [29], [30], [31], [32].

The work done in [31] has proposed a CoFit pruning method while [27] has proposed a structural pruning method which re-parameterize the Convolutional Neural Networks (CNN) into two parts, remember part to minimize the size of model with slightly loss in accuracy. Similar previous work that is inspired by neurobiology is also done in [33]. The paper [33] has discussed on the pruning plasticity, which means the speed of a model recover its accuracy after pruning. It suggested that pruning techniques should use low pruning with high learning rate to recover faster in terms of accuracy. The paper [33] also proposed GraNet, which prune the model during training and introduce the regeneration of connections. In addition of this previous work, the work done in[30] proposed an EagleEye pruning method which can be used to plug-in on other pruning methods to increase the effectiveness of structural pruning. In [32], the paper had proposed a pruning on LLM using dependency graph. Where similar work done in [26] also constructed dependency graph for structured pruning on neural networks (NN) but the proposed method is more general which it can be applied to any type of NN. The work in [28] also justifies that identify the redundancy layer instead of unimportant

layer is more effective. For example, the proposed work in [28] prune the filters in the most redundant layer that identified by the dependency graph.

2.4.2 Unstructured pruning

Unstructured pruning refers to pruning the redundant weights by setting their corresponding masks to zero [25], resulting in a sparse weight matrix. This method can achieve higher sparsity ratios compared to structured pruning, as it offers more flexibility in identifying and eliminating less important parameters. However, Specialized hardware or software libraries are often required to realize the full benefits of unstructured pruning in terms of inference speedup [25]. Examples of previous work which done on unstructured pruning includes [34], [35]. [34] has proposed Wanda pruning technique, which utilizes the weights and activation to prune a pretrained Large Language Model (LLM). On the other hand, [35] has use movement pruning to prune language model. These unstructured pruning techniques are simple to be implemented and lead to high compression rate [24].

2.5 Quantization

Quantization refers to reducing the precision of the data type to reduce the computations power needed [36]. An example of quantization is converting data type of F32 (32-bit floating-point) to INT8 (8-bit integer). By quantization, the required computational power is saved as the data are stored in less bits. Quantization can be categorized into two groups; the first one is Post Training Quantization (PTQ) and the second is Quantization Aware Training (QAT) according to the time to apply quantization on the model.

In addition of PTQ and QAT, quantization can also divide into two groups, which are symmetric quantization and asymmetric quantization. The main difference between these two quantization techniques is, symmetric quantization restricts the zero-point to 0, while asymmetric quantization does not [37].

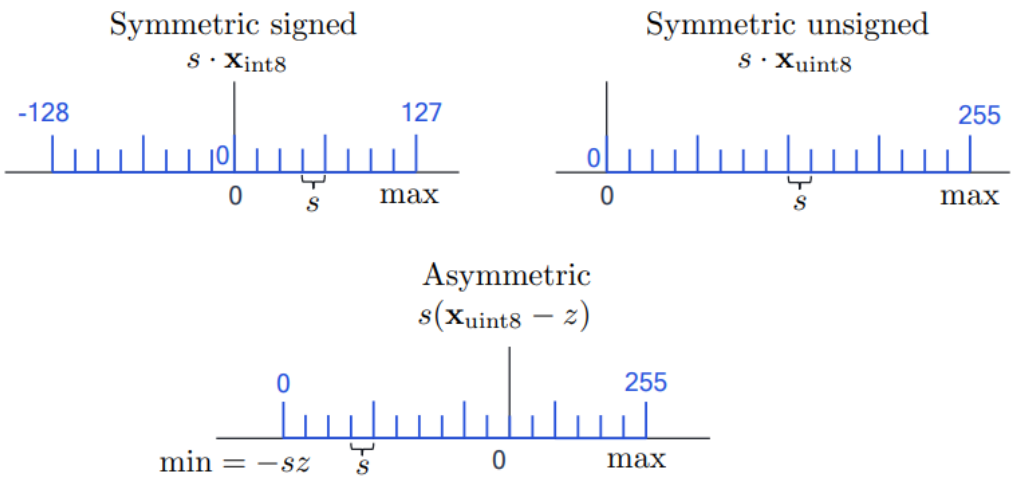


Figure 2.2 Visualization of symmetric and asymmetric quantization

As shown in Figure 2.2, the centre of symmetric signed quantization is 0, while the centre of asymmetric quantization is not. In this project, the asymmetric quantization will be applied to activation while symmetric weight quantization will be applied to weight as referred to [37].

2.5.1 Post Training Quantization

PTQ refers applying quantization on a trained model. Previous work for PTQ such as [38] has developed Brecq and try to push the limit of PTQ to 2-bit. PTQ is done by applying the quantization on the trained model. For example, quantization of the weights and activations of a trained model so that the parameters will not take too much space to be stored.

2.5.2 Quantization Aware Training

QAT is quantization of the parameters when the machine learning model is still within training. This project will focus on the implementation of QAT on machine learning model. Previous works that implement QAT is [39]. In [39], the study has proposed a novel QAT by introducing the block-wise QAT into the traditional QAT which only use end-to-end quantization. In addition, the work done in [40] has tried to proposed Bit width-adaptive quantization. In which, the bit of quantization will vary and not fixed like traditional quantization. However, this will require more computational time. Furthermore, the work done in [41] had proposed two quantization algorithm which can

avoid the overcoming oscillations in QAT. Oscillations means the change of quantized value between two adjacent layers.

Chapter 3 Proposed Method and Experiment Setup

This project conducted two experiments: the first focused on predicting the SOC with EIS spectrum, and the second on predicting the RUL with EIS spectrum.

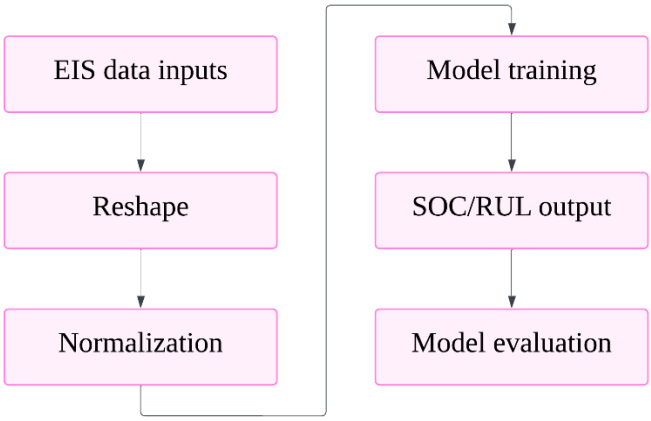


Figure 3.1 Workflow for SOC and RUL prediction

3.1 SOC prediction with EIS

This project begins by replicating the SOC prediction with EIS experiment from [17] using a list of selected models. The implementation of the machine learning models was done with Python Sklearn Library for non-neural network models and Tensorflow Library for ANN.

3.1.1 Data acquisition

In this project, the dataset was sourced from [42]. The dataset was originally generated using four newly manufactured Samsung cylindrical ICR18650-26J rechargeable lithium-ion batteries, each with a nominal capacity of 2600 mAh and a nominal voltage of 3.6 V. In the original study, experiments were repeated six times for each battery, with the batteries fully charged according to the manufacturer's specifications prior to testing. The open-circuit voltage after charging was measured at 4.12 V.

EIS measurements were carried out using a broadband measurement strategy across State of Charge (SOC) levels 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100%, with 14 frequencies sweeping from 0.05 Hz to 1 kHz after a relaxation period of 1800 seconds per SOC measurement, as proposed in [42]. This procedure resulted in a

total of 3360 impedance values (6 measurements × 4 batteries × 10 SOC levels × 14 frequencies).

3.1.2 Experiment Setup

Following [17], two types of exploratory experiments were conducted:

1. In this experiment, impedance values across the entire frequency range were used as input features for the machine learning (ML) models. This yielded a total of 240 samples (6 measurements × 4 batteries × 10 SOC levels), with 200 samples used for training and 40 for testing.
2. Individual EIS Frequency–Impedance Pair Analysis: In this case, each impedance value and its corresponding frequency were treated as individual features, resulting in 3360 samples. The dataset was divided into 2800 training samples and 560 testing samples.

Feature scaling was applied using standardization to ensure zero mean and unit variance. The impedance at each frequency is represented as:

$$Z = Z_{\text{real}} + jZ_{\text{imag}}$$

$$Z_{\text{real}} = \left[Z_{\text{real,f01}}, Z_{\text{real,f02}}, \dots, Z_{\text{real,fn}}\right]$$

$$Z_{\text{imag}} = \left[Z_{\text{imag,f01}}, Z_{\text{imag,f02}}, \dots, Z_{\text{imag,fn}}\right]$$

The index f01–fn corresponds to the frequency number. The data was normalized to zero mean and unit variance using:

$$x^* = \frac{x - \bar{x}}{\sigma}$$

where x is the original value, $\bar{x}$ is the mean, and σ is the standard deviation.

After normalization, the impedance data is expressed as:

$$Z_{\text{real,norm}} = \left[\frac{Z_{\text{real,f01}} - Z_{m,\text{real,f01}}}{\sigma_{\text{real,f01}}}, \frac{Z_{\text{real,f02}} - Z_{m,\text{real,f02}}}{\sigma_{\text{real,f02}}}, \dots, \frac{Z_{\text{real,f60}} - Z_{m,\text{real,fn}}}{\sigma_{\text{real,fn}}}\right]$$

$$Z_{\text{imag,norm}} = \left[\frac{Z_{\text{imag,f01}} - Z_{m,\text{imag,f01}}}{\sigma_{\text{imag,f01}}}, \frac{Z_{\text{imag,02}} - Z_{m,\text{imag,f02}}}{\sigma_{\text{imag,f02}}}, \dots, \frac{Z_{\text{imag,f60}} - Z_{m,\text{imag,fn}}}{\sigma_{\text{imag,fn}}}\right]$$

$$Z_{\text{norm}} = Z_{\text{real,norm}} + Z_{\text{imag,norm}}$$

A summary of the data modeling workflow for both experiments is illustrated in Figure 3.2.

