Practical implementation of an advanced State of Charge Algorithm on a LiFePO4 and NMC battery pack.

by

Aseem Saxena

Department of Electronic & Electrical Engineering
University of Bath

Abstract

This project addresses the critical challenge of accurately estimating the State of Charge (SoC) in lithium-ion (LiFePO4) and nickel manganese cobalt (NMC) battery packs, a crucial factor in the rapidly evolving fields of electric vehicles and renewable energy storage. The need for a precise, robust, and adaptive SoC estimation method is even more pivotal for effective battery optimization. This research focuses on developing an advanced SoC estimation algorithm that accounts for charge/discharge cycles and temperature variations, two key factors that significantly influence battery performance and longevity. By incorporating these variables, the project aims to overcome the limitations of conventional estimation methods and provide a more accurate representation of a battery's remaining energy at any given time.

The methodology encompasses a comprehensive comparison of traditional SoC estimation techniques, such as extended Kalman filters, recursive least squares, and model-based adaptive observers, against machine learning approaches. These include artificial neural networks, Extreme gradient boosting algorithm, the AdaBoost algorithm as an ensemble method in machine learning, and Bayesian optimization with hyperparameter tuning. The comparative analysis showcases the strengths and weaknesses of each method and discusses the hybrid approaches that leverage both conventional and data-driven techniques.

Central to the project is the exploration of abundant raw BMS data and data preprocessing. Deep learning algorithms are used for modelling the complex, nonlinear behaviour of batteries. This approach promises to capture subtle patterns and relationships in battery performance data that traditional methods may overlook, potentially leading to significant improvements in SoC estimation accuracy and reliability.

The outcomes of this research have significance for battery management strategies and the potential to extend battery life, enhance safety, and improve the overall performance of battery-dependent systems. By exploring deeper into the non-conventional techniques to predict precise SoC estimates, this work aims to optimise energy usage in electric vehicles and grid-scale energy storage systems, ultimately supporting the broader transition to sustainable energy solutions.

Keywords: State of Charge, Li-ion batteries, NMC batteries, Charge/discharge cycle data, Temperature variations, BMS (Battery Management System), Deep learning, Extended Kalman filters, Recursive least squares, Artificial neural networks, Support vector machines, Random forests, Extreme gradient boosting algorithm, AdaBoost algorithm, Bayesian optimization, Hyperparameter tuning, Data-driven SoC estimation, Electric vehicles.

.

Contents

List of Abbreviations

SoC	State of Charge
Li-Ion	Lithium Ion
LiFePO4	Lithium Iron Phosphate
EV	Electric Vehicle
NMC	Nickel Manganese Cobalt
BMS	Battery Management System
RUL	Remaining Useful Life
OCV	Open Circuit Voltage
CC	Coulomb Counting
ANN	Artificial Neural Network
CVNN	Complex-valued neural networks
DNN	Deep Neural Networks
DL	Deep Learning
ML	Machine Learning
EKF	Extended Kalman Filter
RLS	Recursive Least Squares
SVM	Support Vector Machine
RF	Random Forest
GBM	Gradient Boosting Machine
LSTM	Long Short-Term Memory
BNN	Backpropagation Neural Network
RBFNN	Radial Basis Function Neural Network
NARXNN	Nonlinear Autoregressive Exogenous Neural Network
WNN	Wavelet Neural Network
GPR	Gaussian Process Regression
RVM	Relevance Vector Machine
UKF	Unscented Kalman Filter
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks

List of Symbols

*	Research questions

1. Introduction

The State of Charge of a battery showcases the available stored energy in a battery at any given point in time. Currently, we heavily rely on electronic devices for almost all our utilities. All the major portable devices sold in the global market primarily run on a Li-Ion battery. Smartphones, Laptops, Cameras, Power banks, peripheral devices, EVs, Drones, and Cordless power tools are some of the common examples. As of recent estimates, rechargeable batteries account for approximately 55% of global lithium consumption, followed by ceramics (21%), greases (6%), and other applications. (Alonso, Pineault and Nassar, 2023). The lithium-ion battery industry, which utilizes lithium carbonate and lithium hydroxide as key raw materials, constitutes a significant portion of the total cost of battery production, estimated to be between 50-70%(Y. Zhang et al., 2023).

In general, most of the devices demonstrate the need for reliable and robust SoC estimation for usage and charging optimisation. With huge consumption comes a greater need to determine accurate SoC for effective usage of batteries and charging infrastructure. This project tackles the need to develop SoC estimation systems with greater accuracy than their conventional counterparts.

The State of Charge of a battery cell represents the available capacity relative to its rated capacity, typically ranging from 0% to 100%. A SoC of 100% signifies a fully charged cell, while 0% indicates complete discharge.(Abdi et al., 2017)

In practical scenarios, the SoC is often maintained below 50%, prompting recharging once it reaches this threshold. As the cell ages, the maximum attainable SoC diminishes. Consequently, a 100% SoC in an aged cell may correspond to approximately 75%–80% SoC in a new cell.(Abdi et al., 2017)

The common SoC equation can be described as:

$$SoC(t) = \frac{Qremaining(t)}{Qmax(t)} * 100\,[\%]$$

Where SoC(t) is the State of Charge as a percentage at time t.

Qremaining(t) is the remaining charge or capacity in the battery at time t.

Qmax(t) is the maximum charge or capacity that the battery can hold at time t. (Saxena, n.d.)

1.1 Overview

The project aims for the development and practical implementation of an advanced State of Charge (SoC) estimation algorithm for LiFePO4 and NMC battery packs. The objective is to precisely estimate the SoC of a battery pack at any given time, by examining the charge/discharge cycle data with temperature variations. This practical implementation of the project has applications in several different LiFePO4-operated battery applications such as robots, electric vehicles and renewable energy storage, where accurate SoC estimation is crucial for battery usage and battery charging optimization. Accurate SoC prediction is essential for effective battery management, as it directly influences charge/discharge cycles, battery longevity, and overall system reliability.(Alonso, Pineault and Nassar, 2023)

1.2 Motivation and potential impact

The project is more engineering and research-based, where the need to find a precise non-conventional method to determine SoC and use the untapped potential of raw battery diagnostics data using machine learning and deep learning techniques to eliminate the need for an onboard bulky BMS system for every standalone robot.

This project transactions with the amalgamation of industry practices and standards with the research-oriented approach in academics. The study of effective prediction methods for determining SoC for batteries using data broadens the horizon of effective usage of batteries along with saving on energy cost by cutting on over-charging and also under-charging of batteries which if happened over a prolonged time, often results in battery malfunction.

Being able to accurately calculate SoC from data will help in preventing battery overuse or misuse, contributing to longer battery life. It will also help in achieving more safe and reliable operation by preventing overheating or overcharging.

The successful application of such algorithms to handle both LiFePO4 and NMC

batteries means it can be applied across a wide range of industries like automotive, grid storage and consumer electronics. They can also be integrated into BMS and help in retaining the performance and battery health. Notably, According to estimates, approximately 50 million tons of e-waste are produced annually, with batteries constituting a substantial portion of this waste(Alonso, Pineault and Nassar, 2023).

