Politecnico di Torino

Master Degree in AUTOMOTIVE ENGINEERING

Master Degree Thesis

Comparison of state of charge estimation methods based on Artificial Intelligence algorithms for lithium-ion batteries used in automotive applications

Supervisors

Prof. Angelo Bonfitto

Ing. Sara Luciani

Candidate

Zhang Mingyuan

October 2022

Abstract

In recent years, laws and regulations on vehicle emissions have become increasingly stringent in various countries, leading to the gradual replacement of internal combustion engine vehicles by electric vehicles (EVs) and hybrid electric vehicles (HEVs) as the mainstream.

To ensure optimal management and safe operation of the battery, a battery management system (BMS) is introduced to estimate the state of the battery through two basic parameters: State of Charge (SOC) and State of Health (SOH). Due to the complex chemical reaction inside the battery and the aging phenomenon caused by frequent use, it is difficult to clearly observe the status of the battery, so estimating SOC and SOH in real time becomes a challenge.

This thesis presents a novel methodology based on Artificial Intelligence. Feed forward neural network (FNN), Long short-term memory (LSTM) and Nonlinear autoregressive neural network with external input (NARX) are taken into SOC estimation performance comparison, then the best neural network is chosen and applied to a specific Li-ion battery module. Through the aging characteristics (specific SOH decreasing range) of this module, three different network classification options are constructed (1-Class,3-Class and 5-Class) and incorporated into SOC subsystems. These SOC subsystems are combined with SOH subsystem respectively, for estimating both SOC and SOH. Finally, comparison between these three combined options is implemented.

The developed method is able to achieve on-board and real-time functionality. The results and experiments prove that this methodology is not only feasible, but also has excellent performance of fast and accurate estimation with compared to other state-of-the-art methodologies.

Content

1 Introduction

Recently, the automotive industry has been giving more and more attention to sustainability with the aim of mitigating the negative impact of vehicle mobility on the environment. OEMS have focused their efforts on developing advanced powertrain architectures in response to ever more stringent CO2 emissions regulations. Based on solutions using an battery electric vehicle(BEV) or an internal combustion engine (ICE) combined with electric motor. It is now established as a reliable alternative to conventional powertrains.

A large number of BEV drivers commonly suffer from range anxiety. The main parameters to be evaluated for proper battery monitoring are the available energy remaining in the battery pack, known as the state of charge (SOC), and the degradation of the battery (SOH). These two states cannot be measured directly because the technology to create a sensor that acts as a fuel gauge does not exist. Therefore, we need to employ some corresponding estimation techniques.

SOC: State of charge (SOC) is an important indicator for assessing the remaining capacity of the battery. An accurate SOC estimation is crucial for ensuring the safe operation of lithium batteries and preventing from over-charging or over-discharging in battery pack of EV. As well as implement corresponding control strategy, in order to obtain lower energy consumption and better performance of the vehicle.

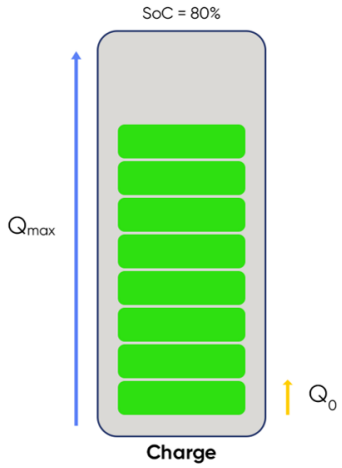


Fig.1.1 Dynamic SOC display

$$SOC_t = SOC_{t-1} \pm \int \frac{I_{batt}}{C_{batt}} dt$$

SOH: While State of health (SOH) is another important indicator that characterizes the actual available capacity of the battery and its degradation state, which also has an important reference value in evaluating the health level of the retired battery and the estimation of the driving range. With the aging phenomenon of the battery caused by the daily use of the battery, the nonlinear characteristic of the battery is also significantly affected. Each small segment of SOH corresponds to different characteristics and will affect the estimation of SOC(shown in fig.3). If the estimates of these two parameters are not combined, each of them will cause a particularly large deviation.

Fig.1.2 Dynamic SOH display

$$SOH = \frac{Q_{max}}{Q_{nominal}}$$

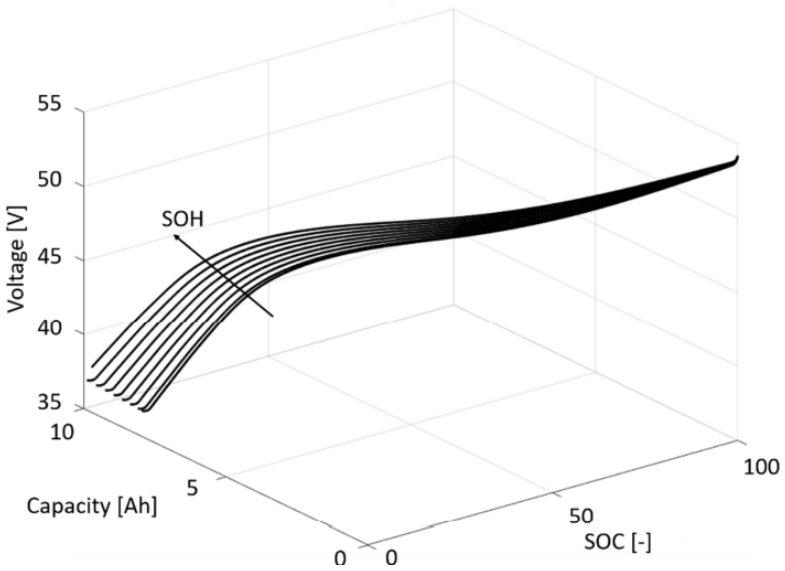


Fig.1.3 Non-linear relationship between the battery parameters and SOC/SOH [1]

The content is divided into 3 steps:

1. SOC estimation Networks comparison, which includes Feed forward neural network (FNN), Long short term memory (LSTM) and Nonlinear autoregressive neural network with external input (NARX). In this step, each networks' RMSE error, ground truth error, and single cycle training time are taken into account. Some additional robustness analysis are applied such as addition noise, and wrong start ground truth value.
2. The best performance network is taken from the previous step and then training with another specific battery module(LIM battery test plan dataset). This dataset considered battery SOH variation, and it can be used for training both SOC network and SOH network. Due to high non-linear behavior and aging phenomenon of Lithium battery module, simultaneous estimation of two parameters becomes challenging. In order to get lower error due to the internal chemical changes caused by aging. Different SOC estimation classes dedicated to corresponding SOH are introduced:1-class SOC network with additional SOH input, 3-classes SOC networks triggered by corresponding SOH value, 5-classes SOC networks triggered by corresponding SOH value. SOH value goes from 100% (new battery) to 80% (scrap and recycling thresholds). In the end, high performance SOC networks for different classes are taken to the next combination step.
3. In the process of combination estimation, the system is separated into two parts: The SOC part received inputs (current, voltage and temperature) measured from sensors and output the SOC value precisely. The SOH part received the estimated SOC and current, voltage as inputs, then output the SOH value. This SOH will be fed back to SOC part , in order to get a real-time loop system that continuously output these two values.

Through continuous practice and with compared to other state-of-the-art methods, The result shows that we have achieved relatively low error(SOC estimating overall RMSE error~1%-3%, SOH estimating RMSE error~1%-2%), both of these errors are lower than results of other studies.

2 SOC estimation State-of-the-art analysis

Although electric vehicles have been in mass production for many years, the technology for battery SOC estimation is actually still far from mature.

Although there are many estimation methods, all of them have certain shortcomings. Kalman filter or neural network are mostly in the paperwork stage. In practical applications, the correction is usually done by adding some influence factors to Ampere-hour method. The battery SOC estimation methods are introduced in the following sections.

2.1 Ampere-Hour Integration

The Ampere-Hour Integration (AHI) method is the simplest SOC estimation method. According to the definition of quantity of electricity , the electricity consumed by the battery in a period of time can be obtained by directly integrating the current, as shown in equation below:

$$SOC_t = SOC_{t-1} \pm \int \frac{I_{batt}}{C_{batt}} dt$$

SOC_t is the value of state of charge at time t, I_{batt} is the current flows in the circuit during charging/discharging phase. C_{batt} is the nominal capacity of the lithium battery, or the so called the maximum capacity of the battery we use. t is the current time discrete, while $t-1$ is the previous time discrete.

The disadvantage of the AHI method is that it is an open-loop estimation system, and this type of system often does not have the ability to correct the error of the initial value and the ability to adjust the error caused by noise and measurement deviation. Since the measurement devices and sensors have small measurement errors (tenths of a percent), however, these errors accumulate over time (integration process), and will eventually cause a relatively large error. An improved method is to use the adaptive Kalman filtering method for initial value calibration [2], and use the set threshold value to switch the estimation between the AHI method and the filtering method [3]. This method can overcome the estimation error caused by the initial value and speed up the calculation, but due to the long-term calculation of the

AHI method, it is still difficult to overcome errors caused by noise and measurement bias.

2.2 Open circuit voltage

Open-circuit voltage (abbreviated as OCV or VOC) is the difference of electrical potential between two terminals of a device when disconnected from any circuit. There is no external load connected. No external electric current flows between the terminals [4].