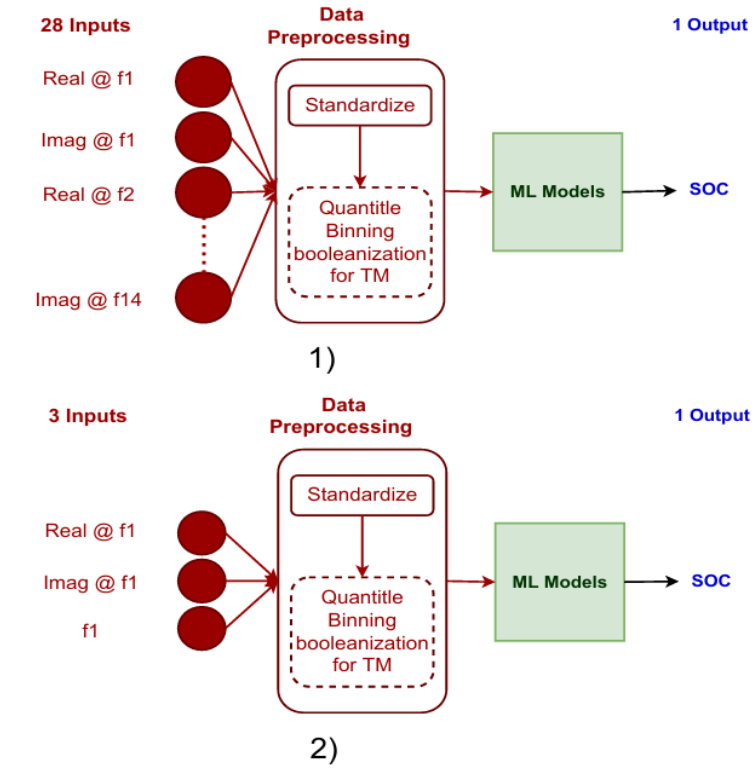


Figure 3.2 Flow of input features and output of experiment 1 and 2

3.1.3 Machine Learning Models

In this project, Gaussian Process Regression, Linear Regression, Support Vector Machine, Kernel Ridge Regression, Random Forest and Artificial Neural Network were used as regression models. K-Nearest Neighbor, Quadratic Discriminant Analysis, Gaussian Naïve Bayes, Support Vector Machine, Random Forest and Artificial Neural Network were used as classification models.

i. Gaussian Process Regression (GPR)

Gaussian Process Regression (GPR) is a non-parametric, probabilistic method that defines a distribution over functions using a mean and a kernel (covariance) function. Given training data, Bayesian inference updates the prior to a posterior distribution, providing predictions with associated uncertainty. GPR works well with small datasets, allows incorporation of prior knowledge, and gives interpretable, probabilistic outputs, but it can be sensitive to kernel choice and does not scale efficiently to large datasets [43].

ii. Linear Regression (LR)

Linear Regression is a method used to examine the relationship between a dependent variable and a single independent variable. Simple Linear Regression involves one predictor and one response variable, aiming to model their relationship by fitting a regression line to the observed data [44].

iii. Support Vector Machines (SVM)

Support Vector Machines (SVMs), originally developed for classification, can also be applied to regression tasks (Support Vector Regression). SVMs aim to find a hyperplane that best fits the data points within a certain margin of tolerance (ϵ), while minimizing the complexity of the model [45]. Instead of minimizing the squared error, SVR minimizes the error within a defined margin, making it robust to outliers. The use of kernel functions allows SVMs to model non-linear relationships by mapping the data into a higher-dimensional feature space.

iv. Kernel Ridge Regression (KR)

Kernel Ridge Regression is a regression method that combines the kernel trick, enabling nonlinear modeling, with regularization to prevent overfitting. It minimizes the squared prediction error while controlling model complexity through a regularization parameter that balances accuracy and stability [46].

v. Random Forest (RF)

Random Forest (RF) is an ensemble learning method consisting of multiple decision trees. By using the bagging approach, it improves prediction accuracy and reduces the risk of overfitting compared to a single decision tree, while enabling parallel execution of trees for more efficient problem-solving [47].

vi. Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the human brain, consisting of interconnected nodes (artificial neurons) that process and transmit information collaboratively [48].

vii. K-Nearest Neighbor (KNN)

The k-Nearest Neighbors (k-NN) algorithm is a classical nonparametric method used for both classification and regression. It is an instance-based, or "lazy learning," approach, meaning that during training, the algorithm simply stores the data, and all computations occur during testing to identify the k nearest neighbors. For a test sample, k-NN predicts its class (for classification) by taking a majority vote among its k closest neighbors, or its value (for regression) by averaging the labels of these neighbors [49].

viii. Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a classification technique that assumes that the data for each class's data follows a multivariate Gaussian distribution and that each class has its own covariance matrix. Unlike Linear Discriminant Analysis, which assumes equal covariance matrices across classes, QDA allows for more flexibility by modeling quadratic boundaries between classes. This makes QDA more powerful when the decision boundaries are non-linear, but it requires more parameters to be estimated than LDA

ix. Gaussian Naïve Bayes (GNB)

Gaussian Naïve Bayes is a probabilistic classification method that uses Bayes' theorem, assuming that all features are conditionally independent given the class. Specifically, for continuous features, it assumes that the likelihood of the features given the class is distributed according to a Gaussian distribution. This simplicity makes Gaussian Naïve Bayes computationally efficient and effective, particularly for high-dimensional datasets, despite the often-unrealistic independence assumption

3.1.4 Hyperparameter settings

GPR	Kernel scale	3.2772
LR	Alpha	0.0023
	Solver	lsqr
SVM	Kernel	rbf
	C	7.7298
	Gamma	0.2582
	Epsilon	0.9447
KR	Alpha	0.000005249
	Kernel	rbf
	Gamma	6.7342
RF	N Estimators	486

	Min samples leaf	1
	Boostrap	True
ANN	Activations	Sigmoid
	LayerSizes	[298 275]

Table 3.1: Hyperparameter setting for regression models, experiment 1

Model	Hyperparameter	Value
KNN	n_neighbors	1
	metric	chebyshev
QDA	reg_param	0.2576
GNB	(No specific hyperparameters tuned)	
SVM	kernel	rbf
	C	0.0011
	gamma	0.0196
	decision_function_shape	ovo
RF	estimator (DecisionTreeClassifier)	max_depth=3, min_samples_leaf=5
	n_estimators	18
	learning_rate	0.9665
ANN	layers	[295, 30]
	lambda_reg	0.000073798
	optimizer	Adam(learning_rate=0.001)
	loss	sparse_categorical_crossentropy

Table 3.2: Hyperparameter setting for classification models, experiment 1

Model	Hyperparameter	Value
GPR	kernel (RBF)	length_scale=22.572
	alpha	3
LR	alpha	0.0024
	solver	lsqr
SVR	kernel	rbf
	C	186.9
	gamma	56.177
	epsilon	0.1187
KR	alpha	0.00000256
	kernel	rbf
	gamma	0.62218
RF	n_estimators	230
	min_samples_leaf	1
	bootstrap	TRUE
ANN	layers	[251, 267]
	lambda_reg	0.0093249

Table 3.3: Hyperparameter setting for regression models, experiment 2

Model	Hyperparameter	Value
KNN	n_neighbors	5
	metric	minkowski
QDA	reg_param	0.01
GNB	(No specific hyperparameters tuned)	
SVM	kernel	rbf
	C	1
RF	n_estimators	486
ANN	layers	[25, 268]
	lambda_reg	4.04×10^{-9}

Table 3.4: Hyperparameter setting for classification models, experiment 2

3.2 RUL prediction with EIS

3.2.1 Data acquisition

The dataset used for RUL prediction is from [20]. It was generated from cycling tests on 12 commercial 45 mAh Eunicell LR2032 Li-ion coin cells ($LiCoO_2$/graphite). The cells were aged in three climate chambers at 25 ℃ (8 cells, labelled 25C01 to 25C08), 35 ℃ (2 cells, labelled 35C01 and 35C02), and 45 ℃ (2 cells, labelled 45C01 and 45C02). Each cycle involved charging at 1C (45 mA) in CC–CV mode up to 4.2 V, followed by discharging at 2C (90 mA) in CC mode down to 3 V. Electrochemical impedance spectroscopy (EIS) was recorded every second cycle at nine states of charge/discharge between 0.02 Hz and 20 kHz using a 5 mA excitation current, after a 15-minute rest at 0% and 100% SOC. Capacity fade was measured after every odd cycle. Figure 3.3 shows 9 stages where EIS data is collected. This project will use stage V (15 minutes rest after charging). Figure 3.4 shows the degradation pattern of batteries.