By improving battery life and performance, the cause also supports the broader goals of sustainability by potentially reducing e-waste linked with premature battery failure.

1.3 Problem description

The precise estimation of the SoC of a battery is still one of the most challenging aspects of battery management. Conventional methods like Coulomb counting and voltage-based techniques are unable to provide the best accuracy as they cannot take into account the nonlinear behaviour of batteries. Also, they are unable to precisely determine the initial charged state of the battery which then leads to a cumulative error development in the SoC calculation. The problem becomes more difficult when dealing with different battery chemistries like LiFePO4 and NMC. As they show distinct characteristics in terms of charging profiles, energy density, and temperature sensitivity. This poses a difficult bar to overcome to further optimise these batteries. Using ANNs on a large amount of charge/discharge cycle data will help in understanding the non-linear behaviour of batteries and factor it into the precise predictions of SoC without decoding the effects of these behaviours by classical means.

The use of ANNs requires careful data collection, model training, and validation to make sure that the algorithm is reliable and can be useful under different operating conditions. In terms of machine learning, this problem is called a Regression problem.

1.4 Aims and Objectives

- Design and implement machine learning models modelled for precise SoC estimation for LiFePO4 and NMC battery packs, considering battery-specific characteristics.

- Testing with a range of algorithms and neural networks (e.g., ANN, XGBoost, Bayesian Ridge) to select the best-performing algorithm.
- Improve the model's performance by experimenting with hyperparameters using cross-validation techniques.
- Validate the algorithm's performance against actual battery data, refining it based on validation outcomes.
- Fine-tune the algorithm to enhance accuracy and reliability based on feedback received during testing.

**For safe battery operation, how to determine a battery's state of charge (SoC) with accuracy*

2. SoC estimation (conventional and non-conventional methods)

2.1 History and advancements in SoC estimation techniques

Substantial development in SoC estimation has taken place over time in academics. The early developed methods included Coulomb counting which involves measurement of current flow to estimate charge. Later, when voltage-based techniques were developed, open-circuit voltage relations with SoC showed up.

These techniques have, however, been confronted by the challenge of sustaining their accuracy under changed conditions. ANN was adopted in more recent developments with the express purpose of enabling such fine-tuning of accuracy by realistically modelling complex non-linear behaviour exhibited by the batteries in several operational scenarios.

2.2 Conventional Methods for SoC estimation

Classical SoC estimation techniques, such as the Coulomb counting and the voltage-based methods, have some merits and demerits, for instance:

Advantage: simple and easy to implement; low computational cost.

For example, Coulomb counting finds wide applications due to its simple implementation, while voltage-based methods allow fast estimation under steady conditions.

$$SoC(t) = SoC(t0) + (\int I(t)\, dt) / Qrated$$

Where:

- SoC(t) is the State of Charge at time t
- SoC(t0) is the initial State of Charge
- I(t) is the current flowing in or out of the battery over time
- Qrated is the rated capacity of the battery
  (Plett, 2015)

There are also several very strong drawbacks to these methods. Coulomb counting, which calculates SoC by integrating the current over time, is straightforward but suffers from cumulative errors. These errors arise from inaccuracies in current measurement and initial SoC estimation, leading to drift over time(Zhao et al., 2014).

Voltage-based methods, such as the open-circuit voltage (OCV) approach, also encounter difficulties. The OCV method relies on the relationship between voltage and SoC, which can be nonlinear and influenced by factors such as temperature and battery ageing.(Homan et al., 2016).

$$SoC = f(OCV)$$

Where:

- SoC is the State of Charge
- f is a function that relates OCV to SoC
- OCV is the Open Circuit Voltage
  (Plett, 2015)

The Extended Kalman Filter (EKF) has been employed to combine current integration with voltage measurements, improving the robustness of SoC estimation.(Zhang et al., 2020). This approach accounts for the nonlinearities in battery behaviour and adjusts for parameter variations in real-time, enhancing accuracy significantly compared to Coulomb counting and simple voltage methods.(Wang et al., 2021)

2.3 Non-Conventional Methods for SoC estimation

Unconventional techniques include the use of ANN and fuzzy logic, which have been perfect approaches in comparison with more conventional ones for the estimation of SoC. They map complex and nonlinear dynamics of batteries, considering ageing effects and temperature variations under different load conditions. Also, ANN can be trained in tandem with other complex neural networks like CVNN, and DNN for precise SoC prediction in comparison with conventional models.

Fuzzy logic embeds rule-based decision-making to estimate SoC in uncertain or variable conditions, also it makes the estimator robust under diverse operating scenarios. There are some examples of running Fuzzy logic with ANN.

The combination of Fuzzy Logic and ANN allows for a more comprehensive analysis of the battery's state. Fuzzy Logic can handle the inherent uncertainties in battery behaviour, while ANNs can learn complex patterns from data, leading to improved accuracy in SoC estimation(Faisal et al., 2019)

As the battery ages or as environmental factors shift, the hybrid model can recalibrate its estimations based on new data inputs, ensuring that SoC predictions remain reliable.(Xu and Cen, 2021)

2.4 Comparison between conventional and non-conventional methods

Various issues plague the implementation of both Coulomb counting and voltage-based techniques for SoC estimation in a dynamic environment.

Coulomb counting seriously suffers from integration drift and requires the rest of the battery for an accurate estimate, most times impracticable. The better non-conventional approaches use ANNs for temperature, ageing, and load-variation modelling with greater accuracy.

There are several problems associated with ANNs: high computational cost and complexity while handling real-time variation and battery degradation. ANNs perform competently in problems where the execution of high-precision deterministic tasks is required.