At a certain temperature, there is a one-to-one numerical relationship between the SOC and OCV of the battery, so the SOC can be estimated by obtaining the OCV-SOC curve (shown in Fig.2.2.1)

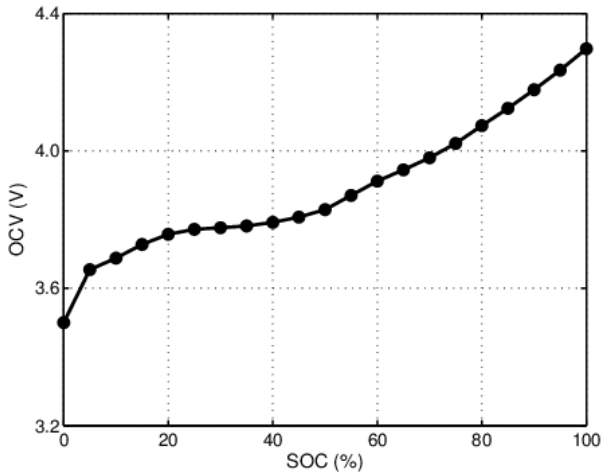


Fig.2.2.1 OCV-SOC curve at room temperature [5]

Since the acquisition of OCV data often requires a long period of rest, it is difficult to carry out the OCV method in real time estimation of SOC. The particularity of the OCV-SOC curve leads to the limitation of using OCV for SOC estimation. If the OCV-SOC curve of the battery has an obvious flat range (just take the example of Fig.2.2.1,and marked in Fig.2.2.2). In this range, even if the SOC changes in a large range, the numerical change reflected in the OCV is small, which leads to a large SOC estimation error even a small OCV error.

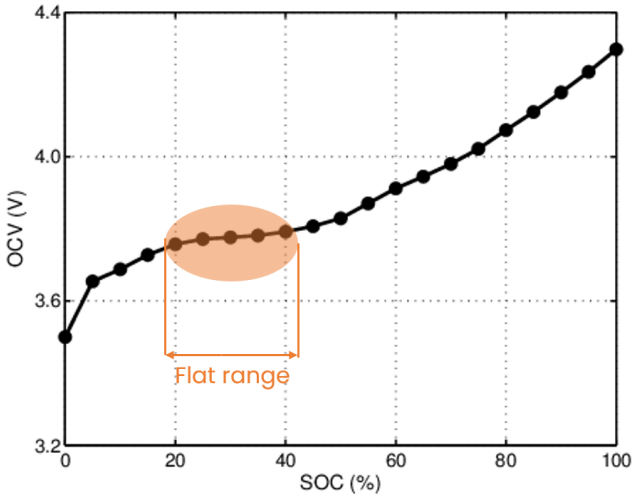


Fig.2.2.2 Non-precise range in OCV-SOC diagram

2.3 Equivalent circuit model

Before applying some additional algorithms (except artificial intelligence), we first need to build up a model to describe the electrochemical reaction and its characteristics of the battery. The models of lithium-ion batteries can be divided into electrochemical models and equivalent circuit models. Here we mainly establish the equivalent circuit model and give an example.

Battery modeling is extremely challenging due to the high level of non-linearity of electrochemical processes. Battery model depends on : SOC, Temperature, Ageing, Current direction, Charging/Discharging rate.

There are four typical equivalent models for lithium batteries: Rint model, PNGV model, GNL model and Thevenin model [6]. Where the most employed models used for system-level simulations of HEVs/EVs are Thevenin-based circuital models. Since Thevenin model has the most convenient voltage calculation than the other three models and the error is lower.

So, in this part of state-of-the-art analysis, Thevenin model is chosen as the equivalent circuit model for lithium batteries.

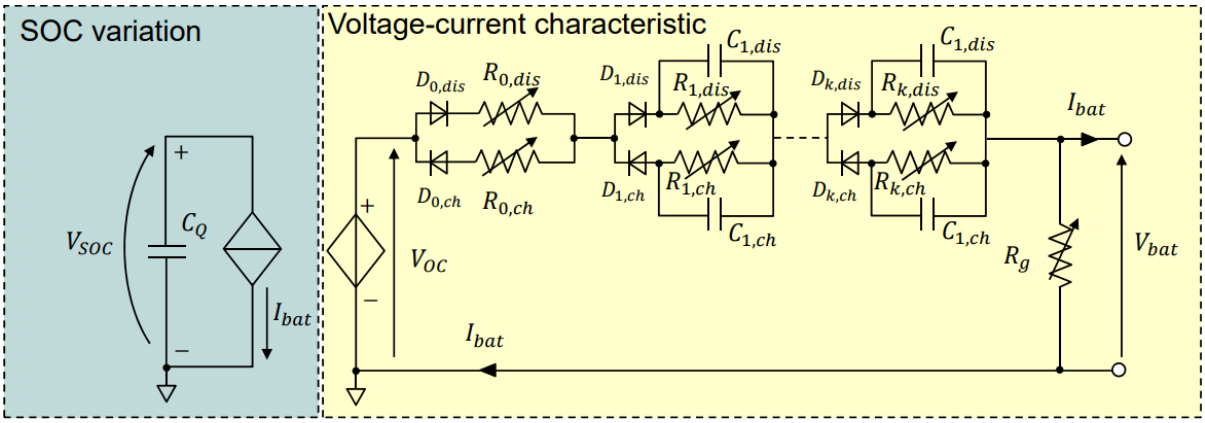


Fig.2.3.1 Most detailed battery Thevenin circuital model [7]

In this model, main parameters are:

$V_{oc}= OCV$ = Open-Circuit Voltage = voltage source depending on the SOC and temperature

$R_{0.ch}/R_{0.dis}$= total internal series resistance for charging/discharging that are dependent on the SOC and temperature

R_{sd}=resistance that represents the battery self-discharging

$R_{k.ch}/R_{k.dis}$, $C_{k.ch}/C_{k.dis}$,$(k = 1, \dots n)$ = RC groups modeling the voltage transients due to the chemical reactions, all resistances are depending on the SOC and temperature

$D_{k.ch}/D_{k.dis}(k = 0, \dots n)$ = ideal bidirectional switches (diodes) conducting the charging/discharging currents

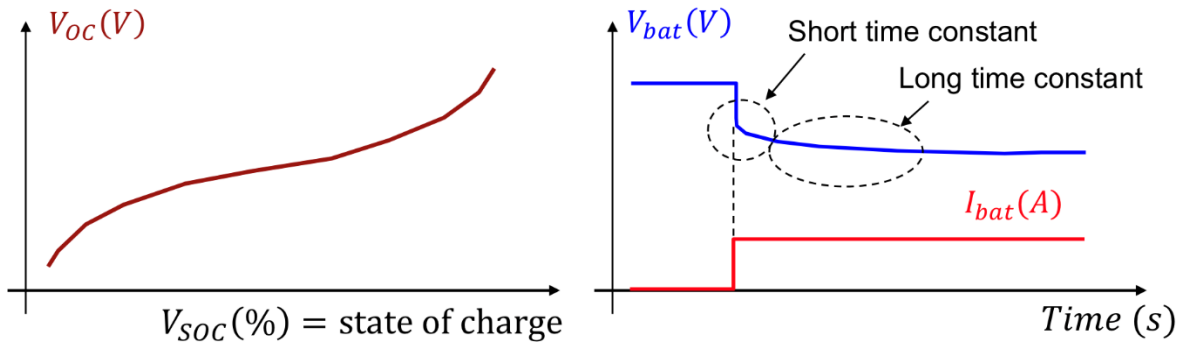


Fig.2.3.2 Battery voltage characteristics

In Fig.5 we can see: Typical variations of the $V_{OC}(SOC)|_{T=CT}$ and battery output voltage V_{bat} for a step variation of the battery current I_{bat}. The battery voltage variation for a step variation of battery current exhibits a waveform with two different time constants. In this way, we would get a simplified second-order model with two RC networks having two different time constants:

$$\tau_1 = R_1 \cdot C_1 \qquad \tau_2 = R_2 \cdot C_2$$

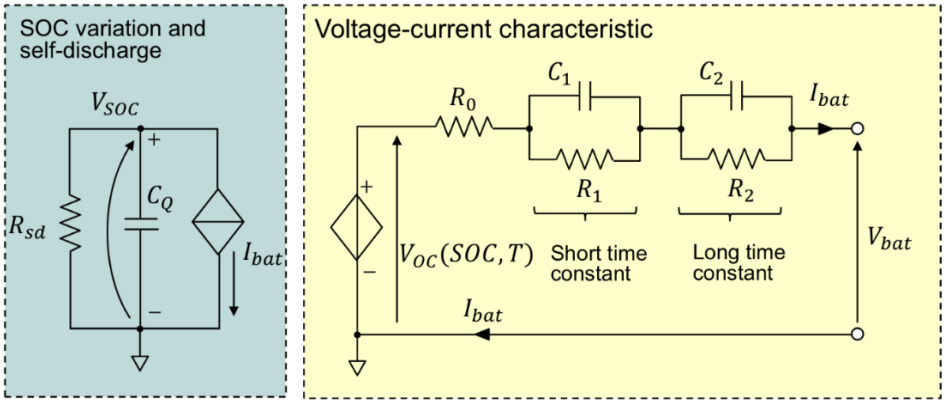


Fig.2.3.3 Simplified battery Thevenin circuital model

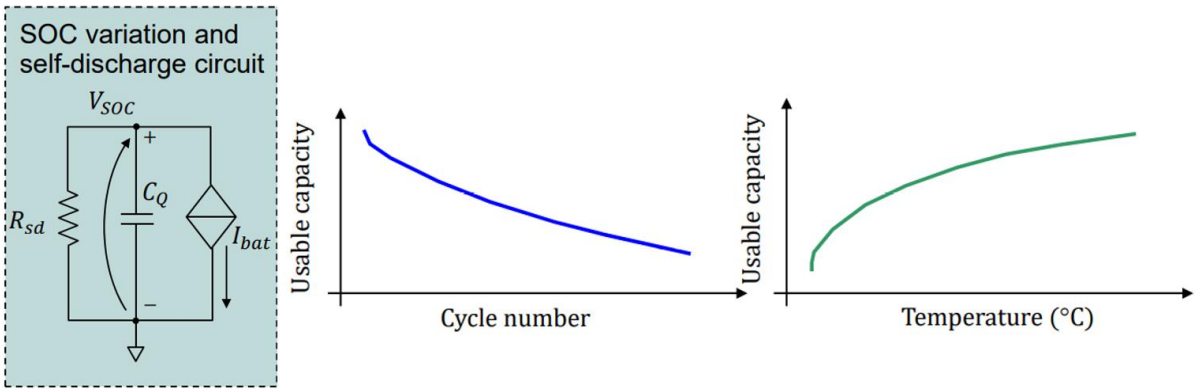


Fig.2.3.4 SOC variation and self-discharge circuit

$$C_q(F) = 3600 \cdot Q_{rated} \cdot K_{Temp} \cdot K_{Cycle}$$

Where:

R_{sd}= self-discharging resistance (if unknown, set to tens of $k\Omega$ or eliminate)

Q_{rated}= rated capacitance

K_{Temp}= corrective gain depending on the temperature (1 if unknown)

K_{Cycle} = corrective gain depending on the number of battery cycles (1 if unknown)

Then we can have the relationship between I_{bat} and V_{soc}(In Simulink form):

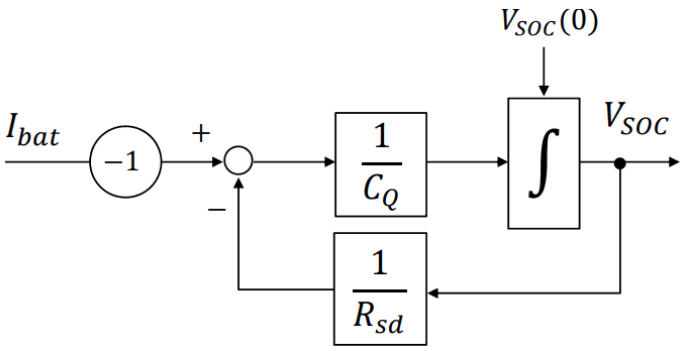


Fig.2.3.5 Simulink form of SOC variation and self-discharge circuit

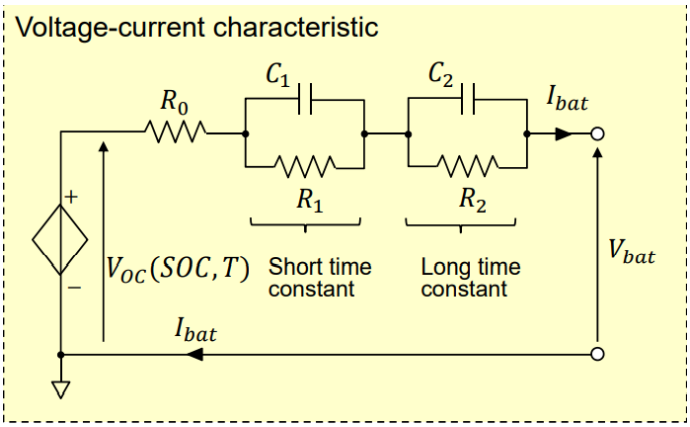


Fig.2.3.6 Second order Thevenin battery electrical model

As Fig.8 shows above, Battery electrical model includes a voltage source, a resistance and two RC networks, Open circuit voltage V_{oc} depend on the SOC and on the temperature(V_{oc} is a rated parameter of a battery pack and should be provided by the battery manufacturer), Typically, the V_{oc} is provided as Look-Up-Tables (LUTs) or as multiple curves (corresponding to different temperatures) using polynomial interpolation with respect to SOC. Then we can have the relationship between V_{oc} and V_{Bat}(In Simulink form):

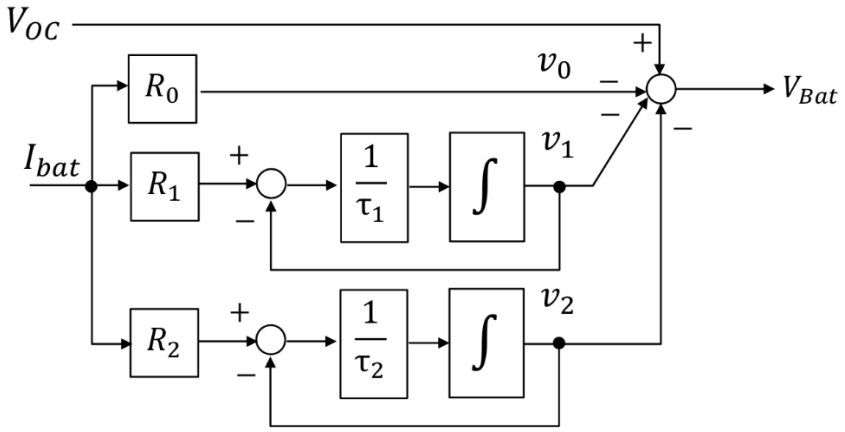


Fig.2.3.7 Simulink form of Battery electrical model

$$V_{bat} = V_{oc}[SOC(t)] - V_0 - V_1 - V_2$$

$$\tau_1 = R_1 \cdot C_1 \quad \tau_2 = R_2 \cdot C_2$$

After combining Fig.7 and Fig.9, we can have a complete simplified Thevenin battery electrical model, where I_{bat} is the input, V_{bat} is the output, the complete block in the middle acts as a precise transfer function:

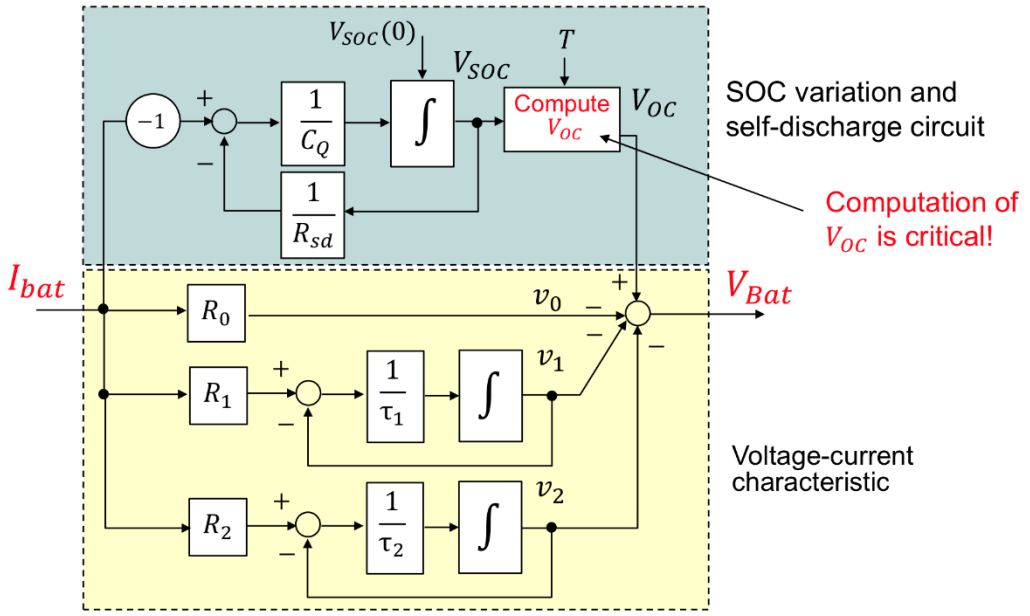


Fig.2.3.8 Final simplified Thevenin battery electrical model

The parameters of the cells/modules or battery packs can be identified with experimental procedures. The most employed one is the discharging with current pulses having known amplitudes I_{Pluse}.

The parameters are identified by measuring the battery voltage, the first step is due to the internal resistance R_0. And The voltage after a sufficient relaxation time (>5 $\cdot\tau_2$) is the V_{OC}. As the Fig.11 shows below:

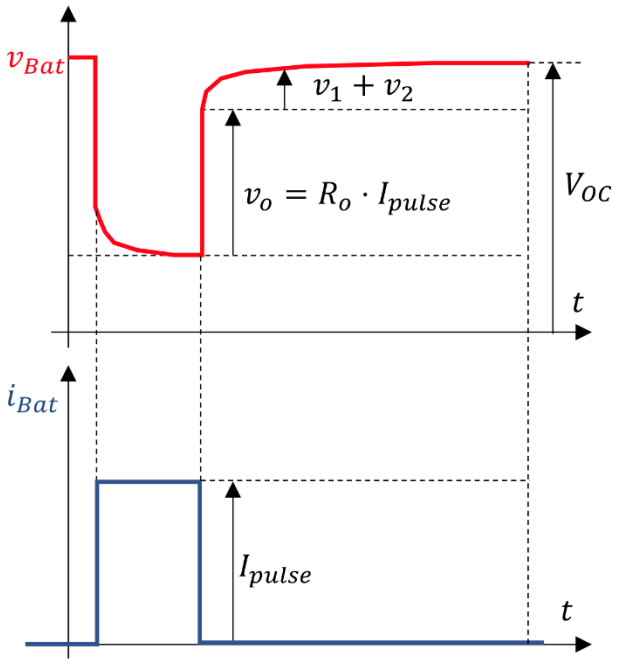


Fig.2.3.9 Battery voltage characteristics

After knowing the equivalent circuit model, we could compute SOC based on the known current I_{bat}:

$$SOC(t) = SOC(t-1) - \int \frac{I_{bat}(t)}{C_q} dt \qquad (1)$$

After knowing the value of SOC, from the equation in Fig.9, we could evaluate the output V_{bat} at any given time:

$$V_{bat} = V_{oc}[SOC(t)] - V_1(t) - V_2(t) - I_{bat}(t) \cdot R_0 \qquad (2)$$

Discretizing Equations (1) and (2), the state-space equation of the second-order RC equivalent circuit model is as follows:

$$x(k+1) = A \cdot x(k) + B \cdot u(k)$$

$$y(k) = C \cdot x(k) + D \cdot u(k) \qquad (3)$$

Where from the equation (3):

$$x(k) = \begin{bmatrix} SOC(k) \\ V1(k) \\ V2(k) \end{bmatrix} \qquad u(k) = I_{bat}(k) \qquad y(k) = V_{bat}(k)$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{\frac{-\Delta t}{\tau_1(k)}} & 0 \\ 0 & 0 & e^{\frac{-\Delta t}{\tau_2(k)}} \end{bmatrix} \qquad B = \begin{bmatrix} \frac{-\Delta t}{C_q} \\ R_1(k) \cdot (1 - e^{\frac{-\Delta t}{\tau_1(k)}}) \\ R_2(k) \cdot (1 - e^{\frac{-\Delta t}{\tau_2(k)}}) \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{\partial V_{oc} SOC(k)}{\partial SOC(k)} & -1 & -1 \end{bmatrix} \qquad D = -R_0$$

Δt is the interval of sampling time, τ represents the time constant, $\tau_1 = R_1 \cdot C_1 \quad \tau_2 = R_2 \cdot C_2$

2.4 Kalman filter

The Kalman filter is an efficient autoregressive filter, which can estimate the state of a dynamic system in the combined information of many uncertain situations. It is a powerful and versatile tool. Its author, Rudolf E. Kalman, during a visit to NASA's Ames Research Center, found that this method could help solve the Apollo program's orbit prediction problem, and later NASA did in the Apollo spacecraft's navigation system. This filter is also

used. In the end, the spacecraft sailed to the moon correctly, completing the first moon landing in human history.

|For this filter, we can almost conclude that Kalman filter can make well-founded prediction about what the system is going to behave in the next step, as long as there are dynamic systems with uncertain information. Even in the presence of noisy disturbance, Kalman filter is usually very good at figuring out what is going on and finding imperceptible correlations between phenomena.

This makes Kalman filter ideal for systems that are constantly changing. It also has the advantages of a small memory usage (only the previous state needs to be retained) and speed, making it ideal for real-time problems and embedded systems.

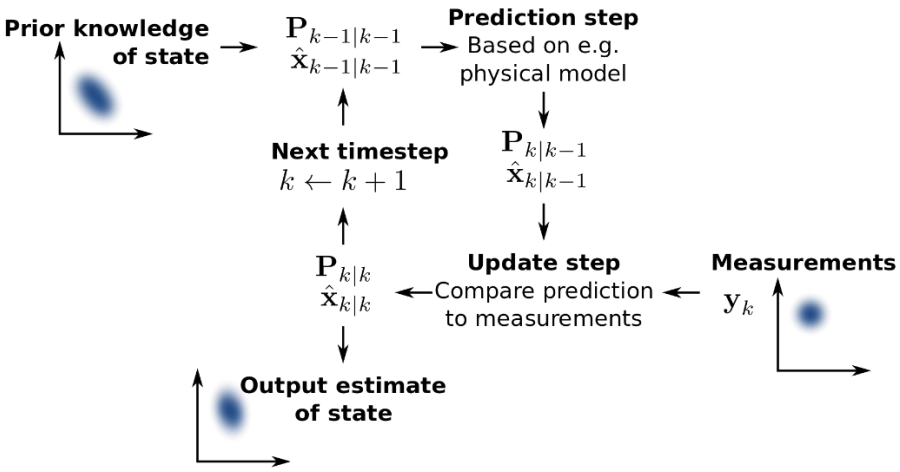


Fig.2.4.1 Schematic of Kalman filter [8]

Step-by-step guide

Step1 Building the model:

First, we should make sure that the Kalman filter conditions fit our problem.

Let's remember the two typical equations of the Kalman filter:

$$x_k = Ax_{k-1} + Bu_k + W_{k-1} \quad (1)$$

$$z_k = Hx_k + v_k \quad (2)$$

(1) implies that each x_k can be represented by a linear stochastic equation. Any x_k is a linear combination of the value of its previous state plus the control signal u_k and the processing noise W_{k-1}. In most cases, it is not necessary to control the signal u_k .

(2) tells us that any measured value (here we are not sure if it is accurate) is a linear

combination of the current state signal value and the measurement noise, which we default to a Gaussian distribution. The processing noise and measurement noise in these two formulas are considered to be statistically independent.

The coefficients A, B and H in the equations are matrices of general form. But in most signal processing problems, we use models where these coefficients are just numerical values. And when these values change from state to state, most of the time we can assume they are constants. If we are sure that our system fits this model (as most systems do), then only the mean and standard deviation of the estimated noise function W_{k-1} and v_k are left. We know that in real life, no signal is purely Gaussian distribution, but we can make assumptions by approximating.

Here we always have to keep in mind: “The more accurate the noise parameters, the better estimates we can get".

Step2 Start processing:

If we successfully fit our model to the Kalman filter, the next step is to determine the necessary parameters and initial values.

We have two different sets of equations: time update (prediction) and measurement update (correction). Both sets of equations are applied to the *K* state:

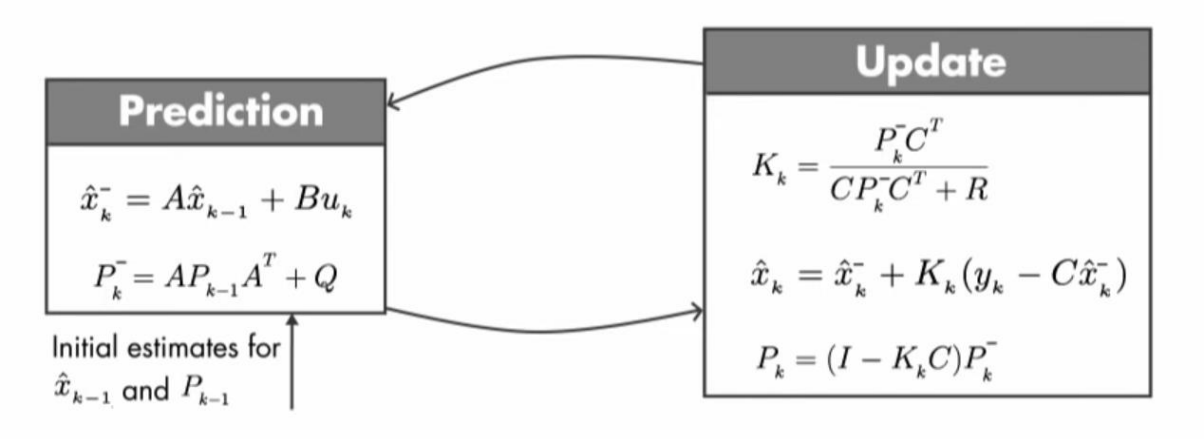


Fig.2.4.2 Time update and measurement update [9]

Step3 Iteration:

After we have gathered all the information we need, we start the process and now we can iterate the estimation. We must keep in mind that the estimate of the previous state will be the input to the estimate of the current state.

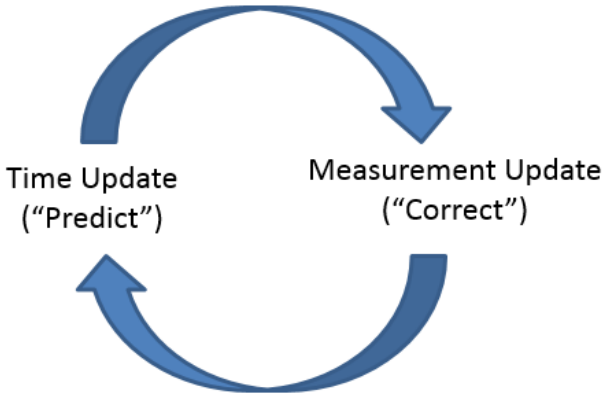


Fig.2.4.3 The Kalman Filter operates in a "predict – correct" loop [10]

Here, a priori estimate refers to a rough estimate before measurement update correction. Here we use the prior in the measurement update formula.

In the measurement update formula, we find an estimate of x at state k, and found the value needed for state $k+1$ estimation.

The Kalman gain K_k we compute is not needed in the next iterative step. The K_k value we evaluate during the measurement update phase is also called the posterior value.

2.5 Artificial Intelligence (Artificial Neural Network (ANN))

Artificial Neural Network (ANN), abbreviated as Neural Network (NN), is a mathematical model based on the basic principles of neural networks in biology, and after understanding and abstracting the structure of human brain and the response mechanism of external stimuli, it simulates the processing mechanism of complex information by the nervous system of human brain based on theoretical knowledge of network topology. Characterized by parallel distributed processing capability, high fault tolerance, intelligence and self-learning, this model combines the processing and storage of information and has attracted attention in various disciplines with its unique knowledge representation and intelligent adaptive learning capability. It is actually a complex network with a large number of simple components interconnected, with a high degree of nonlinearity, capable of complex logical operations and systems realized by nonlinear relationships [11].