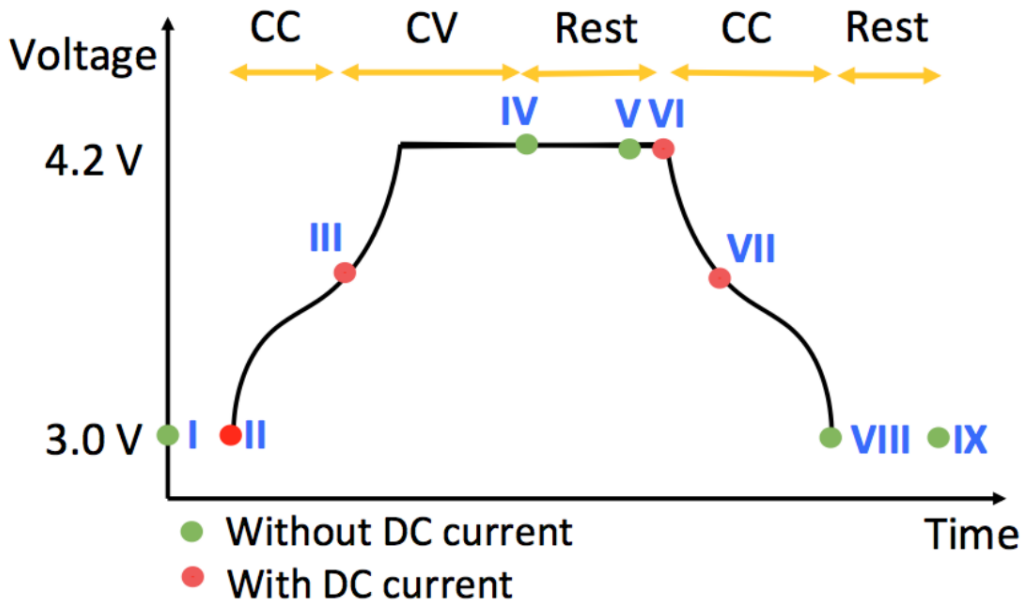


Figure 3.3 Nine Stages where EIS data is collected

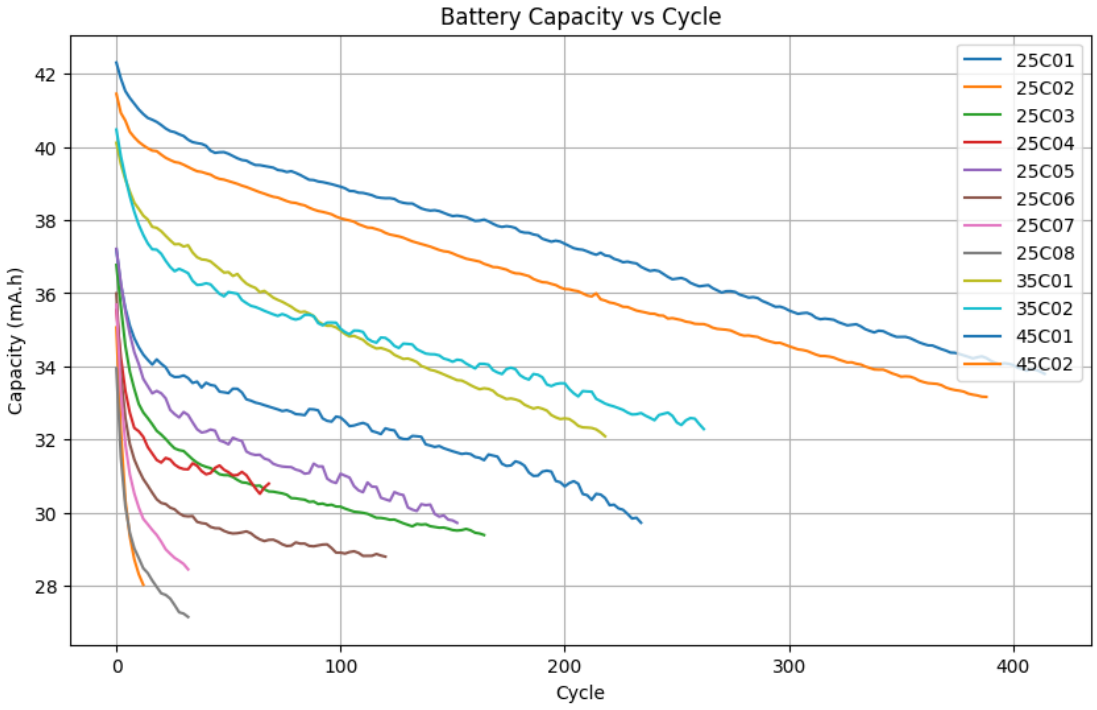


Figure 3.4 Degradation pattern of capacity for each battery

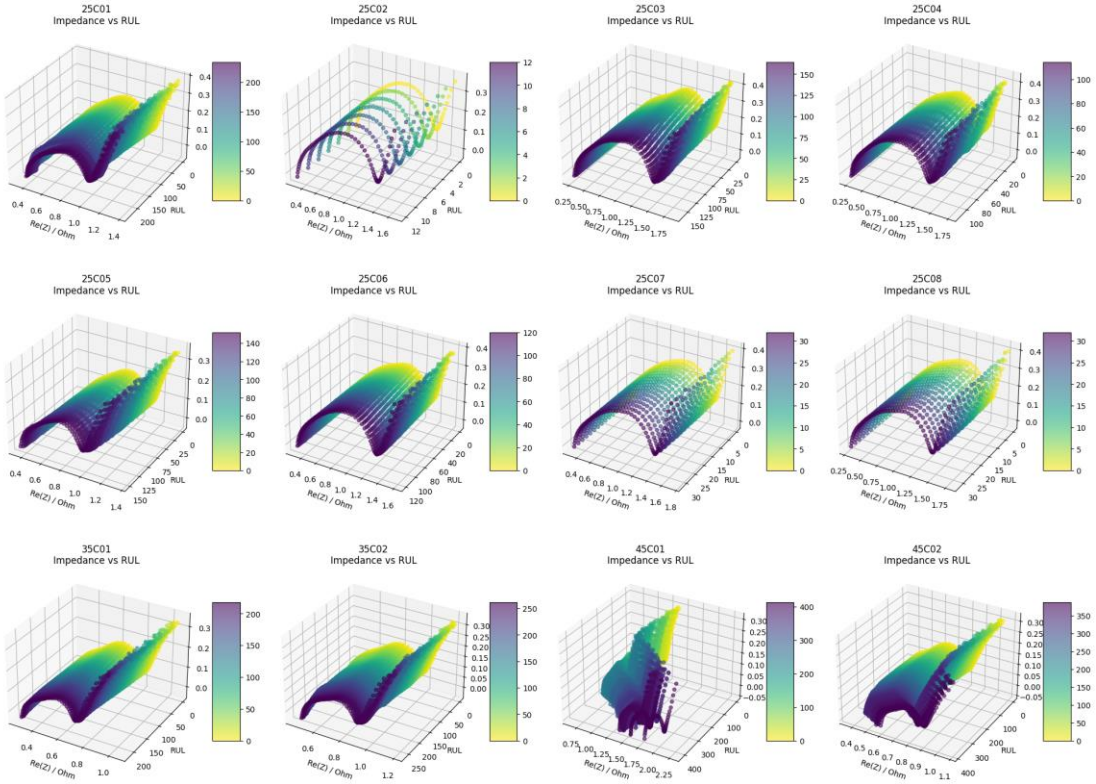


Figure 3.5 Change of EIS spectrum according to cycles

3.2.2 Data pre-processing

The electrochemical impedance spectroscopy (EIS) measurements cover a frequency range of 0.02 Hz to 20 kHz, with n number of frequency points in total. The EIS data was standardized using the same standardization method explained in Section 3.1.2.

To improve model robustness and reduce overfitting, Gaussian noise augmentation was applied to the training data. For each impedance feature, zero-mean Gaussian noise was added with a standard deviation proportional to the feature's inherent variability. Specifically, the standard deviation of the noise was set to a fraction of the per-feature standard deviation:

$$X_{\text{aug}} = X + \mathcal{N}(0, (\text{frac} \cdot \sigma_X)^2)$$

Where X is the original input feature, $\mathcal{N}(0, (\text{frac} \cdot \sigma_X)^2)$ denotes Gaussian noise, σ_X is the per-feature standard deviation, *frac* is the noise scaling factor.

In this study, the noise fraction (*frac*) was set to 0.02, which means the added noise was 2% of the standard deviation of each feature. For every original sample, two extra noisy versions were created, so the training dataset became three times larger. The target

values stayed the same, since only the input features were changed by the noise. This strategy introduces realistic variability into the training process, encouraging the models to generalize better to new data while maintaining the physical consistency of the impedance spectrum.

3.2.3 Machine Learning Models

i. Artificial Neural Network

The ANN baseline was implemented as a fully connected feedforward network. It consists three hidden layers consisting of 128, 64, and 32 neurons, each using ReLU activation, dropout for regularization, and the output layer is a neuron with linear activation.

ii. Convolutional Neural Network with Self-Attention (CNN-SAM)

The CNN-SAM model [14] applies 1D convolutional layers to extract local frequency-dependent features from the impedance spectrum. A custom self-attention mechanism (SAM) is introduced after the convolutional blocks to capture global dependencies and assign importance weights to relevant features. The outputs from convolution and attention are combined via pooling and passed to fully connected layers for regression. This hybrid design enhances interpretability and improves feature learning beyond standard CNNs.

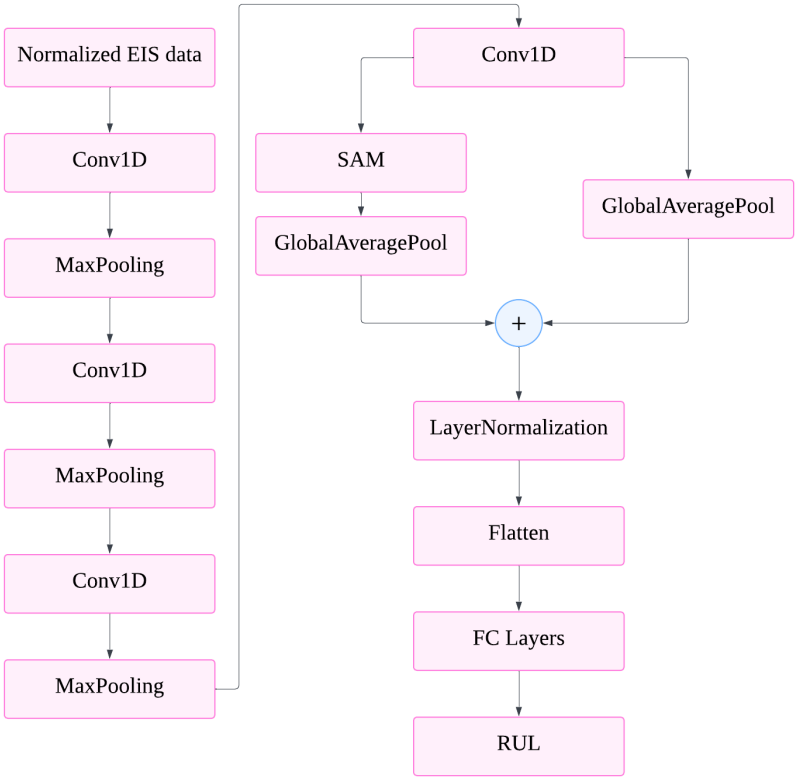


Figure 3.6 Architecture of CNN-SAM model

iii. Long Short-Term Memory (LSTM)

The LSTM model consists of stacked recurrent layers (256, 128, 64 units) designed to capture sequential dependencies across the EIS frequency domain. Dropout regularization was used to prevent overfitting. Following the recurrent layers, dense layers were applied before the regression output. This design leverages LSTM's strength in modeling long-range dependencies within sequential data.

iv. CNN + LSTM

This hybrid model integrates convolutional and recurrent layers. CNN blocks first extract local features from the impedance spectrum, which are then fed into stacked LSTMs to capture sequential dependencies. The combined approach leverages CNN's ability to detect localized patterns and LSTM's capability to model temporal relationships, leading to richer feature representation.

v. **CNN + LSTM + Attention**

In this variant, an attention mechanism is added after the CNN-LSTM backbone. Specifically, multi-head self-attention (MHA) layers are employed to highlight the most informative sequence elements when learning from impedance data. The attention-enhanced sequence representation is then pooled and passed through fully connected layers for regression. This improves model interpretability and performance by adaptively weighing important features.

vi. **CNN + LSTM + Transformer**

This architecture extends the CNN-LSTM model by integrating Transformer encoder blocks. CNN layers capture local features, LSTM layers capture sequential dependencies, and Transformer layers use multi-head attention along with feed-forward networks and residual connections. to capture global dependencies across the spectrum. The output is pooled and passed through dense layers for regression. This model combines the strengths of convolution, recurrence, and self-attention.

vii. **CNN + Transformer**

The CNN-Transformer model removes recurrent layers and directly combines CNN feature extraction with Transformer encoder blocks. CNNs first extract local patterns, then Transformer layers capture long-range relationships using multi-head attention and feed-forward networks. Compared to LSTM-based hybrids, this model benefits from parallel computation and more effective global dependency modeling.