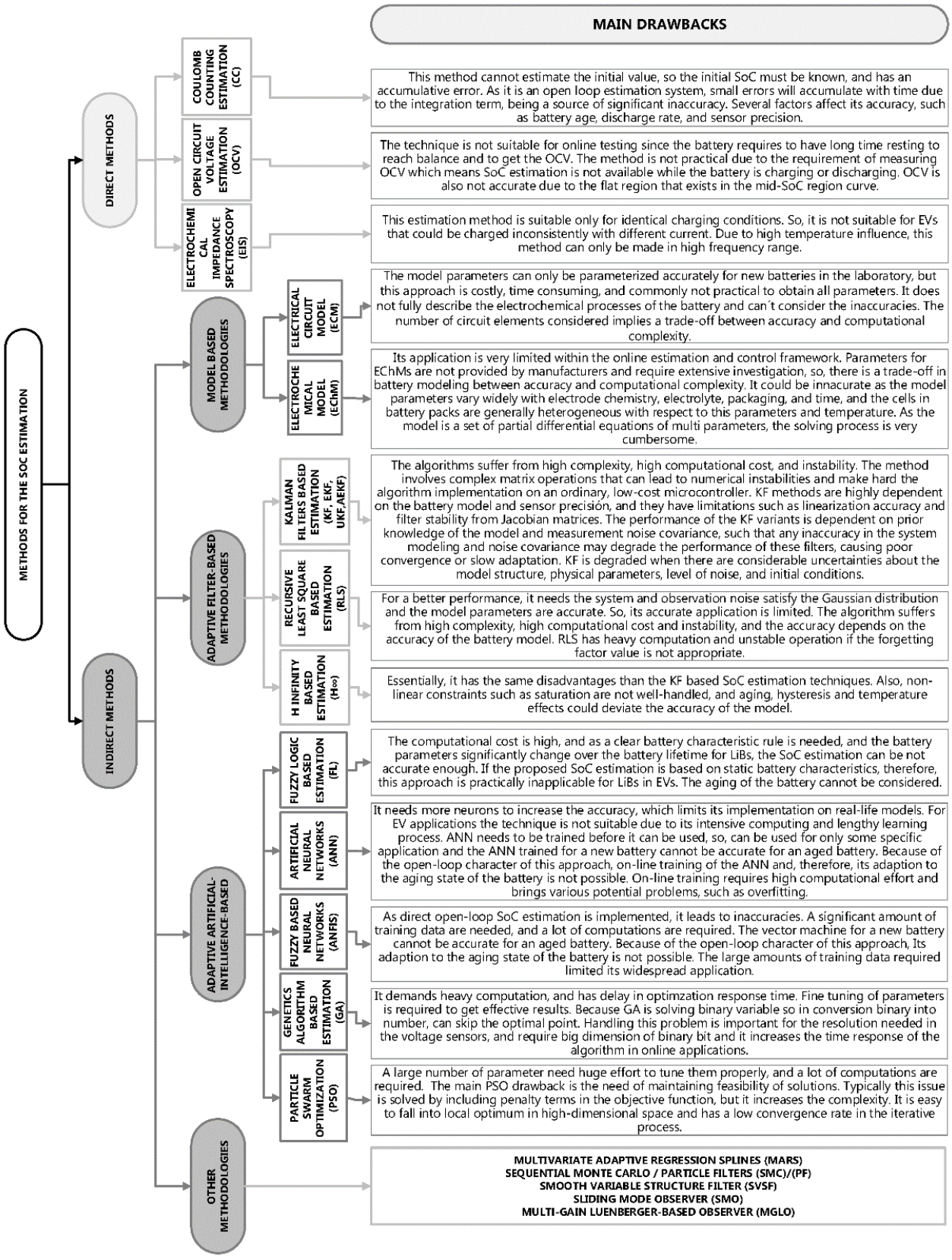


Figure 1. Summary of methods for the SoC estimation and main disadvantages.(Rivera-Barrera, Muñoz-Galeano and Sarmiento-Maldonado, 2017a)

2.5 Analysis of techniques for approach selection

The major considerations for the selection of SoC estimation methods can be described as:

Application Demands: Match the complexity of the method with the application. EVs require high accuracy and real-time performance; thus, hybrid approaches combining Kalman filters and machine-learning techniques are ideal (Maurya, Gawade and Zope, 2024). While simpler devices would suffice from simpler and conventional methods.

Resource Constraints: The need and availability for computational power and data can be varied through different application types. Edge devices work effectively on lightweight models, while cloud-connected systems with plentiful data can refer to high resource demands in techniques like algorithms such as XGBoost or deep learning. The cloud-connected systems leverage more complex algorithms, such as Kalman filters and machine learning approaches, to achieve higher accuracy in SoC estimation. These systems can handle the computational load associated with advanced techniques.(Li et al., 2022).

Battery Characteristics: The approach needs to adapt to battery chemistry and potentially, integration with BMS. LiFePO4 cells have flatter voltage curves, and more complex estimation will be required.(Fan et al., 2018). whereas NMC often benefits from more specific models while considering certain issues encountered with ageing patterns. They provide higher energy density and better performance at elevated temperatures. This exhibits a substantial voltage drop as they discharge, allowing for more accurate SoC estimation using voltage-based methods.(Yu et al., 2018)

Operational needs: These include real-time needs, regulatory compliance, and long-term adaptability. Fast-charging applications need quick algorithms; uses in safety-critical systems might require interpretable models; methods allowing adaptation to battery ageing offer sustained accuracy. The choice of estimation technique often depends on operational needs, such as accuracy, response time, and computational efficiency.

* W*hat are the current SOC estimation methods for lithium-ion batteries, specifically LiFePO4 and NMC batteries?*

3. SoC estimation methods for LiFePO4 and NMC battery

3.1 Battery Chemistry and Challenges in SoC estimation for LiFePO4 and NMC batteries

Some of the significant chemical features of LiFePO4 and NMC result in their flat voltage curves which makes voltage-based estimation a challenge and therefore influences the estimation of SoC. LiFePO4 batteries are also known for showcasing thermal stability, long cycle life, and safe operation. The relatively flat voltage profile during discharge complicates SoC estimation using voltage-based methods.(Moral et al., 2020).

In LiFePO4 batteries, the ANN model may need to incorporate additional features to account for the gradual voltage changes.(Z. Zhang et al., 2023)

NMC batteries provide higher energy density and better performance at elevated temperatures. The Open Circuit Voltage (OCV) and SoC relationship for NMC batteries are more sensitive, making them suitable for dynamic applications where rapid SoC estimation is required. However, NMC batteries experience substantial capacity drop over time, which must be accounted for while selecting SoC estimation algorithms.(Xu et al., 2017). ANNs can leverage the more sensitive OCV-SoC relationship to improve estimation accuracy

3.2 Industry Benchmarks

Consumer electronics and stationary energy storage systems aim for under 5% error of the SoC estimated by the technique, while applications like electric vehicles, require significantly higher accuracies and aim for less than 1% error on the estimated SoC.(Sarda et al., 2023)

Additional parameters that influence standard benchmarks are computational efficiency, re-adaptation to ageing, and performance in different operating conditions.

3.3 Selection of suitable approach (basis and advantages)

The approach was made by including multilayer neural networks with further developed algorithms. For the neural networks used, there were 3 and 4 layers combined with the XGBoost, AdaBoost, and Bayesian Ridge algorithms. This approach aims to capture complex nonlinear relationships in battery behaviour and hence may lead to better computation times because deep networks, with 4 layers, can model more complex patterns.

The system can use a combination of various algorithms, each at its point of strength for the different stages of SoC estimation. This can offer higher accuracy than the single-method approach by utilizing various advanced techniques in combination.

Diversity of methods brings robustness against various operating conditions hence dealing with uncertainty. Bayes Ridge could provide more accurate prediction confidence, which is useful in safety-critical applications like this.

XGBoost and AdaBoost can be applied to realize which parameters are of the most importance for SoC estimation, therefore probably yielding more effective sensor deployment in battery management.

3.4 Practical feasibility of approach

This methodology requires high computational resources, especially for applications needing real-time processing. Advanced microcontrollers or even dedicated processors may be used in the BMS to run these models efficiently. Cloud-based processing can be considered an option.

The multi-algorithm approach (neural networks, XGBoost, AdaBoost, Bayesian Ridge) could increase implementation complexity which can be tricky to tackle and might show model fit issues like under-fitting or over-fitting and requires knowledge in both machine learning and battery technologies for proper installation and maintenance.

These large models involve continuous data acquisition and periodic retraining to maintain accuracy throughout battery life. Such models face challenges in real-time

SoC estimation under fast-changing conditions.

It provides a great level of adaptability to changing battery characteristics. This could require retraining or recalibration periodically, thereby increasing maintenance burdens.