A neural network is an operational model that consists of a large number of nodes (or neurons) interconnected with each other. Each node represents a specific output function, called the

activation function. Each connection between two nodes represents a weighted value for the signal passing through the connection, called *weight*, by which the neural network simulates human memory. The output of the network depends on the structure of the network, its connections, weights, and activation functions. The network itself is usually an approximation of some algorithm or function in nature, or it may be an expression of a logical strategy. The idea of constructing neural networks is inspired by the operation of biological neural networks (As Fig.2.2.4 shows below). Artificial neural networks, on the other hand, combine the understanding of biological neural networks with mathematical statistical models and are implemented with the help of mathematical statistical tools. On the other hand, in the field of artificial perception in artificial intelligence, we enable neural networks to have human-like decision making abilities and simple judgments by means of mathematical statistics, an approach that is a further extension of traditional logical algorithms.

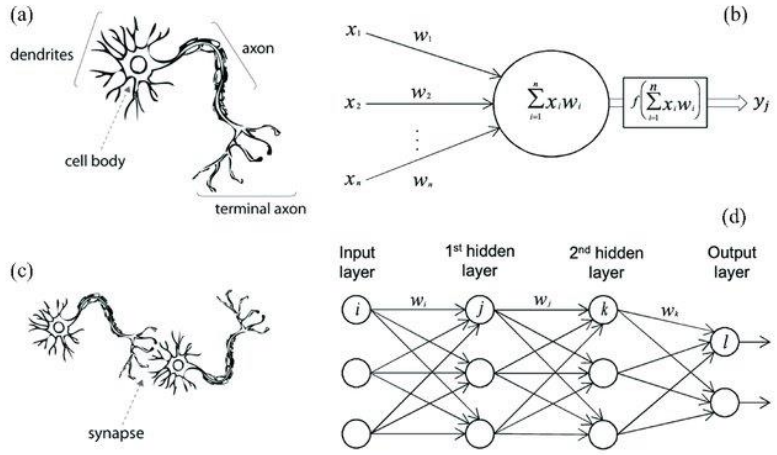


Fig.2.5.1 A biological neuron in comparison to an artificial neural network [12]

The lithium-ion battery is a highly complex nonlinear time-varying electrochemical system, which makes it difficult to find a simple model to describe its behavior in all operating conditions. The artificial Neural Network (ANN) approach allows to find the networks that are not influenced by any physical or chemical principle by simply finding relationships that can relate a given input to a given output. In this way, it is possible to establish direct relationships that correlate input measurements (current, voltage and temperature) with the target signals of interest (SOC/SOH).

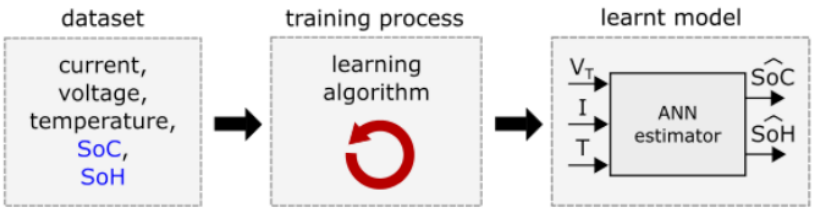


Fig.2.5.2 schema of ANN SOC estimator [13]

For the future, with the continuous accumulation of battery dataset, the further improvement of current and voltage measurement accuracy, Artificial intelligence-based estimation methods have very high potential for application, the algorithm of SOC will be further combined with the whole vehicle controller in the direction of simple, efficient, accurate and low cost.

3 SOC estimation networks performance comparison

3.1 Networks' concept introduction

In this chapter, three types of networks are chosen. Each concept and architecture are different from the others. First, each networks' basic concept is introduced, and then training, testing and analysis procedures for each of the networks are done.

3.1.1 FNN (Feed Forward Neuron Networks)

FNN, also as known as multi-layer perceptron (MLP). MLPs are one of the artificial intelligence methods in the sub-class of machine learning used in different fields such as system modeling, anomaly detection, classification applications [14]. The MLP consists of a series of structures called neurons that mimic the information processing and information acquisition capabilities of the human brain. MLPs are frequently used especially in modeling processes of systems with complex mathematical models [15].

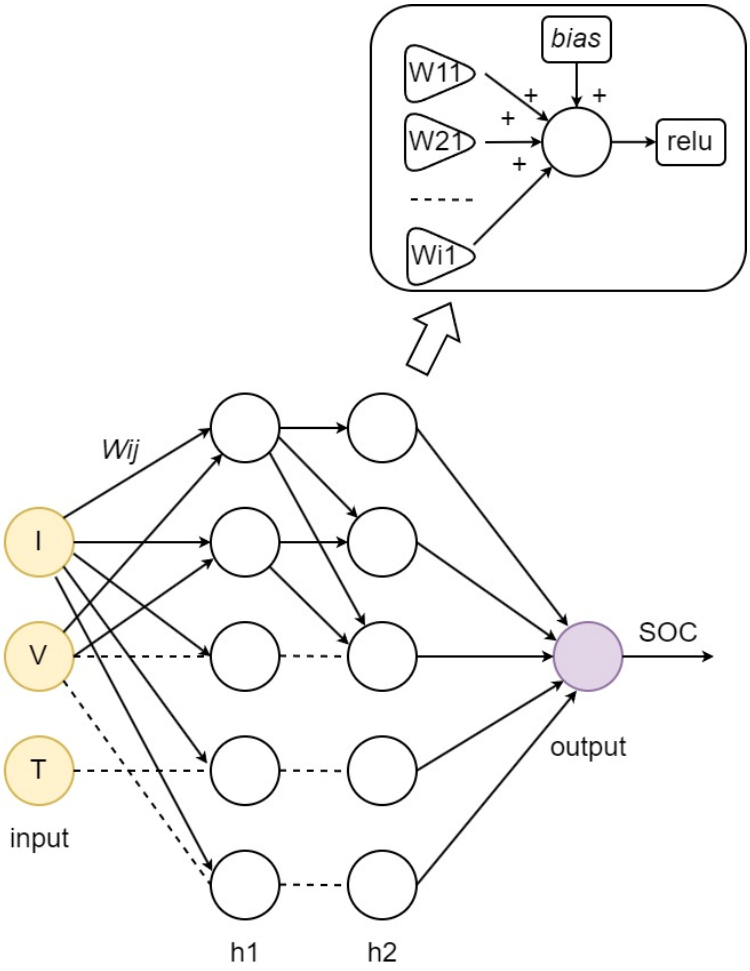


Fig.3.1.1.1 Schematic of FNN

As figure 1 shows above, there are three basic elements of a neural network: weights, biases, and activation functions.

Weight (ω_{ij}): The strength of the connection between neurons is represented by the weight, and the size of the weight represents the size of the possibility.

Bias(b_j): Bias is set to correctly classify samples and is an important parameter in the model, that is, to ensure that the output value calculated from the input cannot be activated casually.

Activation function: plays the role of nonlinear mapping, which can limit the output amplitude of neurons within a certain range, generally between (-1~1) or (0~1). The most commonly used activation function is the log sigmoid function which maps numbers from (-∞, +∞) to the range (0~1). And tan sigmoid function, which maps numbers from (-∞, +∞) to the range (-1~1). The sigmoid functions are given in Eq. (1) and Eq. (2). The neuron outputs in the layers are mathematically given in Eq. (3).

$$logsig(u) = \frac{1}{1+e^{-u}} \quad (1)$$

$$tansig(u) = \frac{2}{1+e^{-2u}} - 1 \quad (2)$$

$$yj = \mathcal{F}(uj) = \mathcal{F}\left(\sum \omega_{ij} x_i + b_j\right) \quad (3)$$

3.1.2 LSTM (Long Short-Term Memory)

Fig.5 shows the typical scheme of recurrent neural network (RNN). For an input sample $x = [x1; x2; \ldots; xn]$ where n denotes the sequence length, RNN calculates the hidden state vector sequence $h = [h1; h2; \ldots; hn]$ and output the sequence $y = [y1; y2; \ldots; yn]$ through iteration of the following equations from $t = 1$ to n.

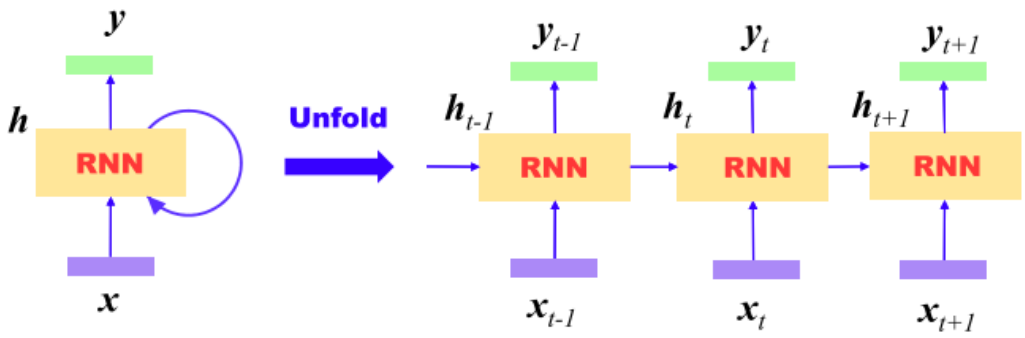


Fig. 3.1.2.1 Illustration of recurrent neural network structure.

$$h_t = H(Wxh\, X_t + W_{hh} h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{xh} h_t + b_y \quad (2)$$

where W is the weight matrices and b denote the bias term. Wxh represents the weight matrix between the input and the hidden states, b_h denotes the hidden bias vectors, and H is the non-linear activation function in the hidden layer [16].