3.2.4 Experiment setup

The dataset was divided into training and testing sets based on cell identifiers. Training set includes cells 25C01–25C04, 35C01, and 45C01. While testing set includes cells 25C05–25C08, 35C02, and 45C02 following [20]. This split ensures that the models are evaluated on unseen operating conditions and cells.

The end-of-life (EOL) cycle was defined as the cycle at which the capacity of a cell dropped to 80% of its initial capacity. For most cells, the EOL cycle was determined directly from the capacity measurements. However, for cell 25C04, the recorded

capacity did not fall below the 80% threshold. Following previous work in [21], its EOL cycle was fixed at 114 cycles to ensure consistency with the literature.

The Remaining Useful Life (RUL) was then calculated as the number of cycles between the current cycle and the EoL.

$$\text{RUL} = Cycle_{\text{EoL}} - Cycle_{\text{current}}$$

A Spearman correlation analysis was performed to investigate the relationship between EIS features and the RUL of the batteries. The Spearman correlation coefficient is defined as

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)},$$

Where n is the number of paired observations and d_i is the difference between the ranks.

The analysis in Figure 3.7 showed the spearman correlation of EIS data with RUL at each frequency point. The left plot is the real part of EIS, and the right plot is the negative part of EIS data. The frequency of EIS data is increasing from left to right. The more red or blue color shows stronger correlation of the EIS value towards RUL, where red means stronger positive correlation and blue means stronger negative correlation. As a result, the first 30 frequencies of the imaginary component of the impedance spectrum exhibited a strong negative correlation with RUL. This result suggests that as the battery approaches its end of life, the magnitude of the imaginary impedance at these frequencies increases, making them potential predictors of degradation.

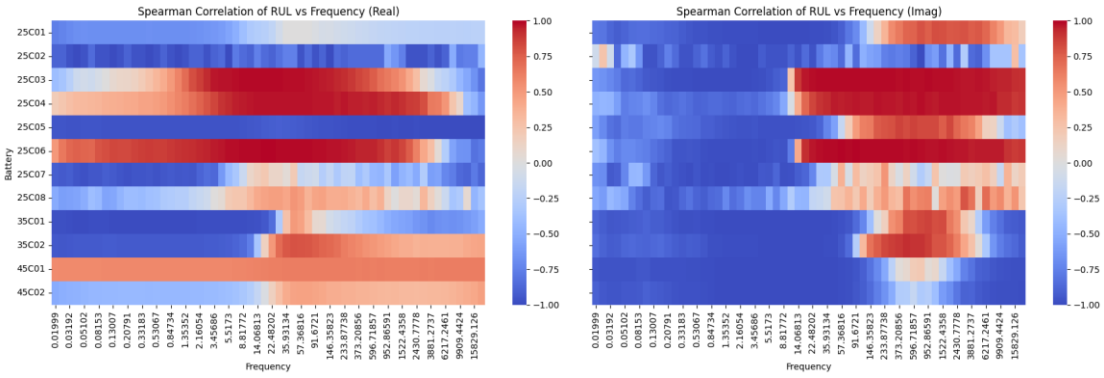


Figure 3.7 Spearman correlation analysis

To evaluate model performance under different input representations, three types of EIS features were extracted and used:

1. Full spectrum (Real + Imag): using the whole EIS spectrum with both real and imaginary parts of the impedance at all frequency points.
2. Imaginary only (Imag): only the imaginary part of the impedance.
3. Lower-half Imaginary (Half-Imag): only the imaginary impedance at the lower half (the first 30 frequencies) of the frequency range.

This design allows assessment of how different impedance components and frequency ranges affect RUL prediction performance.

3.3 Evaluation metrics

To evaluate the performance of the models, both classification and regression metrics were employed.

3.3.1 Classification metrics

Accuracy measures the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

Precision indicates the proportion of correctly predicted positive samples out of all samples predicted as positives:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Sensitivity) measures the proportion of correctly identified positive samples among all actual positives:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Specificity evaluates the proportion of correctly identified negative samples among all actual negatives:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

F1 score is the harmonic mean of precision and recall, offering a balanced metric for classification performance:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.3.2 Regression metrics

Root Mean Squared Error (RMSE) is a metric that quantifies the average magnitude of prediction errors by taking the square root of the mean of the squared differences between predicted and actual values. It provides a measure of how closely a model's predictions match the observed data, with lower values indicating better accuracy.:

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

Mean Absolute Error (MAE) measures the average absolute difference between actual and predicted values:

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

Coefficient of Determination (R^2) evaluates how well the predictions approximate the actual values:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

An R^2 value close to 1 indicates high predictive accuracy, while values near or below 0 indicate poor performance.

Chapter 4 Result Analysis

4.1 SOC prediction experiment results

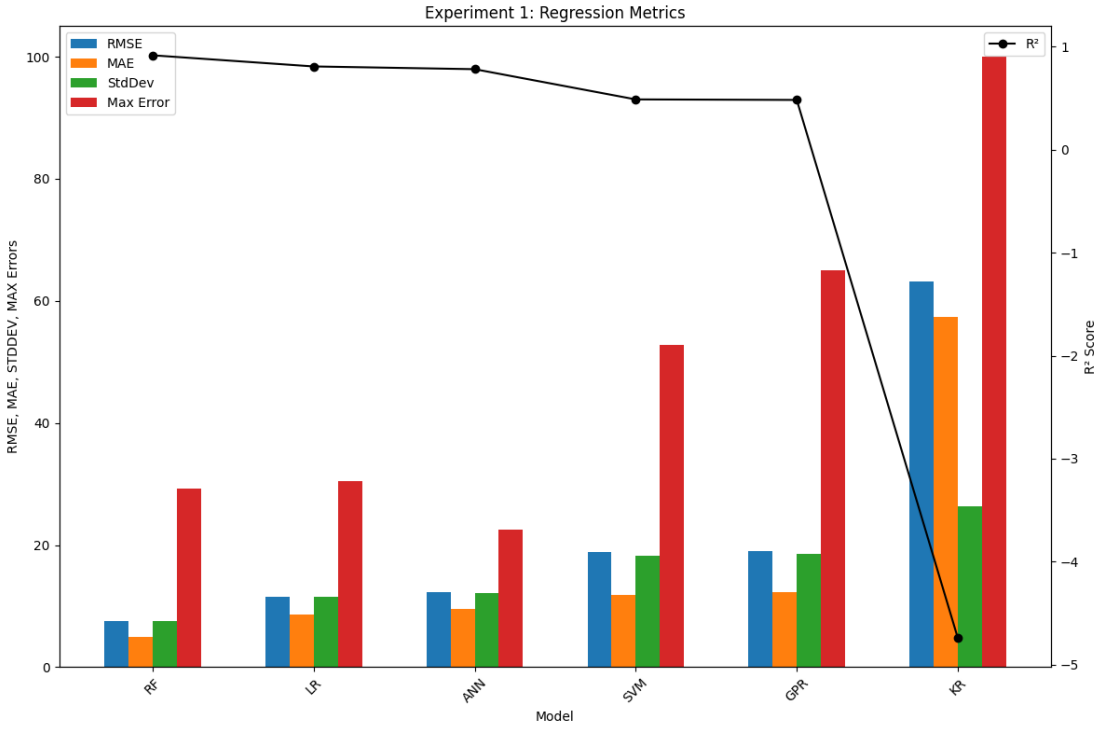


Figure 4.1 Experiment 1 full EIS frequency range regression SOC prediction result graph

Model	MAE	RMSE	R^2	StdDev	Max Error
RF	4.9567	7.5635	0.9178	7.5634	29.2387
LR	8.6615	11.5089	0.8096	11.5066	30.4726
ANN	9.4828	12.2886	0.7836	12.1070	22.5890
SVM	11.8805	18.8464	0.4894	18.2601	52.8337
GPR	12.2574	18.9421	0.4842	18.4905	64.9640
KR	57.4339	63.2048	-4.7425	26.3855	100.0000

Table 4.1 Experiment 1 full EIS frequency range regression SOC prediction result

From the result, the random forest achieved the highest R^2 score with the lowest standard deviation, indicating it performed better than other models. It is followed by linear regression and artificial neural network, which has similar R^2 scores of 0.8096 and 0.7836. This proves the robustness of linear regression and ANN in SOC prediction with the given hyperparameters.

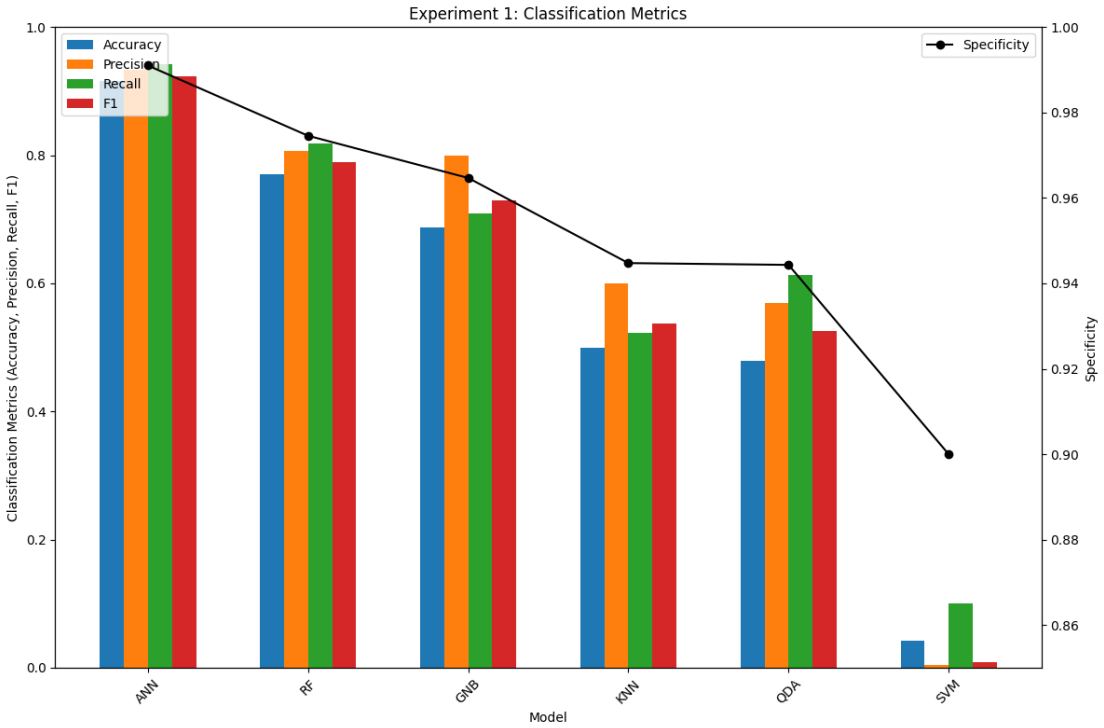


Figure 4.2 Experiment 1 full EIS frequency range SOC classification result graph

Model	Accuracy	Precision	Recall	F1	Specificity
ANN	0.9167	0.9333	0.9425	0.9234	0.9910
RF	0.7708	0.8071	0.8189	0.7890	0.9746
GNB	0.6875	0.8002	0.7086	0.7291	0.9647
KNN	0.5000	0.6000	0.5225	0.5373	0.9448
QDA	0.4792	0.5691	0.6133	0.5263	0.9443
SVM	0.0417	0.0042	0.1000	0.0080	0.9000

Table 4.2 Experiment 1 full EIS frequency range SOC classification result

This result of SOC classification using full EIS frequency range of impedance with classification showed that ANN has the best performance with highest F1 score compared to other models. This result can be explained by the ANN's ability to effectively capture complex nonlinear relationships in the impedance data. The poor performance of SVM may be due to unsuitable hyperparameter settings, which indicate that hyperparameter fine tuning is important to control the robustness of a model.