Increased up-front costs because of higher demands on hardware and development. The complex computations also make the BMS more power-consuming. While perhaps more accurate, this would require much more extensive testing and validation, especially for safety-critical applications.

Integration with existing systems requires changes to the existing BMS architecture. This allows for high potential in the accurate operation under various conditions of LiFePO4 and NMC batteries, which is enabled within the SoC estimation approach presented. This approach is more practically feasible for applications that require high accuracy and already possess high computation and resource maintenance architecture onboard.

4. Data Collection and Analysis

4.1 BMS Data Collection

The role of BMS is critical in battery operational data necessary to carry out an efficient and accurate SoC estimate. As it provides orderly and appropriate records of various important parameters, such as voltage, current, temperature, and charge/discharge cycles required to diagnose the status and performance of the battery with time.

The data consists of data points that are precisely stored and timestamped by the BMS within the reading frequency in microseconds. More than 30 Gigabytes of raw numerical data was extracted, which ran over the course of 4 months.

Since the data is acquired at a high rate and accurate value, it has characteristics such that all relevant changes and dynamic properties will be captured accordingly.

The voltage and current measurements with timestep and frequency in microseconds

serve as a crucial asset. BMS provided tracking of the bidirectional current that was flowing 'in' and 'out' of the battery and it also provided bidirectional voltage readings along with each cell voltage and the overall charge state of the battery.

The application requires a large database of raw BMS data for determining accuracy, and to learn the compact, complex, nonlinear nature of batteries, which involves temperature variations, load variation, and long-time degradation features.

The temperature data recorded in the present BMS is significant for modelling temperature-related behaviour, which enhances the reliability of the SoC algorithm under variant operational conditions for both LiFePO4 and NMC battery packs.

4.2 Data preprocessing and feature extraction

Data preprocessing of raw data from BMS is one of the most rigorous parts of the project, to ensure the appropriateness of the dataset for the estimation of SoC could be ascertained.

Given the size excess of 30 gigabytes collected over four months with measurements done at microsecond recording intervals-the first logical step was that of data cleansing.

For instance, 1 big charge/discharge cycle data file of size 12 Gigabytes was processed and had cascaded values of around 80 different types of recorded values with rows running well over 207 million. Due to the nature of the file, It could not be processed in more generic data analysis too like Microsoft Excel but required custom-built queries and scripts in SQL, R, and Python programming languages.

The issue of data check and validation before splitting it into training and testing datasets was faced frequently.

The data cleaning steps involved the removal of incomplete or corrupt records, and null values, and checking data uniformity with data cardinality.

Uniform time alignment for all the timestamps across the parameters: voltage, current, temperature, and charge/discharge cycles. Since the voltage and current readings are

quite varied, normalization involves scaling their value range. This step is significant to avoid biasing because of any parameter, which may act as an influential bias in the SoC estimation model.

Feature extraction was done to identify variables from the cleaned and normalized data preprocessing for the SoC estimation algorithms. Examples of such features extracted include the current direction of charge/discharge, bidirectional voltage, cell voltage variations, the sum of voltage across the battery, and temperature fluctuations.

The preprocessing and feature extraction steps resulted in an organized and quality dataset usable for both developing and testing the model.

4.1 Data Augmentation

Further, data augmentation was performed to fulfil this existing dataset for enhancement of the SoC estimation model by adding data points to counter data cardinality errors.

Extreme temperature fluctuations, multiple readings for the same battery module and deliberately left out from data obtained from the original BMS.

The augmented data filled in the gaps of the original dataset for conditions like rapid discharge or extreme weather conditions.

Data augmentation is necessary towards SoC estimation using non-conventional methods by making it robust and adaptable.

5. ANN with boosting algorithms (XGBoost, AdaBoost, and Bayesian Ridge)

5.1 ANN training

The input data is split into two parts, training data and test data in a 3:1 ratio. 75% of train data was allotted for training the model and the rest 25% of input data was kept for testing and validation for post-training analysis.

After feature extraction and data cleaning, the dataset had 2+ million rows and the model was breaking down frequently during the training. This led to splicing the dataset in half, i.e. 1+ million rows.

The input data consists of 4 different features type or time-stamped Bi-directional current, Bi-directional voltage, Cell voltage, and Temperature reading.

The output layer consist of 1 data type i.e. BMS measured SoC value for each

time-step.

Three similar models were created with each using the identical datasets.

1. ANN with XGBoost algorithm
2. ANN with Adaboost algorithm
3. ANN with Bayesian Ridge Regression algorithm

ANN with XGBoost algorithm

The model used an XGBRegressor with 42 random states. The neural model used 4 fully connected dense layers, along a Rectified Linear Unit as an activation layer (ReLU) as it showcases effectiveness in promoting faster training and better performance in deep neural networks.

```
# XGBoost model
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(train_x, train_y["RBATSOC"])

# xgb Predictions
xgb_pred = xgb_model.predict(test_x)

# Evaluate the model
mse = mean_squared_error(test_y["RBATSOC"], xgb_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 264.0972497565362
```

```
# Neural Network model
nn_model = Sequential()
#nn_model.add(Dropout(0.2))  # Add dropout layer
nn_model.add(Dense(units=64, activation='relu'))
#nn_model.add(Dropout(0.2))  # Add dropout layer
nn_model.add(Dense(units=32, activation='relu'))
#nn_model.add(Dropout(0.2))  # Add dropout layer
nn_model.add(Dense(units=16, activation='relu'))
nn_model.add(Dense(units=1))  # Output layer for regression
```

```
nn_model.compile(optimizer='adam', loss='mse')  # Choose appropriate optimizer and loss function
```

They passed into the model via 4-connected layers of neurons so that a connection between two neurons has an associated weight factor. This smooth adjustment of weights is how the network learns to minimize error between predicted and real outputs.

Three dropout layers were added along with 4 dense layers. They are a part of regularization techniques used in neural networks to prevent overfitting. It works by randomly 'deactivating' a fraction of neurons during the training process.

```
# Create callbacks for early stopping and reducing learning rate
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
 factor=0.2, patience=5, min_lr=0.0001)


# Train the neural network on train_x with callbacks
history = nn_model.fit(train_x, train_y, epochs=1000, batch_size=64, validation_split=0.2,

# Neural Network predictions
nn_pred = nn_model.predict(test_x)

# Evaluate both models
xgb_mse = mean_squared_error(test_y["RBATSOC"], xgb_pred)
nn_mse = mean_squared_error(test_y["RBATSOC"], nn_pred)

print("xgbBoost Mean Squared Error:", xgb_mse)
print("Neural Network Mean Squared Error:", nn_mse)
```

The model also used the callbacks for early stopping and reducing the learning rate as the model progressed to prevent over-fitting.

Similarly, the other two models were created with different sets of layer combinations and activation functions.