LSTM (Long short-term memory) can be considered as an upgrade of RNN, it solves the problems of gradient disappearance, gradient explosion, and long-term dependence of RNN. The traditional RNN node output is determined only by the weights, bias, and activation function (Fig.5). An RNN is a chain-structure where the same parameters are used for each time discrete. The reason why LSTM can solve the long-term dependency problem of RNN is that LSTM introduces the gate mechanism to control the flow and loss of features.

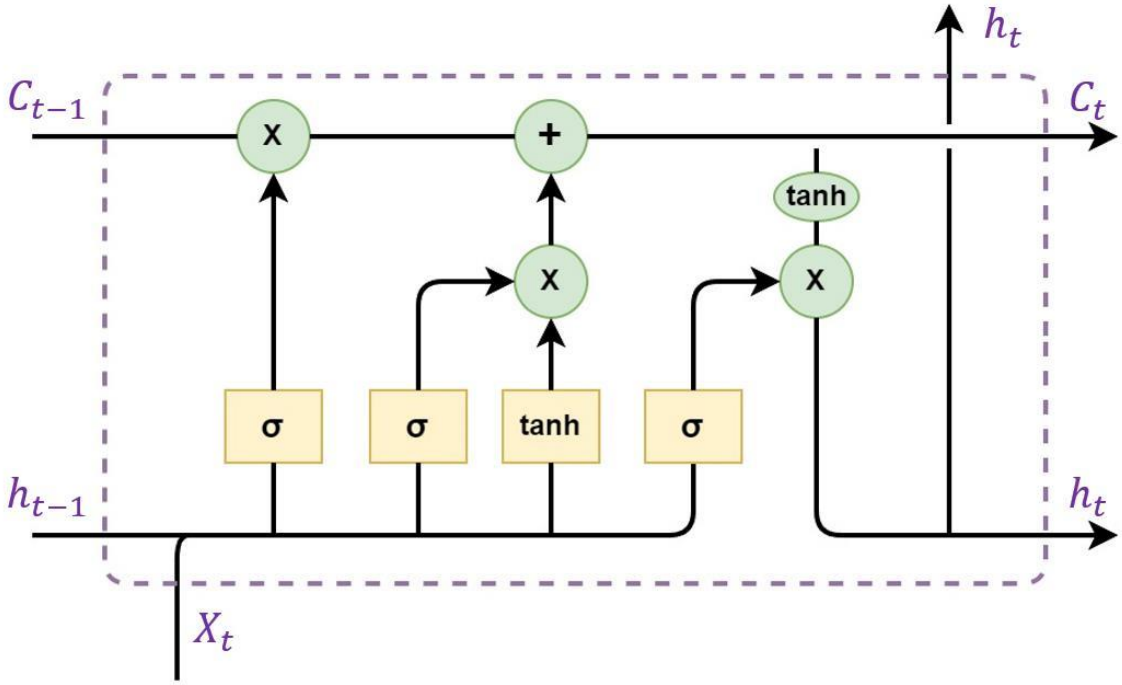


Fig.3.1.2.2 Details of LSTM

The core part of LSTM is the part similar to the conveyor belt at the top in Fig.6. This part is generally called the cell state and it exists in the entire chain system of LSTM from beginning to the end.

Here are explanations of LSTM equations:

$$C_t = f_t \times C_{t-1} + i_t \, \widetilde{C}_t$$

C_t represents long term memory. f_t is called forget gate, it represents which features of C_{t-1} are used to compute C_t. It is a vector with each element in the range [0,1].Where:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

i_t is called the input gate, it is also a vector whose elements are in the interval [0,1], and it is also calculated by $x_t$and h_{t-1}through the activation function σ. i_t is used to control which features of $\widetilde{C}_t$ is used to update C_t , in the same way as for f_t.

Finally, in order to calculate the predicted value $\hat{y}_t$ and generate the complete input for the next time discrete, we need to calculate the output of the hidden node h_t.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

h_t is obtained from the output gate o_t and cell state C_t, where o_t is calculated in the same way as f_t and i_t.

3.1.3 NARX (Nonlinear autoregressive network with exogenous inputs)

The nonlinear autoregressive network with exogenous inputs (NARX) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX model is based on the linear ARX model, which is commonly used in time-series modeling. The defining equation for the NARX model is:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u))$$

where the next value of the dependent output signal $y(t)$ is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. You can implement the NARX model by using a feedforward neural network to approximate the function f .A diagram of the resulting network is shown below, where a two-layer feedforward network is used for the approximation. This implementation also allows for a vector ARX model, where the input and output can be multidimensional [18].

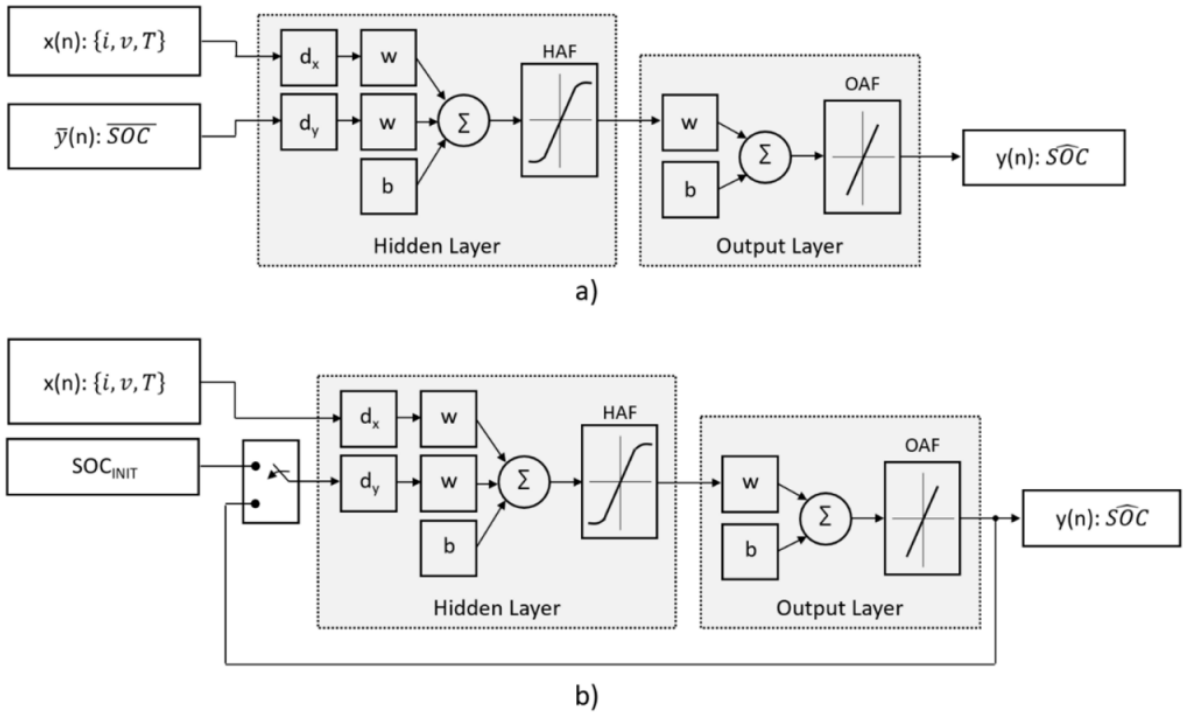


Fig. 3.1.3.1 (a) Series–parallel (SP) mode (open-loop configuration) adopted during the training. (b) Parallel (P) mode (closed-loop configuration) adopted for the estimation when the network is deployed. HAF: hidden activation function. OAF: output activation function. w: weight. b: bias. [1]

In our case, from the a), b) and the equation showed above, our $y(t) \in R$ and $u(t) \in R$ denote the output (state of charge) and inputs (current, voltage, and temperature) of the NARX model at the discrete timestep n, respectively, dx and dy are the input and output memory delays used in the model, respectively, and f is the activation function, usually $tanh$ or $linear$.

During the training phase, mode a) is adopted as supervised training, SOC computed by coulomb counting are used as ground truth, this approach allows for learning and correcting parameters inside the network (w, b), since the correction method mainly adopts gradient descent and backpropagation similar to FNN, an additional Levenberg–Marquardt(LM) algorithm is applied for optimizing the computing process of gradient descent. (The LM algorithm is a nonlinear optimization method between the Newton method and the gradient descent method. It is not sensitive to the problem of over-parameterization, and can effectively deal with the problem of redundant parameters, so that the chance of the cost function falling into a local minimum is greatly reduced.)

While during the testing phase, mode b) is adopted as “closed loop condition”. There will be no ground truth and the predicted SOC value is fed back to the input. Further investigation such as “network wrong initial value” and robustness analysis will be discussed in the next step.

3.2 Dataset introduction and training, testing methodologies for three types of networks

In this chapter it is mainly focusing on the performance (training speed, SOC prediction error and loss function error) of three different machine learning/deep learning neural networks, which are FNN, NARX, LSTM. And finally, the one which has the best performance and its corresponding configurations are chosen in order to get into further updates.

3.2.1 Specific dataset introduction:

This specific dataset is measured by Dr.Vidal et.al [19], thanks to their long exploration, we could use this specific dataset for networks’ performance comparison.

The energy cell we used is a Panasonic NCA18650PF cylindrical battery(Fig.3.2.1.1). The datasets for each battery were acquired using the lab equipment listed in Table 1, including Digatron Power Electronics battery cyclers with +/- 0.1% of full-scale voltage and current accuracy and thermal chambers with +/- 0.5 ºC temperature regulation accuracy. A schematic of the test bench and data logging system is presented in Fig. 3.2.1.2.