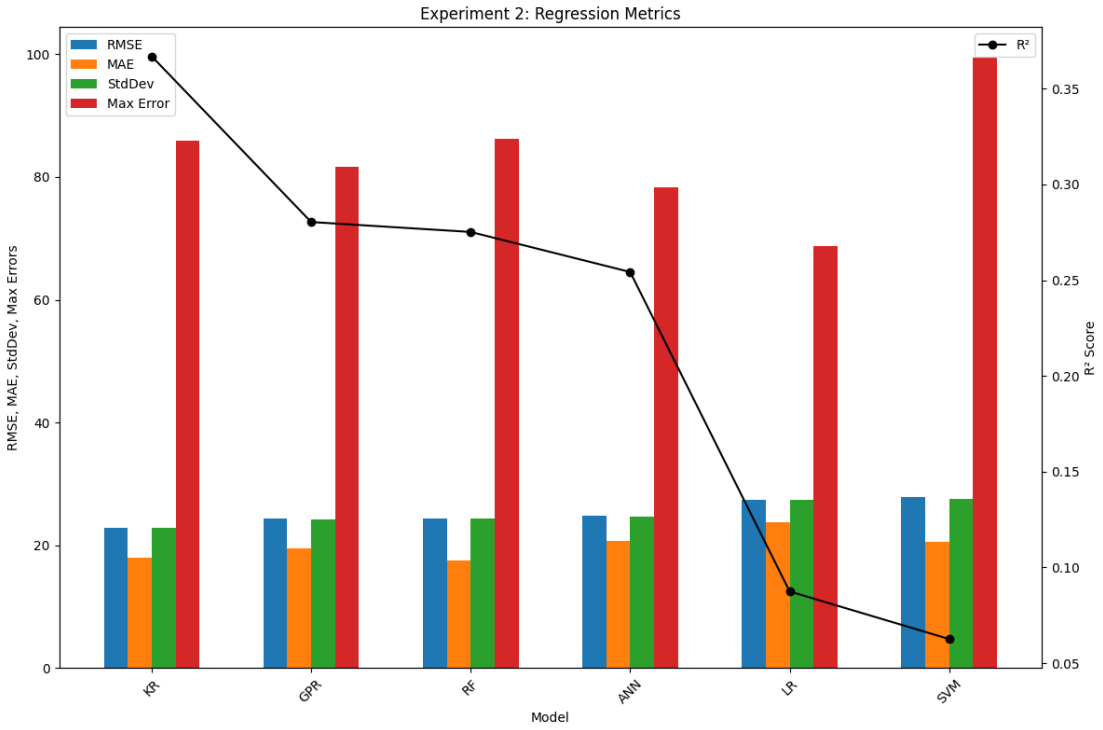


Figure 4.3 Experiment 2 individual frequency regression SOC prediction result graph

Model	MAE	RMSE	R^2	StdDev	Max Error
KR	17.9738	22.8460	0.3670	22.8219	85.9879
GPR	19.4713	24.3580	0.2804	24.2926	81.6593
RF	17.5255	24.4456	0.2752	24.4354	86.2609
ANN	20.7085	24.7957	0.2543	24.6786	78.3676
LR	23.8158	27.4301	0.0874	27.3979	68.7287
SVM	20.6067	27.8010	0.0626	27.5418	99.4350

Table 4.3 Experiment 2 individual frequency regression SOC prediction result

From the results obtained, it can be concluded that using individual frequency-impedance pairs, none of the models were able to predict the SOC accurately. This may be due to the insufficient information provided by a single frequency point, which limits the model's ability to capture the overall electrochemical behavior.

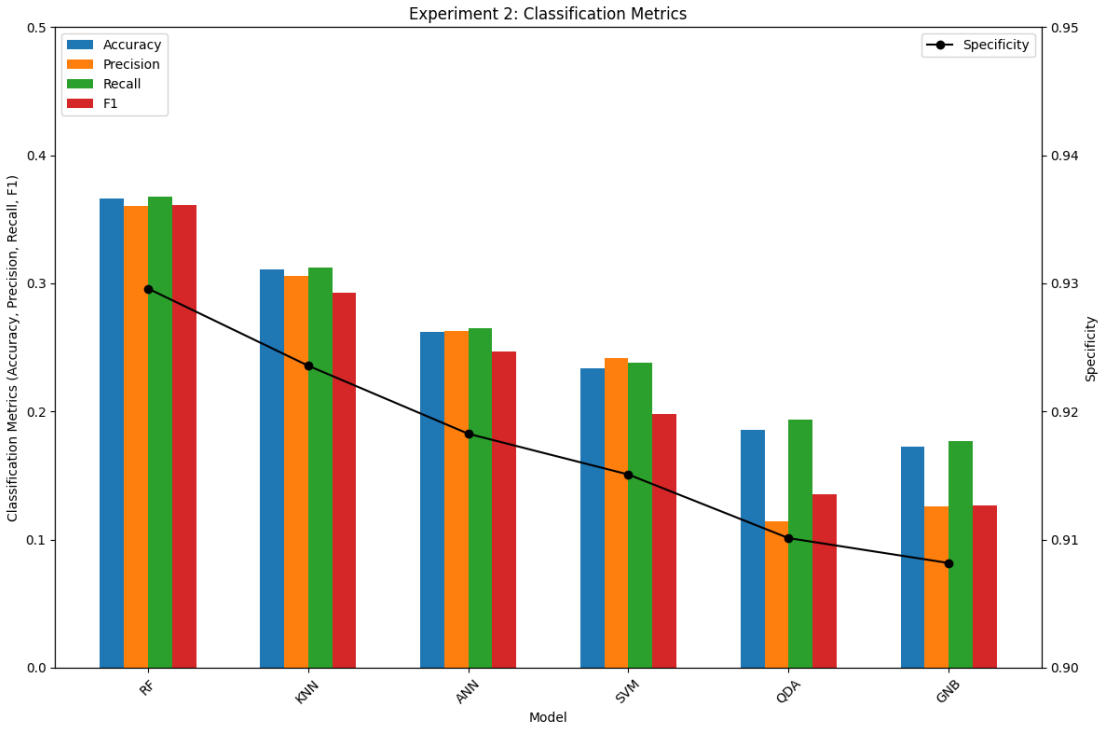


Figure 4.4 Experiment 2 individual frequency SOC classification result graph

Model	Accuracy	Precision	Recall	F1	Specificity
RF	0.3661	0.3606	0.3676	0.3615	0.9296
KNN	0.3110	0.3061	0.3123	0.2925	0.9236
ANN	0.2619	0.2632	0.2650	0.2466	0.9183
SVM	0.2336	0.2417	0.2381	0.1983	0.9151
QDA	0.1860	0.1143	0.1936	0.1357	0.9101
GNB	0.1726	0.1257	0.1769	0.1264	0.9082

Table 4.4 Experiment 2 individual frequency SOC classification result

The poor performance of SOC estimation in the classification models further supports the conclusion that individual frequency-impedance pairs are unsuitable for accurate SOC prediction.

Given ANN achieved third highest R^2 score in regression SOC prediction and top F1 score in SOC classification in experiment 1, the model demonstrated strong overall performance. As a result, this project proceeded with profiling and optimizing the model.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 298)               8642

 dense_1 (Dense)             (None, 275)               82225

 dense_2 (Dense)             (None, 1)                 276

=================================================================
Total params: 91143 (356.03 KB)
Trainable params: 91143 (356.03 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.5 Model summary for ANN regression model in experiment 1

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 295)               8555

 dense_4 (Dense)             (None, 30)                8880

 dense_5 (Dense)             (None, 10)                310

=================================================================
Total params: 17745 (69.32 KB)
Trainable params: 17745 (69.32 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4.6 Model summary for ANN classification model in experiment 1

Figure 4.5 and Figure 4.6 presented the model summaries of the ANN developed for regression and classification-based SOC prediction in Experiment 1. The summaries indicated that the ANN regression model consists of 91,143 trainable parameters with a total size of 356.03 KB, making it significantly larger than the classification model. This increase in size is primarily due to a higher number of neurons in the hidden layers. In contrast, the classification model contains only 17,745 parameters and occupies 69.32 KB of memory, which met the project's objective of developing a lightweight model suitable for deployment on resource-constrained embedded systems. Thus, pruning and quantization techniques were applied to the regression model to make the

model size lower than 200KB and make it compatible with embedded system requirements.

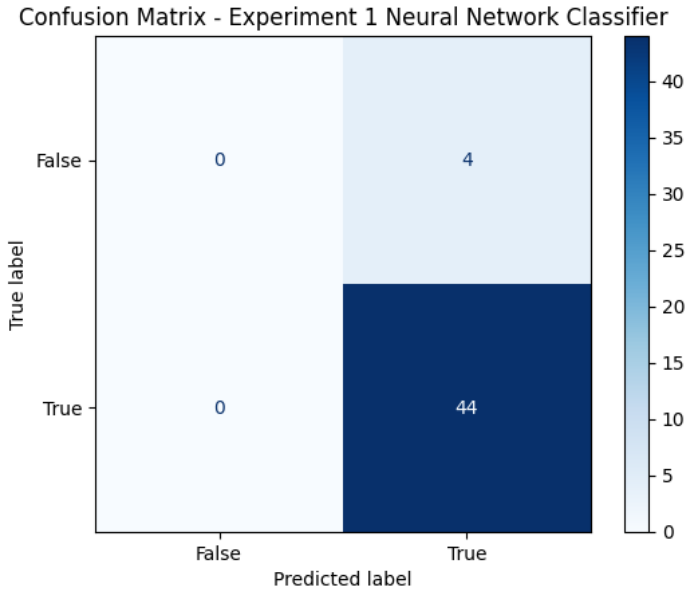


Figure 4.7 Confusion matrix of ANN classification model after pruning and quantization

Model	Accuracy	Precision	Recall	F1	Specificity
ANN	0.9167	0.9333	0.9425	0.9234	0.9910
ANN – Optimized	0.9167	0.9333	0.9425	0.9234	0.9910

Table 4.5 Performance comparison of ANN and optimized ANN for classification

Figure 4.7 presents the confusion matrix of the ANN classification model after applying pruning and quantization. Table 4.5 summarizes the detailed performance metrics, comparing the original ANN and ANN-Optimized (after pruning and quantization) models in terms of accuracy, precision, recall, F1-score, and specificity.