Below are the code snippets from the other two models:

```
# Load data
dataframe = pd.read_csv("Data/input_data.csv")
dataframe_labels = pd.read_csv("Data/RBATSOC.csv")
```

```
length_of_df = len(dataframe)

print("Length of the DataFrame:", length_of_df)

length_of_df2 = len(dataframe_labels)

print("Length of the DataFrame:", length_of_df2)
```

```
Length of the DataFrame: 1048575
Length of the DataFrame: 1048575
```

```
# Train-test split
train_x, test_x, train_y, test_y = train_test_spl
```

```
# AdaBoost model
ada_model = AdaBoostRegressor(random_state=42)
ada_model.fit(train_x, train_y["RBATSOC"])
```

Above is the Adaboost model.

Bayesian Ridge model is showcased below:

```
# Bayesian Regression model
bay_model = BayesianRidge()  # Initialize Bayesian Ridge model
bay_model.fit(train_x, train_y["RBATSOC"])  # Fit the model on the training data

# Predictions
bay_pred = bay_model.predict(test_x)  # Predict on the test data

# Evaluate the model
mse = mean_squared_error(test_y["RBATSOC"], bay_pred)  # Calculate Mean Squared Error
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 1334.9489062733562
```

5.2 Boosting algorithms and model refinement

XGBoost (Extreme Gradient Boosting)

The algorithm employs an ensemble of sequentially trained decision trees which attempt to rectify the errors made by their predecessors. This boosting algorithm is useful for large datasets since it minimizes error rates while reducing overfitting at the same time.

AdaBoost

For every iteration, AdaBoost adjusts weights for incorrectly predicted data points, making sure that it pays more attention to those tricky data points. This way shortcomings such as those arising from SoC estimation during rapid charging/discharging can be dealt with when AdaBoost combines with ANN.

Bayesian Ridge

Bayesian Ridge presents a probabilistic manner and can be united with ANNs, especially in well-known uncertain prediction fields. Through doing so, Bayesian Ridge prevents overfitting by applying penalties on too many complicated models hence there will be good bias and variance trade-off when estimating SoC.

Each model was then validated against each other as per the Mean Squared Error and not on accuracy. Since this is a regression problem and not a classifier problem.

5.3 Hyperparameters tuning

In the models, different numbers of dense layers, dropout layers, learning rates, number of epochs, patience, and batch sizes were varied throughout training and testing. Adam optimizer is used.

In addition, Boosting algorithms have their unique set of hyperparameters like, the number of estimators or trees, and maximum depth in XGBoost. While in AdaBoost, the number of epochs is significant

5.4 Testing and Validation of model fit (under-fitting or over-fitting) training

Once ANN trains the boosting algorithms, performance at this step needs to be evaluated to determine whether it suffers from underfitting or overfitting.

Underfitting is the condition when the model is not complex enough to learn the underlying structures in the data; therefore, it has high bias and poor predictive performance.

Overfitting means that models are getting too complex, reaching for noise and variance rather than general tendencies within a data set, therefore performing weakly when new data is received.

Metrics that compute the fitness of the model, such as Mean Absolute Error, Mean Squared Error, and R-squared are measured. Several different cases that can be used for testing the robustness of the model against overfitting, include the measured SoC values of a battery at various temperatures and charge/discharge rates to evaluate the

model generalization.

6. Results and Performance

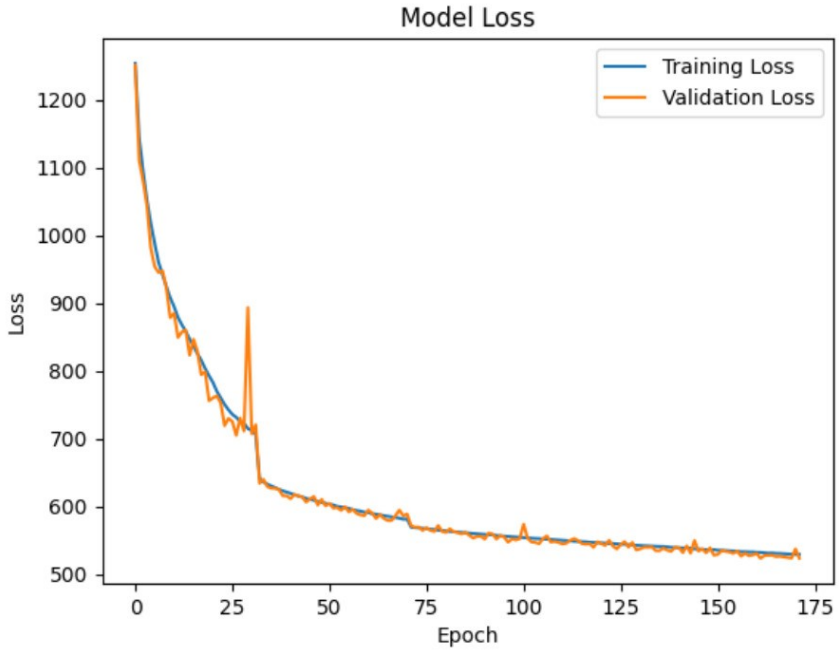
6.1 ANN model comparison with performance metrics

The metric used is Mean Squared Error (MSE) which describes precision and the overall performance of the model.

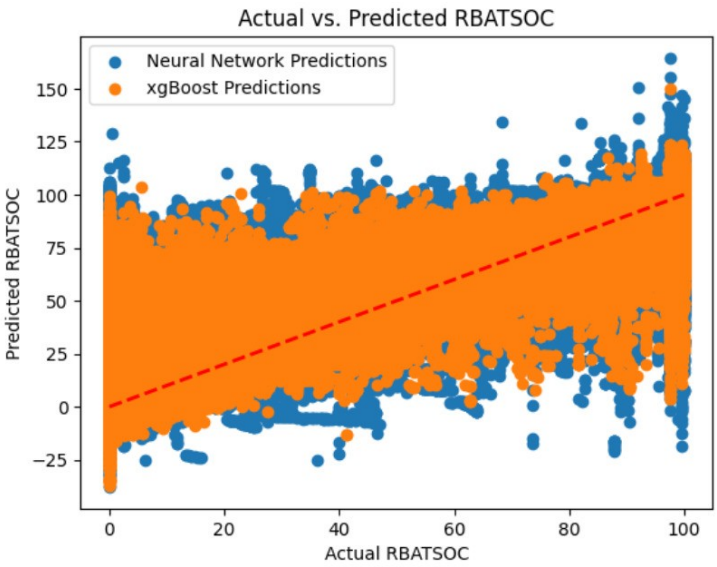
The training and validation loss curves are checked to make sure no overfitting or underfitting occurs.

Then, the performance of ANN is compared with boosting algorithms such as XGBoost and AdaBoost.