Battery	Panasonic
Model	18650PF
Format	Cylindrical
Chemistry	NCA
Rated Capacity	2.9 Ah
Normalized Resistance	104 mΩAh
Specific Power	1.7 kW/kg
Specific Energy	207 Wh/kg
Energy Density	577 Wh/l

Fig.3.2.1.1 Panasonic NCA18650PF

Battery datasets and lab equipment.

Battery Datasets	Panasonic
Training	Mix 1 to 4, and US06
Testing	LA92, NN
Temperatures	-10°C, 0°C, 10°C, 25°C
Cycler Manufacturer	Digatron Firing Circuits
Test Channel	25 A, 0–18 V channel
Data Acquisition	10 Hz
Accuracy	±0.1% full scale
Thermal chamber	Cincinatti Subzero ZP8
Internal Volume	8 cu. Ft.
Accuracy	±0.5 °C

Table.1

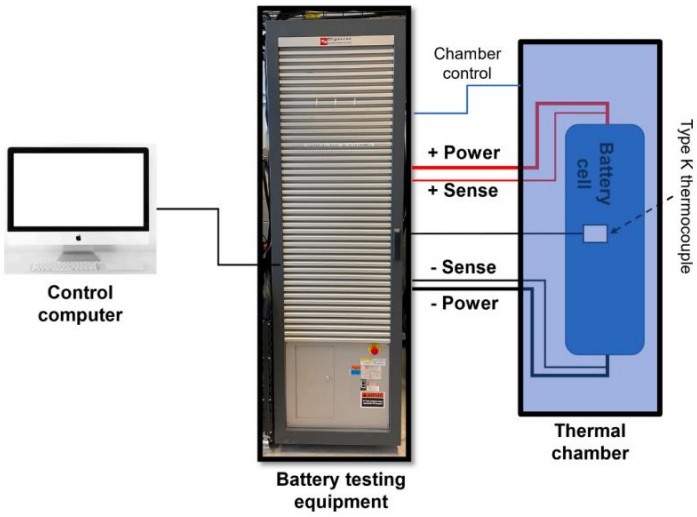


Fig.3.2.1.2 Schematic of the test bench and data logging system.

The following list are main introductions of data characteristics:

- A series of power profiles were calculated for an electric vehicle, including the standard UDDS, US06, LA92, and HWFET (or HW) drive cycles, as well as a series of “Mix” cycles consisting of randomized portions of the standard drive cycles.
- The power profiles were scaled for a single cell in the pack. For the Panasonic cell, the drive cycles were modelled for a Ford F150 electric truck with a 35 kWh pack consisting of 3360 of the Panasonics cells.
- The voltage, current, battery temperature, and ampere-hours were logged at 10 Hz, and the data were down sampled to 1 Hz.

- At -10ºC and -25 ºC, the voltage deviation under load for both cells is relatively high, making SOC estimation challenging.
- for the Panasonic at -10ºC and -25 ºC there is no charging current (positive current) during the drive cycles due to the manufacturer's requirement that charging be performed only above 0ºC.
- About two-thirds of the data is used for training and one-third for testing

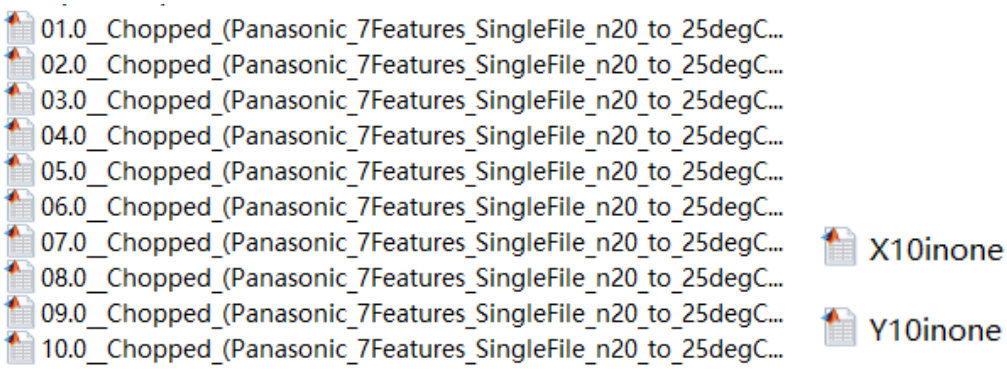


Fig.3.2.2 Separated and combined dataset files

Fig.3.2.2 shows how the datasets are separated, and the training datasets have following characteristics:

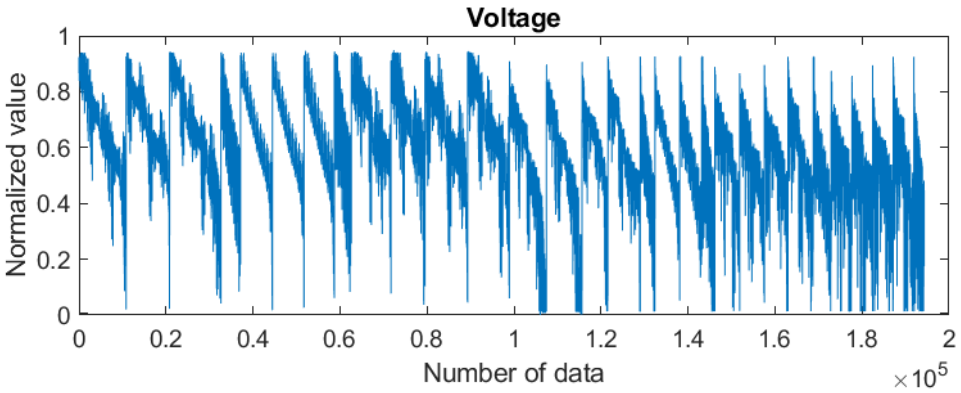
- The dataset used was took from Vidal's paper, the training data are 7X194280, and it is separated into 10 minibatches in order to train FNN and LSTM.
- For the NARX training phase, since the network does not accept minibatches data, so they are combined into one matrix: X10inone (7X194280) Y10inone(7X194280).

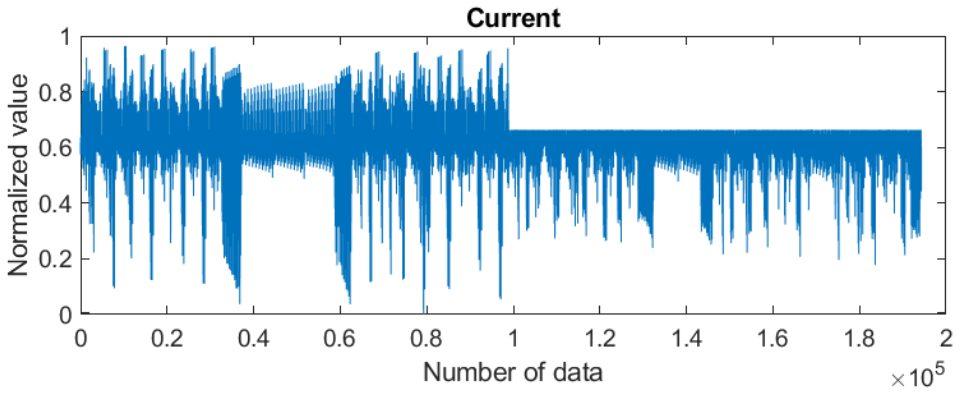
The "X" data has 7 rows, where the data in each row is as follows:

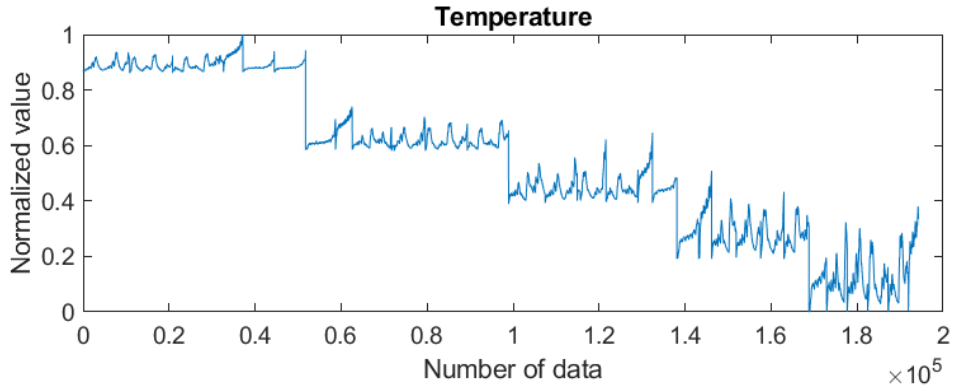
- { V, I, T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz}.
- where V is voltage, I is current, T is temperature, and the _0.5mHz and _5mHz data is filtered with a 1st order low pass Butterworth filter.

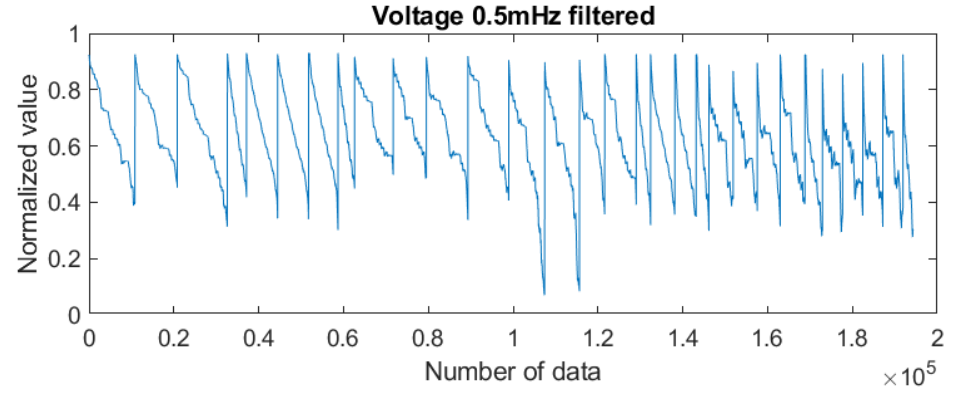
The "Y" data is state of charge calculated via coulomb counting.