It can be observed that the optimized ANN achieves performance almost identical to the baseline ANN, indicating that pruning and quantization do not significantly degrade the classification capability of the model.

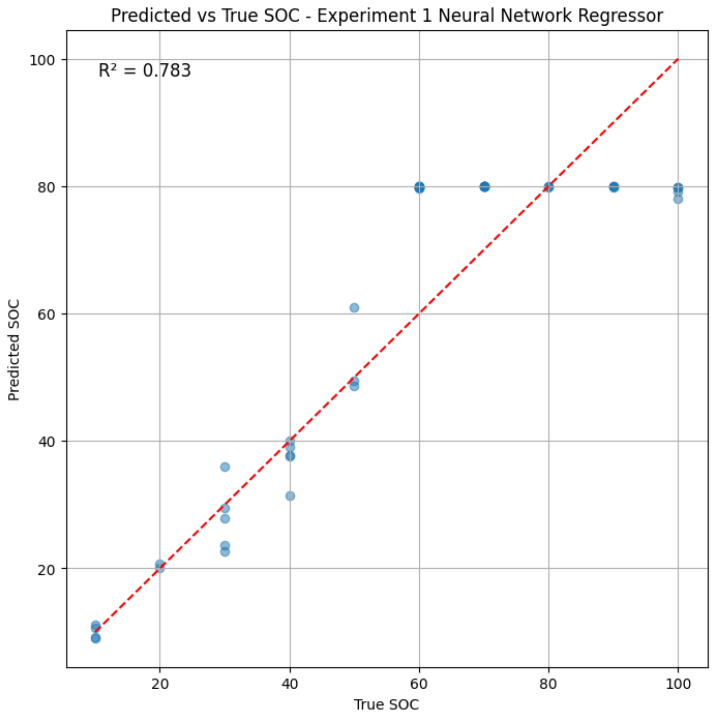


Figure 4.8 Predict vs True SOC of ANN regression model after pruning and quantization

Model	MAE	RMSE	R^2	StdDev	Max Error
ANN	9.4828	12.2687	0.7836	12.1070	22.7176
ANN – Optimized	9.5025	12.2850	0.7831	12.1081	22.0378

Table 4.6 Performance comparison of ANN and optimized ANN for regression

Figure 4.8 presents the predicted versus true SOC of the ANN regression model after pruning and quantization, with detailed regression metrics given in Table 4.6. Both the baseline and optimized ANN models achieve very similar regression accuracy, confirming that predictive capability is well preserved.

In addition to maintaining performance, pruning and quantization result in a substantial reduction in model size. The Experiment 1 NN classifier decreased from estimated 71.5 KB to 25.3 KB, achieving a 64.6% reduction, while the Experiment 1 NN regressor was reduced from estimated 358.1 KB to 100.3 KB, corresponding to a 72.0% reduction. These results highlight the efficiency gains enabled by quantization, allowing the models to run more efficiently on memory-constrained edge devices.

4.2 RUL prediction results

The R^2 score of RUL prediction was recorded as follows:

Model	25°C	35°C	45°C
EIS-GPR [20]	0.87	0.75	0.92
ANN	0.90	0.88	0.75
CNN-SAM	0.86	0.44	-0.02
LSTM	0.22	-0.03	-0.19
CNN+LSTM	-0.09	0.22	-0.07
CNN+LSTM+Attention	0.44	0.39	0.62
CNN+LSTM+Transformer	0.31	0.20	0.17
CNN+Transformer	-0.12	0.11	0.86

Table 4.7 Experiment with real and imaginary EIS data from all frequencies

Model	25°C	35°C	45°C
EIS-GPR [20]	0.87	0.75	0.92
ANN	0.51	0.73	0.90
CNN-SAM	0.60	0.81	0.69
LSTM	0.40	-0.07	-0.04
CNN+LSTM	0.01	-0.38	-0.69
CNN+LSTM+Attention	-3.09	0.40	0.92
CNN+LSTM+Transformer	-4.78	0.78	0.98
CNN+Transformer	0.10	0.52	0.94

Table 4.8 Experiment with imaginary EIS data from all frequencies

Model	25°C	35°C	45°C
EIS-GPR [20]	0.87	0.75	0.92
ANN	-0.13	0.88	0.90
CNN-SAM	-0.72	0.98	0.96
LSTM	0.14	0.00	-0.33
CNN+LSTM	-0.05	0.52	0.57
CNN+LSTM+Attention	-0.90	0.42	0.81
CNN+LSTM+Transformer	0.31	0.26	0.88
CNN+Transformer	-3.56	0.91	0.84

Table 4.9 Experiment with imaginary EIS data from first 30 frequencies

From the results, it can be concluded that ANN was highly competitive with GPR. Using all frequencies, ANN achieved the best performance at 25°C and 35°C, with R^2 scores of 0.90 and 0.88, respectively. In contrast, CNN-SAM showed mixed performance; however, in experiments using only imaginary EIS data, it demonstrated better generalization compared to the other models. In contrast, the LSTM model performed the worst across all input features, indicating that time-series–based approaches may not be well-suited for this dataset or that the temporal dependencies in the data are not strong enough for LSTM to capture effectively.

To continue the investigation, ANN and CNN-SAM were selected for further analysis due to their competitive performance in the previous experiments. The training set was updated to include batteries 25C03, 25C05, 25C06, 35C01 and 45C01, while the testing set consisted of 25C01, 35C02 and 45C02 following previous studies in [22]. The imaginary part of EIS data across all frequencies was selected as input features since both models perform good with the selected feature. The next step involves applying quantization to both models to evaluate how this affects their performance.

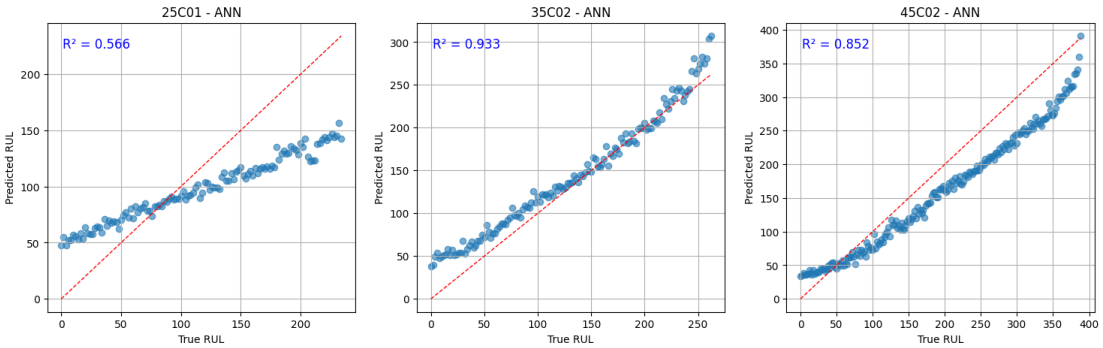


Figure 4.9 RUL prediction result of ANN model

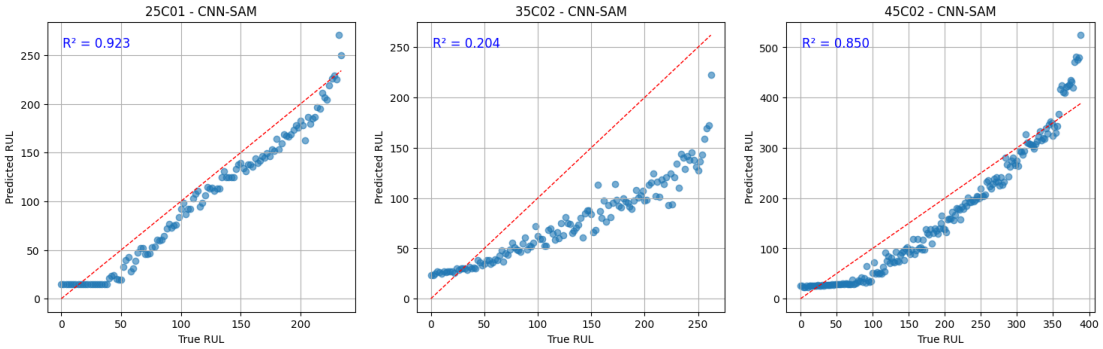


Figure 4.10 RUL prediction result of CNN-SAM model

Model	**25°C**	**35°C**	**45°C**
EIS-GPR [20]	0.87	0.75	0.92
CNN-BiLSTM [22]	0.85	0.86	0.94
ANN	0.57	0.93	0.85
CNN-SAM	0.92	0.20	0.85

Table 4.10 Comparison of R^2 performance of the proposed ANN and CNN-SAM models with previous studies at different operating temperatures

After applying the selected training and testing sets, the performance of ANN and CNN-SAM improved notably. ANN achieved R^2 scores of 0.57, 0.93 and 0.85, while CNN-SAM reached 0.92, 0.20, and 0.85 across the respective conditions. The R^2 score and ANN and CNN-SAM has improved for battery at 25°C and 45°C. This improvement is likely due to the more representative training set, which better captured variations across different batteries and operating conditions, allowing the models to generalize more effectively. However, the CNN-SAM R^2 score for 35°C has dropped.

The next step involves applying quantization to these models to assess how reducing model precision impacts both performance and efficiency.