Model 1 : ANN with XGBoost algorithm

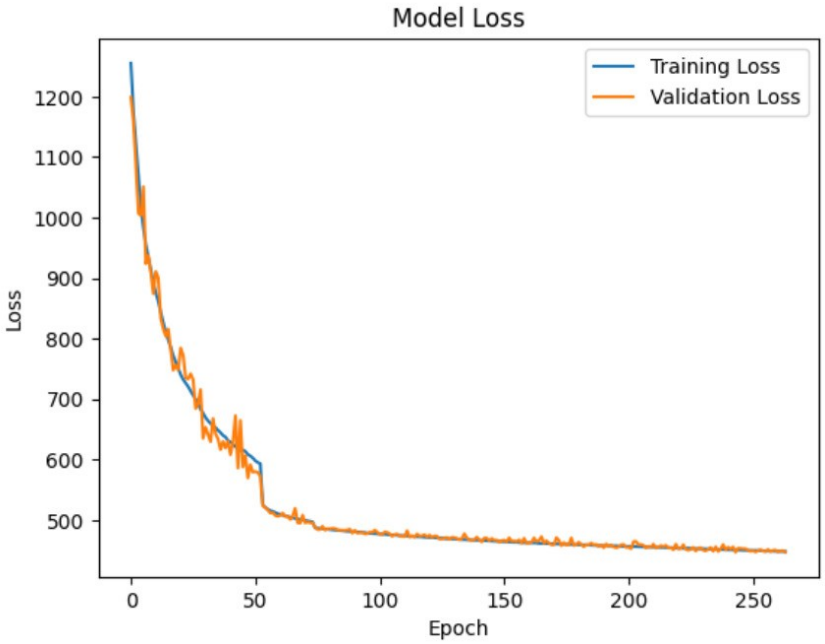


As per the above graph, The model ran up to 170 epochs and the MSE error or loss is still above 500. The ANN model is performing well in terms of generalization and is not showing signs of over-fitting and under-fitting.

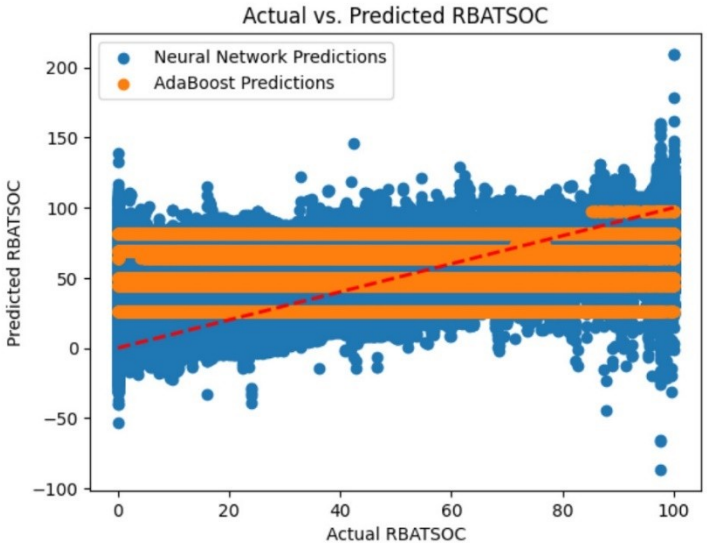


The above scatter chart shows the plot between ANN and XGBoost predictions with real SoC test data prediction from 0% to 100%. The plot points can be seen as adjusting towards the actual SoC curve, but overall, the model failed to encompass the true essence of actual SoC values.

Model 2 : ANN with Adaboost algorithm

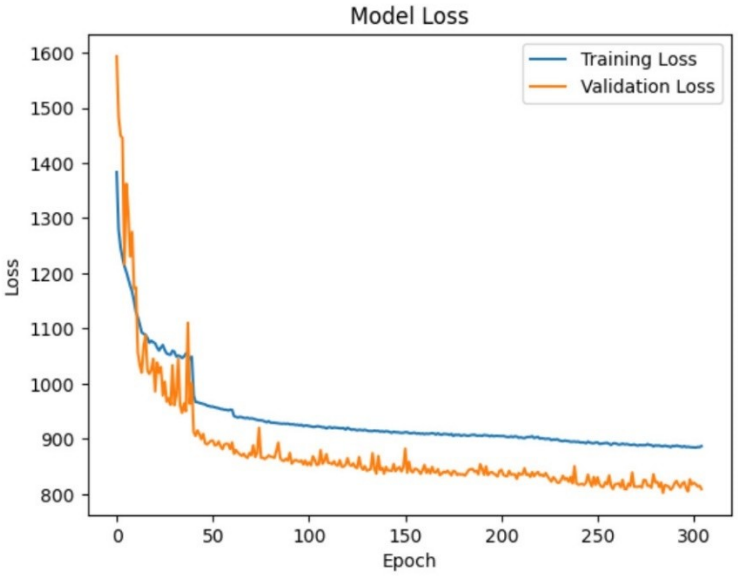


As per the above graph, The model ran up to 270 epochs and the MSE error or loss is below 500 but still above 400. The ANN model is performing better than XGBoost in terms of generalization and is not showing signs of over-fitting and under-fitting.



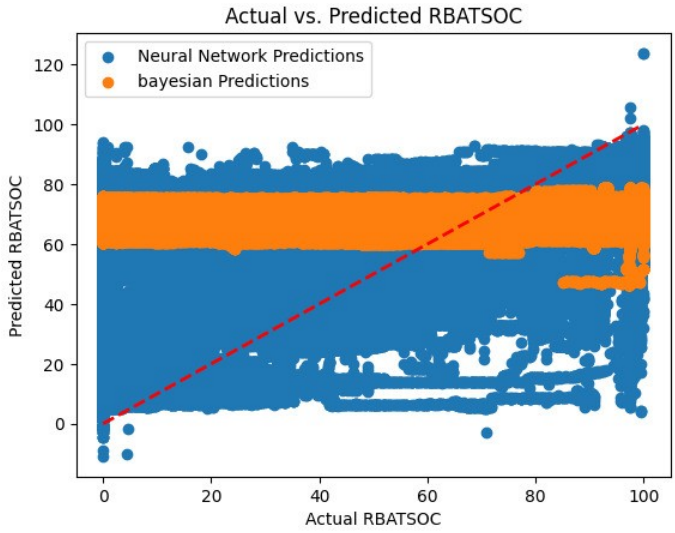
The above scatter chart shows the plot between ANN and AdaBoost predictions with real SoC test data prediction from 0% to 100%. The plot points can be seen as adjusting towards the actual SoC curve, and the Adaboost prediction is not overshooting the boundaries of actual SoC of 0% and 100%. But overall, the model failed to predict the actual SoC values with good accuracy.

Model 3 : ANN with Bayesian Ridge regression algorithm



As per the above graph, the model ran up to 310 epochs and the MSE error or loss is above 800. The ANN model is also not performing well in terms of generalization and is not showing signs of immediate over-fitting or under-fitting but not converging. This implies that the model is stopping early, This might be due to the regularization