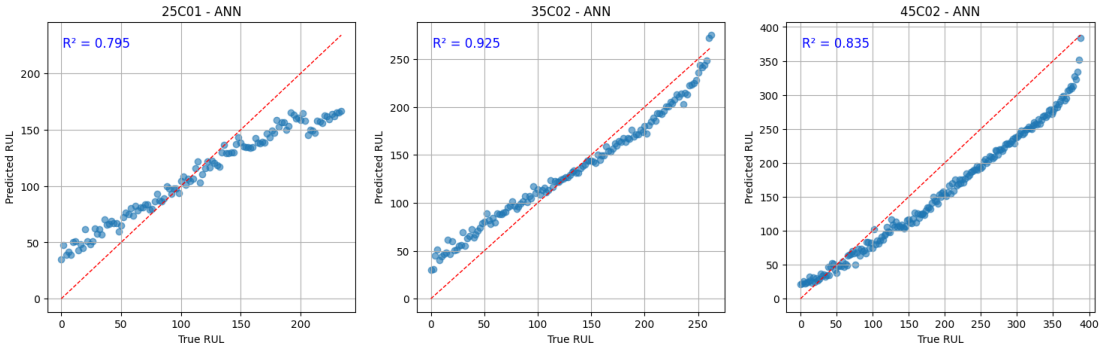


Figure 4.11 RUL prediction result of ANN model after pruning and quantization

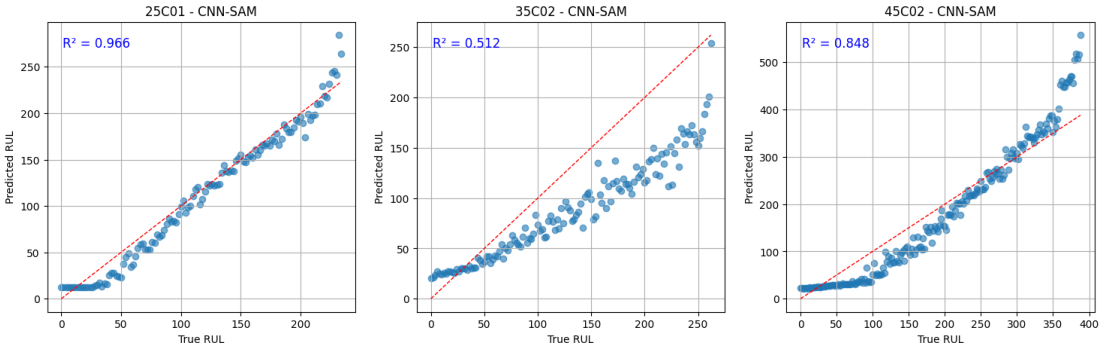


Figure 4.12 RUL prediction result of ANN model after pruning and quantization

Batteries	ANN		CNN-SAM	
	Original	Pruned + Quantized	Original	Pruned + Quantized
25C01	0.5663	0.7947	0.9233	0.9656
35C02	0.9327	0.9250	0.2041	0.5119
45C02	0.8520	0.8346	0.8500	0.8481

Table 4.11 Performance comparison of ANN and CNN-SAM models on individual test batteries at different temperatures (25 ℃, 35 ℃, and 45 ℃), before and after pruning and quantization.

After applying pruning and 8-bit quantization to both the ANN and CNN-SAM models, the results indicate that these optimization techniques effectively reduced model size without significantly compromising predictive accuracy as shown in Table 4.11.

Pruning involves removing less significant weights from the model, resulting in a smaller more and efficient structure. In this project, pruning was applied to increase sparsity from 50% to 80% over the training epoch. This process resulted in a reduction of the model sizes. These reductions are indicative of the efficiency gains achieved through pruning, which eliminates less important parameters without substantially affecting the model's performance. In this case, the pruning method has increased the R^2 score of battery 25C01 for ANN and 35C02 for CNN-SAM because pruning removed less important weights, which helped prevent overfitting and allowed the models to generalize better.

Quantization was then applied to the pruned models. This technique lower the precision of the model's weights and activations from 32-bit floating number to 8-bit integers, leading to further reductions in model size and potential improvements in inference speed. Despite the reduced numerical precision, the R^2 scores remained largely unchanged or improved slightly across the tested batteries, suggesting that both models are robust to reduced numerical precision. For instance, the ANN model's R^2 scores showed only a slight decrease across the tested batteries, while the CNN-SAM model maintained nearly identical performance before and after quantization.

In terms of model size, the ANN model decreased from 260 kB to 26 kB, and the CNN-SAM model reduced from 1679 kB to 158.32 kB. By reducing model size and computational requirements, these techniques enable the deployment of ML models on resource-limited devices without significantly sacrificing predictive accuracy. This is particularly beneficial for applications in edge computing, where storage and processing capabilities are limited.

Chapter 5 Conclusion

This project showed that machine learning algorithms can be effectively used and optimized to estimate battery states using Electrochemical Impedance Spectroscopy (EIS) data. For State of Charge (SOC) estimation, Random Forest achieved strong regression performance, while Artificial Neural Networks (ANN) excelled in classification tasks when trained on the full-frequency EIS spectrum. In contrast, models relying on individual frequency–impedance pairs showed poor accuracy, confirming the importance of comprehensive spectral features for reliable predictions.

For Remaining Useful Life (RUL) estimation, both ANN and CNN-SAM models were explored. The results showed that these deep learning approaches can capture degradation trends effectively, with pruning and quantization further improving model efficiency. Specifically, pruning helped reduce overfitting by removing redundant weights, while quantization significantly compressed model size (e.g., ANN reduced from 260 kB to 26 kB) without substantial accuracy loss. These optimizations make the models lightweight and suitable for deployment on embedded systems, which is essential for real-time applications in BMS.

Overall, this study highlights that combining EIS data with machine learning and model optimization techniques offers a promising pathway toward efficient, real-time SOC and RUL monitoring in lithium-ion batteries. Future work may include extending the framework to State of Health (SOH) estimation, integrating hybrid physics-informed and data-driven models, and validating performance on larger, more diverse datasets to ensure robustness across different battery chemistries and operating conditions.

REFERENCES

[1] Z. Nozarijouybari and H. K. Fathy, "Machine learning for battery systems applications: Progress, challenges, and opportunities," *J Power Sources*, vol. 601, p. 234272, May 2024, doi: 10.1016/J.JPOWSOUR.2024.234272.

[2] K. Movassagh, A. Raihan, B. Balasingam, and K. Pattipati, "A Critical Look at Coulomb Counting Approach for State of Charge Estimation in Batteries," *Energies 2021, Vol. 14, Page 4074*, vol. 14, no. 14, p. 4074, Jul. 2021, doi: 10.3390/EN14144074.

[3] A. M. R. Al Hadi, C. Ekaputri, and M. Reza, "Estimating the state of charge on lead acid battery using the open circuit voltage method," *J Phys Conf Ser*, vol. 1367, no. 1, Nov. 2019, doi: 10.1088/1742-6596/1367/1/012077.

[4] W. Zhou, Y. Zheng, Z. Pan, and Q. Lu, "Review on the Battery Model and SOC Estimation Method," *Processes 2021, Vol. 9, Page 1685*, vol. 9, no. 9, p. 1685, Sep. 2021, doi: 10.3390/PR9091685.

[5] Y. Bai, B. Yan, C. Zhou, T. Su, and X. Jin, "State of art on state estimation: Kalman filter driven by machine learning," *Annu Rev Control*, vol. 56, p. 100909, Jan. 2023, doi: 10.1016/J.ARCONTROL.2023.100909.

[6] D. Obuli Pranav, P. S. Babu, V. Indragandhi, B. Ashok, S. Vedhanayaki, and C. Kavitha, "Enhanced SOC estimation of lithium ion batteries with RealTime data using machine learning algorithms," *Sci Rep*, vol. 14, no. 1, pp. 1–17, Dec. 2024, doi: 10.1038/S41598-024-66997-9;SUBJMETA=166,639,987;KWRD=ELECTRICAL+AND+ELECTRONIC+ENGINEERING,ENGINEERING.

[7] Z. Ren and C. Du, "A review of machine learning state-of-charge and state-of-health estimation algorithms for lithium-ion batteries," *Energy Reports*, vol. 9, pp. 2993–3021, Dec. 2023, doi: 10.1016/J.EGYR.2023.01.108.

[8] F. Zhao, Y. Guo, and B. Chen, "A Review of Lithium-Ion Battery State of Charge Estimation Methods Based on Machine Learning," *World Electric Vehicle Journal*, vol. 15, no. 4, p. 131, Mar. 2024, doi: 10.3390/wevj15040131.

[9] L. Wei, Z. Ma, C. Yang, and Q. Yao, "Advances in the Neural Network Quantization: A Comprehensive Review," *Applied Sciences 2024, Vol. 14, Page 7445*, vol. 14, no. 17, p. 7445, Aug. 2024, doi: 10.3390/APP14177445.

[10] P. Coscia *et al.*, "A comparative study of neural network pruning strategies for industrial applications," *Front Comput Sci*, vol. 7, p. 1563942, Apr. 2025, doi: 10.3389/FCOMP.2025.1563942.

[11] S. T. Bakenhaster and H. D. Dewald, "Electrochemical impedance spectroscopy and battery systems: past work, current research, and future opportunities," *J*

Appl Electrochem, vol. 55, no. 7, pp. 1657–1681, Jul. 2025, doi: 10.1007/S10800-025-02273-6/FIGURES/9.

[12] H. S. Magar, R. Y. A. Hassan, and A. Mulchandani, "Electrochemical Impedance Spectroscopy (EIS): Principles, Construction, and Biosensing Applications," *Sensors 2021, Vol. 21, Page 6578*, vol. 21, no. 19, p. 6578, Oct. 2021, doi: 10.3390/S21196578.

[13] N. Meddings *et al.*, "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review," *J Power Sources*, vol. 480, p. 228742, Dec. 2020, doi: 10.1016/J.JPOWSOUR.2020.228742.

[14] J. Li, S. Zhao, M. S. Miah, and M. Niu, "Remaining useful life prediction of lithium-ion batteries via an EIS based deep learning approach," *Energy Reports*, vol. 10, pp. 3629–3638, Nov. 2023, doi: 10.1016/J.EGYR.2023.10.030.

[15] I. Babaeiyazdi, A. Rezaei-Zare, and S. Shokrzadeh, "State of charge prediction of EV Li-ion batteries using EIS: A machine learning approach," *Energy*, vol. 223, p. 120116, May 2021, doi: 10.1016/J.ENERGY.2021.120116.

[16] E. Buchicchio, A. De Angelis, F. Santoni, and P. Carbone, "Lithium-Ion Batteries state of charge estimation based on electrochemical impedance spectroscopy and convolutional neural network," 2022.

[17] S. J. Ojukwu, S. Maheshwari, R. Shafik, A. Yakovlev, and M. Mamlouk, "AI-Driven Battery State-of-Charge Estimation using Electrochemical Impedance Spectroscopy," *2023 International Symposium on the Tsetlin Machine, ISTM 2023*, 2023, doi: 10.1109/ISTM58889.2023.10454954.