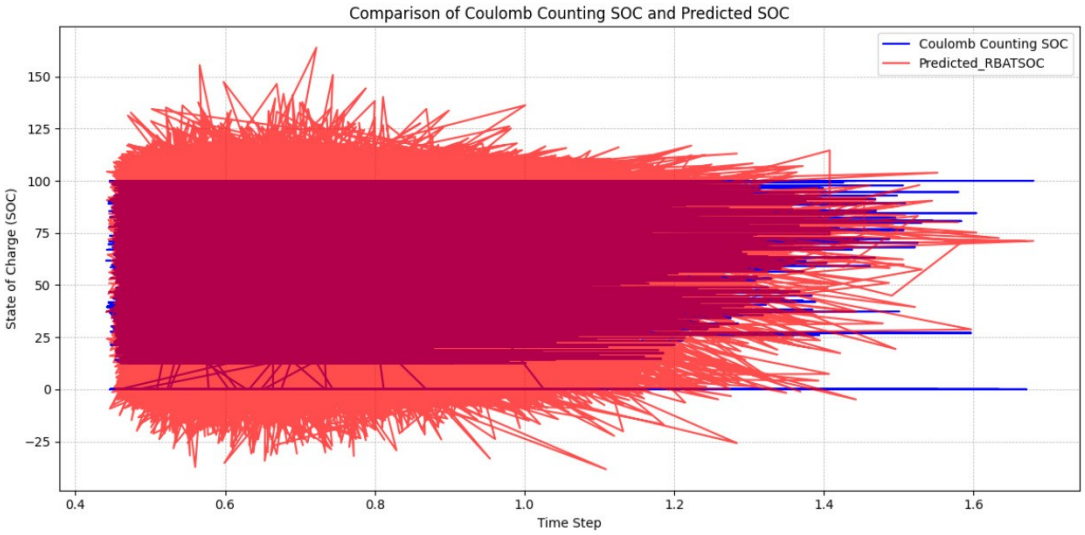
effects of using Dropout layers.



The above scatter chart shows the plot between ANN and Bayesian predictions with real SoC test data prediction from 0% to 100%. The plot points are not to be seen as adjusting towards the actual SoC curve, And the model failed to encompass the true essence of actual SoC values.

6.2 Comparison with Traditional Approach (such as Coulomb Counting)

In the comparison of performance comparison among the ANN-based technique and Coulomb counting. The comparatively better results among the three models were shown by XGBoost.



The Coulomb counting SoC lies on the 0% and 100% limit on the SoC axis. And the

XGBoost predicted SoC overshoots the boundaries by ± 25% in smaller time steps but converges inside the region as the time step increases. Therefore, the model fits better on the values which have a subsequent recording time step of more than 1.2 seconds.

Regardless, the model is not perfect and has a lot of room for improvement.

7. Discussions

7.1 Observation

The ANN models, in combination with boosting algorithms such as XGBoost, AdaBoost, and Bayesian Ridge, presented variable performance in effectively predicting the SoC.

To check accuracy in all models, the Mean Squared Error or MSE metric was used. The XGBoost model provided a very high MSE after 170 rounds of training; over 500 is not good. However, it generalized well without any trace of underfitting or overfitting. It had many difficulties in mapping the actual SoC curve.

Compared to other regression techniques, the AdaBoost algorithm had less than 500 MSE, generally above 400, in 270 epochs, and maintained the prediction within the limited capacity of SoC between 0% and 100%.

The Bayesian Ridge model applied a total of 310 epochs, where the highest MSE was over 800. The result shows a poor convergence because of dropouts in the layers and partial learning about the SoC behaviour.

Key takeaways

- All models failed to estimate the entire range of SoC, especially at the extreme points, e.g., 0% and 100%.
- In some cases, XGBoost overshoots values of SoC by ±25%, but reaches better convergence with time steps longer than 1.2 seconds.
- While AdaBoost outperformed others, it was still not competent in overall accuracy for SoC limits.
- Bayesian Ridge gave the poorest accuracy and failed to track the true SoC curve.

Among these, XGBoost and AdaBoost showed better generalization than Bayesian Ridge by avoiding major overfitting or underfitting problems.

7.2 Cases of Model Fit Improvement

There are several cases where the model fit improved after certain interventions:

- The XGBoost model performed better with time steps over >1.2 s, suggesting that the more frequent updates of State of Charge might allow for improved performance.
- The model AdaBoost demonstrated better adherence to the SoC boundaries with more epochs; hence, further iterations of training might improve performance in certain cases.
- Preprocessing techniques, like filtering noisy data, enhanced the input's quality-especially for the AdaBoost model since that needed much cleaner data for making predictions as best as possible.
- The model hyperparameter tuning performed had indeed enhanced the performance. This was quite an important tuning of learning rates and probabilities of dropout since it considerably reduced some underfitting errors found for both Bayesian Ridge and XGBoost models.

7.3 Potential Improvements in ANN

The results and observations show several ways through which ANNs' performance could be improved.

The architecture optimization can experiment with deeper networks or tune the number of neurons per layer to capture SoC relations with voltage, current and temperature.

Some of the regularization techniques include increasing the dropout rates and adding L1/L2 regularization.

Transfer learning i.e. pre-training the models on one type of battery and then fine-tuning on another, may indicate improvement in SoC estimation across various

battery chemistries.

8. Conclusion

8.1 Milestones

- No perfect accuracy has been achieved in any model, results were obtained for all developed ANN-based techniques in battery management, concerning model generalization.
- Among the models presented, only AdaBoost will satisfy, remaining within the bounds of SoC = 0% and SoC = 100% and avoiding extreme overfitting and underfitting problems.

8.2 Limitations Identified

Several limitations were identified:

- All the models gave low precision but with overestimation while estimating SoC, mainly at extreme values of 0% and 100%.

8.3 Technical Findings

- AdaBoost ranked best in generalization and meeting SoC boundaries compared to XGBoost and Bayesian Ridge.
- Increasing time intervals beyond 1.2 s showed that XGBoost was more effective, probably indicating the need for improvement in time series data processing to have better fits.
- The regularization techniques including dropout layers which do not overfit might have caused early stopping for the model at Bayesian Ridge.
-

9. Future Works

9.1 Model optimization

Model can be optimized through the tuning of hyperparameters like, learning rate, and the tuning of the layer and neuron count. Dropout and batch normalization can be tuned to avoid early stopping while ensuring overfitting is not achieved.

9.2 Diversification of Algorithms

Using different algorithms with an ANN is a viable option. Emphasis on hybrid approaches where ANN will be combined with ensemble learning methods other than boosting algorithms, such as random forests or stacking.

Looking into other non-artificial neural network models could be done with SVMs or Gaussian Processes, which may be more robust to outliers or noise, in general.

Further work may involve increasing the model's ability to learn time-series dependencies using RNNs or LSTM layers to predict output values over time.

9.3 Test Cases

The operation of these ANN models in different charge/discharge cycles, temperature variations, and under different ageing scenarios show how well they adapt and remain accurate with time.

Adding tests of redundancy will help in controlling noise or missing data, and enable estimates of the robustness and performance of models under less-than-ideal conditions.

This would also involve some kind of fault tolerance mechanisms so that a model would still make good predictions when some of the sensor data is unreliable or incomplete.

9.4 Integration with BMS

The models need deeper integration with the BMS for better usability in real-life scenarios.

Retraining the models will require continuous updating of the voltage, current, and temperature during operation from the collaborative data streams supplied by the BMS.

The real-time feedback loops between BMS and ANN models will enhance the learning and the model will update while running as the battery performance progresses, therefore, updating and enhancing SoC prediction accuracy with time.

Model generalization on a wide variety of battery chemistries, operating conditions, and system architectures with minimal loss in accuracy for its broader application.

Making use of Transfer learning techniques or making models flexible enough to adapt to new types of batteries or systems, makes this solution increasingly adaptable and scalable.