[18] E. Buchicchio, A. De Angelis, F. Santoni, P. Carbone, F. Bianconi, and F. Smeraldi, "Battery SOC estimation from EIS data based on machine learning and equivalent circuit model," *Energy*, vol. 283, p. 128461, Nov. 2023, doi: 10.1016/J.ENERGY.2023.128461.

[19] D. Pietro Pau and A. Aniballi, "Tiny Machine Learning Battery State-of-Charge Estimation Hardware Accelerated," *Applied Sciences 2024, Vol. 14, Page 6240*, vol. 14, no. 14, p. 6240, Jul. 2024, doi: 10.3390/APP14146240.

[20] Y. Zhang, Q. Tang, Y. Zhang, J. Wang, U. Stimming, and A. A. Lee, "Identifying degradation patterns of lithium ion batteries from impedance spectroscopy using machine learning," *Nat Commun*, vol. 11, no. 1, pp. 1–6, Dec. 2020, doi: 10.1038/S41467-020-15235-7;SUBJMETA.

[21] C. Parsons, A. Amin, and P. Guptasarma, "Performance Classification and Remaining Useful Life Prediction of Lithium Batteries Using Machine Learning and Early Cycle Electrochemical Impedance Spectroscopy Measurements," Aug. 2024, Accessed: Sep. 19, 2025. [Online]. Available: https://arxiv.org/pdf/2408.03469

[22] Z. Liu, Y. Sun, Y. Li, Y. Liu, Y. Chen, and Y. Zhang, "Lithium-ion battery health prognosis via electrochemical impedance spectroscopy using CNN-BiLSTM

model," *J Mater Inf 2024;4:9.*, vol. 4, no. 2, p. N/A-N/A, Jun. 2024, doi: 10.20517/JMI.2024.09.

[23] M. Wang, Z. Sui, and L. Zhang, "State-of-Health Estimation of Lithium-Ion Batteries based on EIS and CNN-Transformer Network," *15th Global Reliability and Prognostics and Health Management Conference, PHM-Beijing 2024*, 2024, doi: 10.1109/PHM-BEIJING63284.2024.10874815.

[24] Y. He and L. Xiao, "Structured Pruning for Deep Convolutional Neural Networks: A Survey," *IEEE Trans Pattern Anal Mach Intell*, vol. 46, no. 5, pp. 2900–2919, May 2024, doi: 10.1109/TPAMI.2023.3334614.

[25] H. Cheng, M. Zhang, and J. Q. Shi, "A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations," Aug. 2023, doi: 10.1109/TPAMI.2024.3447085.

[26] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, "DepGraph: Towards Any Structural Pruning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2023-June, pp. 16091–16101, 2023, doi: 10.1109/CVPR52729.2023.01544.

[27] X. Ding *et al.*, "ResRep: Lossless CNN Pruning via Decoupling Remembering and Forgetting," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4490–4500, 2021, doi: 10.1109/ICCV48922.2021.00447.

[28] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 14908–14917, 2021, doi: 10.1109/CVPR46437.2021.01467.

[29] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. Van Gool, "Revisiting Random Channel Pruning for Neural Network Compression," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2022, pp. 191–201. doi: 10.1109/CVPR52688.2022.00029.

[30] B. Li, B. Wu, J. Su, and G. Wang, "EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning," 2020, pp. 639–654. doi: 10.1007/978-3-030-58536-5_38.

[31] Y. Feng, A. Lipani, F. Ye, Q. Zhang, and E. Yilmaz, "Dynamic Schema Graph Fusion Network for Multi-Domain Dialogue State Tracking," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2022, pp. 115–126. doi: 10.18653/v1/2022.acl-long.10.

[32] X. Ma, G. Fang, and X. Wang, "LLM-Pruner: On the Structural Pruning of Large Language Models," May 2023, Accessed: Sep. 11, 2024. [Online]. Available: http://arxiv.org/abs/2305.11627

[33] S. Liu *et al.*, "Sparse Training via Boosting Pruning Plasticity with Neuroregeneration," Jun. 2021, Accessed: Sep. 11, 2024. [Online]. Available: http://arxiv.org/abs/2106.10404

[34] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A Simple and Effective Pruning Approach for Large Language Models," Jun. 2023, Accessed: Aug. 27, 2024. [Online]. Available: https://arxiv.org/abs/2306.11695v3

[35] V. Sanh, T. Wolf, and A. M. Rush, "Movement Pruning: Adaptive Sparsity by Fine-Tuning," *Adv Neural Inf Process Syst*, vol. 2020-December, May 2020, Accessed: Sep. 10, 2024. [Online]. Available: https://arxiv.org/abs/2005.07683v2

[36] J. M. Corchado *et al.*, "A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration," *Electronics 2022, Vol. 11, Page 945*, vol. 11, no. 6, p. 945, Mar. 2022, doi: 10.3390/ELECTRONICS11060945.

[37] M. Nagel *et al.*, "A White Paper on Neural Network Quantization," Jun. 2021, Accessed: Sep. 11, 2024. [Online]. Available: https://arxiv.org/abs/2106.08295v1

[38] Y. Li *et al.*, "BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction," Feb. 2021, Accessed: Sep. 11, 2024. [Online]. Available: http://arxiv.org/abs/2102.05426

[39] M. Chen *et al.*, "EfficientQAT: Efficient Quantization-Aware Training for Large Language Models," Jul. 2024, Accessed: Sep. 11, 2024. [Online]. Available: https://arxiv.org/abs/2407.11062v1

[40] J. Youn, J. Song, H.-S. Kim, and S. Bahk, "Bitwidth-Adaptive Quantization-Aware Neural Network Training: A Meta-Learning Approach," 2022.

[41] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, "Overcoming Oscillations in Quantization-Aware Training," *Proc Mach Learn Res*, vol. 162, pp. 16318–16330, Mar. 2022, Accessed: Sep. 11, 2024. [Online]. Available: https://arxiv.org/abs/2203.11086v2

[42] E. Buchicchio, A. De Angelis, F. Santoni, P. Carbone, F. Bianconi, and F. Smeraldi, "Dataset on broadband electrochemical impedance spectroscopy of Lithium-Ion batteries for different values of the state-of-charge," *Data Brief*, vol. 45, p. 108589, Dec. 2022, doi: 10.1016/J.DIB.2022.108589.

[43] B. Zaparoli Cunha, C. Droz, A. M. Zine, S. Foulard, and M. Ichchou, "A review of machine learning methods applied to structural dynamics and vibroacoustic," *Mech Syst Signal Process*, vol. 200, p. 110535, Oct. 2023, doi: 10.1016/J.YMSSP.2023.110535.

[44] A. Miller, J. Panneerselvam, and L. Liu, "A review of regression and classification techniques for analysis of common and rare variants and gene-environmental factors," *Neurocomputing*, vol. 489, pp. 466–485, Jun. 2022, doi: 10.1016/J.NEUCOM.2021.08.150.

[45] Y. Wu, D. Bai, K. Zhang, Y. Li, and F. Yang, "Advancements in the estimation of the state of charge of lithium-ion battery: a comprehensive review of traditional and deep learning approaches," *J. Mater. Inf. 2025, 5, 18.* , vol. 5, no. 2, p. N/A-N/A, Feb. 2025, doi: 10.20517/JMI.2024.84.

[46] I. H. Witten, E. Frank, and M. A. Hall, "Implementations: Real Machine Learning Schemes," in *Data Mining: Practical Machine Learning Tools and Techniques*, Elsevier, 2011, pp. 191–304. doi: 10.1016/B978-0-12-374856-0.00006-7.

[47] P. Whig, S. Kouser, A. B. Bhatia, and R. R. Nadikattu, "Prediction of breast cancer diagnosis using random forest classifier," *Computational Intelligence and Modelling Techniques for Disease Detection in Mammogram Images*, pp. 55–73, Jan. 2024, doi: 10.1016/B978-0-443-13999-4.00011-0.

[48] L. Barillaro, "Artificial Neural Networks," *Encyclopedia of Bioinformatics and Computational Biology*, pp. 141–145, Jan. 2025, doi: 10.1016/B978-0-323-95502-7.00122-6.

[49] Y. Shi, K. Yang, Z. Yang, and Y. Zhou, "Primer on artificial intelligence," *Mobile Edge Artificial Intelligence*, pp. 7–36, 2022, doi: 10.1016/B978-0-12-823817-2.00011-5.

APPENDIX

Poster

Apply and optimize machine learning algorithms for estimating battery health

Introduction

Accurate estimation of battery **State of Charge (SOC) and Remaining Useful Life (RUL)** is vital for safety and performance. Traditional methods like Coulomb counting and OCV suffer from errors and complexity. This project applies **machine learning with Electrochemical Impedance Spectroscopy (EIS)** to improve estimation and optimize models for embedded deployment.

Methods

EIS data from lithium-ion batteries was pre-processed and used to train models including **Random Forest, SVM, GPR, and ANN**. For RUL, advanced deep learning models such as **ANN and CNN-SAM** were tested. **Pruning** and **quantization** were applied to reduce model size for real-time use.

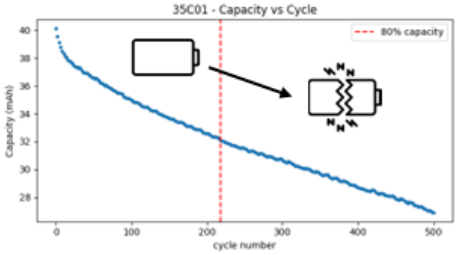
Results

SOC:
Random Forest gave the best regression ($R^2 \approx 0.92$); ANN achieved the highest classification accuracy (≈ 91.7%).
RUL:
ANN and CNN-SAM showed strong performance across different temperatures.
Optimization:
Model size reduced >80% (ANN: 260 kB → 26 kB; CNN-SAM: 1679 kB → 158 kB) with minimal accuracy loss.

The results demonstrate that full-spectrum EIS features capture battery electrochemical behavior more effectively than single-frequency features. ANN models excel in classification due to their ability to learn nonlinear relationships, while CNN-SAM leverages self-attention to identify key patterns in impedance data. Model optimization through pruning and quantization not only reduced storage and computation requirements but also helped mitigate overfitting, in some cases improving accuracy. These findings suggest that lightweight ML models can be effectively applied for **real-time battery monitoring in practical battery management systems (BMS)**.

SOC: How many energy left in the battery?

RUL: How long the battery can still be used?



Name: Chin Wai Yee
Supervisor: Dr. Lee Wai Kong

Data and code availability

The dataset used has been stated in the methodology. The code is available at https://github.com/Chin-Wai-Yee/SOC-and-RULPrediction-via-EIS.