References

1. Abdi, H., Mohammadi-ivatloo, B., Javadi, S., Khodaei, A.R. and Dehnavi, E., 2017. Energy Storage Systems. Distributed Generation Systems [Online]. Elsevier, pp.333–368. Available from: https://doi.org/10.1016/B978-0-12-804208-3.00007-8

2. Alonso, E., Pineault, D. and Nassar, N.T., 2023a. Streamlined approach for assessing embedded consumption of lithium and cobalt in the United States. Journal of Industrial Ecology [Online], 27(1), pp.33–42. Available from: https://doi.org/10.1111/jiec.13337.

3. Alonso, E., Pineault, D. and Nassar, N.T., 2023b. Streamlined approach for assessing embedded consumption of lithium and cobalt in the United States. Journal of Industrial Ecology [Online], 27(1), pp.33–42. Available from: https://doi.org/10.1111/jiec.13337.

4. Faisal, M., Hannan, M.A., Ker, P.J. and Uddin, M.N., 2019. Backtracking Search Algorithm Based Fuzzy Charging-Discharging Controller for Battery Storage System in Microgrid Applications. IEEE Access [Online], 7, pp.159357–159368. Available from: https://doi.org/10.1109/ACCESS.2019.2951132.

5. Fan, B., Luan, X., Zhang, R., Niu, T. and Xie, Y., 2018. Research on SOC Estimation Algorithm for Lithium Battery Based on EKF Algorithm and Ampere-hour Integration Method. Proceedings of the 2017 2nd International Conference on Electrical, Control and Automation Engineering (ECAE 2017) [Online], 2017 2nd International Conference on Electrical, Control and Automation Engineering (ECAE 2017), Xiamen, China. Xiamen, China: Atlantis Press. Available from: https://doi.org/10.2991/ecae-17.2018.22

6. Homan, B., Smit, G.J.M., Van Leeuwen, R.P., Lei Zhu and De Wit, J.B., 2016. Validation of a predictive model for smart control of electrical energy storage. 2016 IEEE International Energy Conference (ENERGYCON) [Online], 2016 IEEE International Energy Conference (ENERGYCON), Leuven, Belgium. Leuven, Belgium: IEEE, pp.1–6. Available from: https://doi.org/10.1109/ENERGYCON.2016.7513872

7. Li, S., He, H., Wei, Z. and Zhao, P., 2022. Edge computing for vehicle battery management: Cloud-based online state estimation. Journal of Energy Storage [Online], 55, p.105502. Available from: https://doi.org/10.1016/j.est.2022.105502

8. Maurya, M., Gawade, S. and Zope, N., 2024. A study of different machine learning algorithms for state of charge estimation in lithium-ion battery pack.

Energy Storage [Online], 6(4), p.e658. Available from: https://doi.org/10.1002/est2.658

9. Moral, C.G., Laborda, D.F., Alonso, L.S., Guerrero, J.M., Fernandez, D., Rivas Pereda, C. and Reigosa, D.D., 2020. Battery Internal Resistance Estimation Using a Battery Balancing System Based on Switched Capacitors. IEEE Transactions on Industry Applications [Online], 56(5), pp.5363–5374. Available from: https://doi.org/10.1109/TIA.2020.3005382

10. Plett, G.L., 2015. Battery management systems: battery modeling. Volume 1. Boston : London: Artech House.

11. Ren, Z., Du, C., Wu, Z., Shao, J. and Deng, W., 2021. A comparative study of the influence of different open circuit voltage tests on model-based state of charge estimation for lithium-ion batteries. International Journal of Energy Research [Online], 45(9), pp.13692–13711. Available from: https://doi.org/10.1002/er.6700.

12. Rivera-Barrera, J., Muñoz-Galeano, N. and Sarmiento-Maldonado, H., 2017. SoC Estimation for Lithium-ion Batteries: Review and Future Challenges. Electronics [Online], 6(4), p.102. Available from: https://doi.org/10.3390/electronics6040102.

13. Sarda, J., Patel, H., Popat, Y., Hui, K. and Sain, M., 2023. Review of Management System and State-of-Charge Estimation Methods for Electric Vehicles. World Electric Vehicle Journal [Online], 14(12), p.325. Available from: https://doi.org/10.3390/wevj14120325.

14. Saxena, A., n.d. Practical implementation of an advanced State of Charge Algorithm on a LiFePO4 and NMC battery pack. https://github.com/AseemLab

15. Wang, Y., Meng, D., Chang, Y., Zhou, Y., Li, R. and Zhang, X., 2021. Research on online parameter identification and SOC estimation methods of lithium-ion battery model based on a robustness analysis. International Journal of Energy Research [Online], 45(15), pp.21234–21253. Available from: https://doi.org/10.1002/er.7175.

16. Xu, D. and Cen, H., 2021. A hybrid energy storage strategy based on multivariable fuzzy coordinated control of photovoltaic grid-connected power fluctuations. IET Renewable Power Generation [Online], 15(8), pp.1826–1835. Available from: https://doi.org/10.1049/rpg2.12152.

17. Xu, R., Sun, H., De Vasconcelos, L.S. and Zhao, K., 2017. Mechanical and Structural Degradation of LiNi x Mn y Co z O 2 Cathode in Li-Ion Batteries: An Experimental Study. Journal of The Electrochemical Society [Online], 164(13), pp.A3333–A3341. Available from:

https://doi.org/10.1149/2.1751713jes.

18. Yu, Q.-Q., Xiong, R., Wang, L.-Y. and Lin, C., 2018. A Comparative Study on Open Circuit Voltage Models for Lithium-ion Batteries. Chinese Journal of Mechanical Engineering [Online], 31(1), p.65. Available from: https://doi.org/10.1186/s10033-018-0268-8.

19. Zhang, L., Li, K., Du, D., Guo, Y., Fei, M. and Yang, Z., 2020. A Sparse Learning Machine for Real-Time SOC Estimation of Li-ion Batteries. IEEE Access [Online], 8, pp.156165–156176. Available from: https://doi.org/10.1109/ACCESS.2020.3017774.

20. Zhang, Y., Dong, Z., Liu, S., Yang, Q. and Jia, Y., 2023. Prediction of the Potential Trade Relationship of Lithium-Ion Battery's Main Element Raw Material Minerals Combined with the Local Characteristics of the Trade Network. International Journal of Energy Research [Online], 2023, pp.1–37. Available from: https://doi.org/10.1155/2023/2280027.

21. Zhang, Z., Chen, Siliang, Lu, L., Han, X., Li, Y., Chen, Siqi, Wang, H., Lian, Y. and Ouyang, M., 2023. High-Precision and Robust SOC Estimation of LiFePO4 Blade Batteries Based on the BPNN-EKF Algorithm. Batteries [Online], 9(6), p.333. Available from: https://doi.org/10.3390/batteries9060333
.

22. Zhao, Y., Yun, H., Liu, S., Jiao, H. and Wang, C., 2014. State-of-charge Estimation for Lithium-ion Batteries Using a Multi-state Closed-loop Observer. Journal of Power Electronics [Online], 14(5), pp.1038–1046. Available from: https://doi.org/10.6113/JPE.2014.14.5.1038.

Bibliography

All the code and the cited Literature Review is available at this GitHub profile: https://github.com/AseemLab