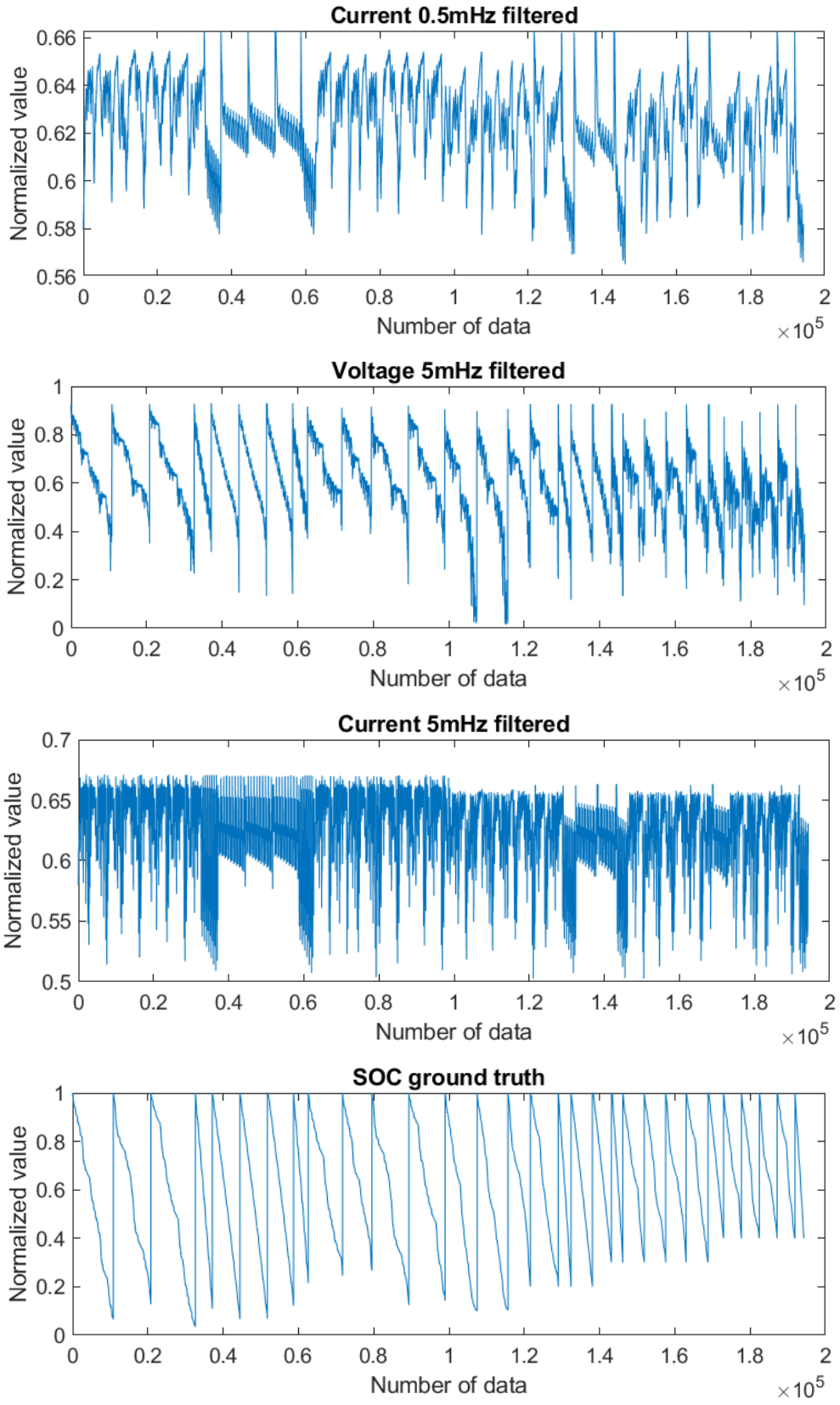
Figures below shows the whole training data, The type of each data is titled above it:


Voltage
Normalized value
Number of data
×10^5


Current
Normalized value
Number of data
×10^5


Temperature
Normalized value
Number of data
×10^5

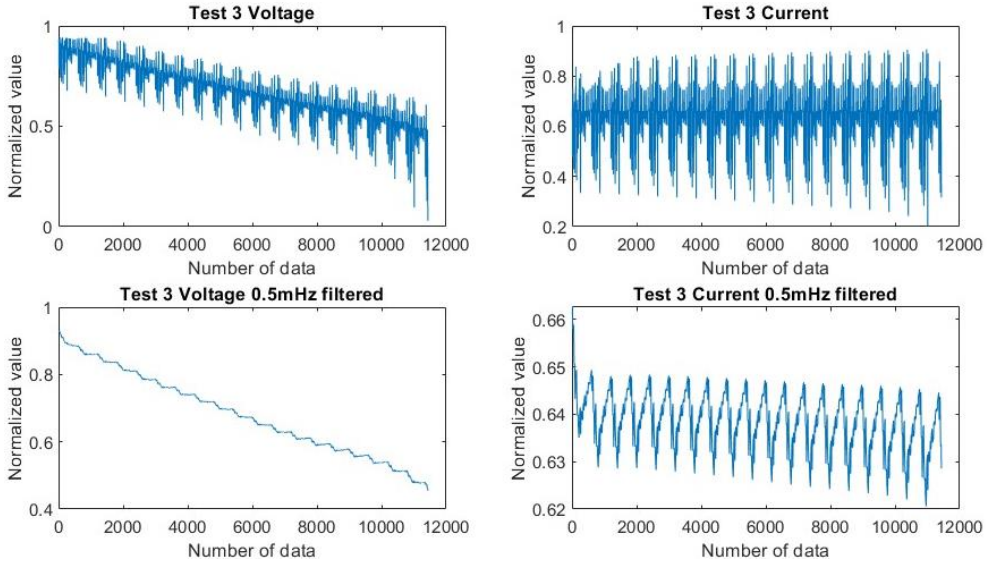

Voltage 0.5mHz filtered
Normalized value
Number of data
×10^5


Current 0.5mHz filtered
Normalized value
0.66
0.64
0.62
0.6
0.58
0.56
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Number of data
×10^5
Voltage 5mHz filtered
Normalized value
1
0.8
0.6
0.4
0.2
0
Number of data
×10^5
Current 5mHz filtered
Normalized value
0.7
0.65
0.6
0.55
0.5
Number of data
×10^5
SOC ground truth
Normalized value
Number of data
×10^5

From the original value and the two filtered values, for example: Voltage, V_0.5mHz filtered and V_5mHz filtered, we can see the higher the filtering frequency, the clearer the outline of data. If we input both of these data into the target network, this procedure may help the network remember more sturdy the overall trend of data. In the following step we will set { V, I, T} and {T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz} as two training options, in order to investigate the performance and characteristics of three types of networks.

After introducing the training dataset, while the testing dataset has the following characteristics:

- Testing data is separated into 15 batch of data with different temperature: -20, -10, 0, 10, and 25degC.
- For FNN and LSTM, we can select the number between 1 and 15 to choose the test data.
- For NARX, the data is selected in the same way.
- The testing data 3(25 degree) and 11(-10 degree) are chosen for testing performance display, in order to see if the model has obtained good temperature generalization ability (two testing data are shown in Fig.3.2.3 and Fig.3.2.4).



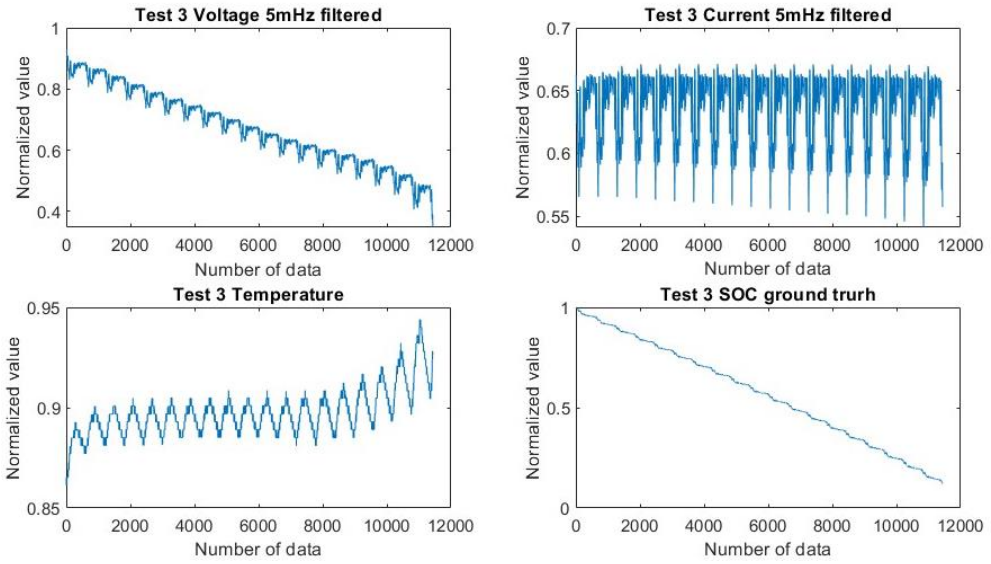


Fig.3.2.3 characteristics of 3rd, 25degree-testing dataset

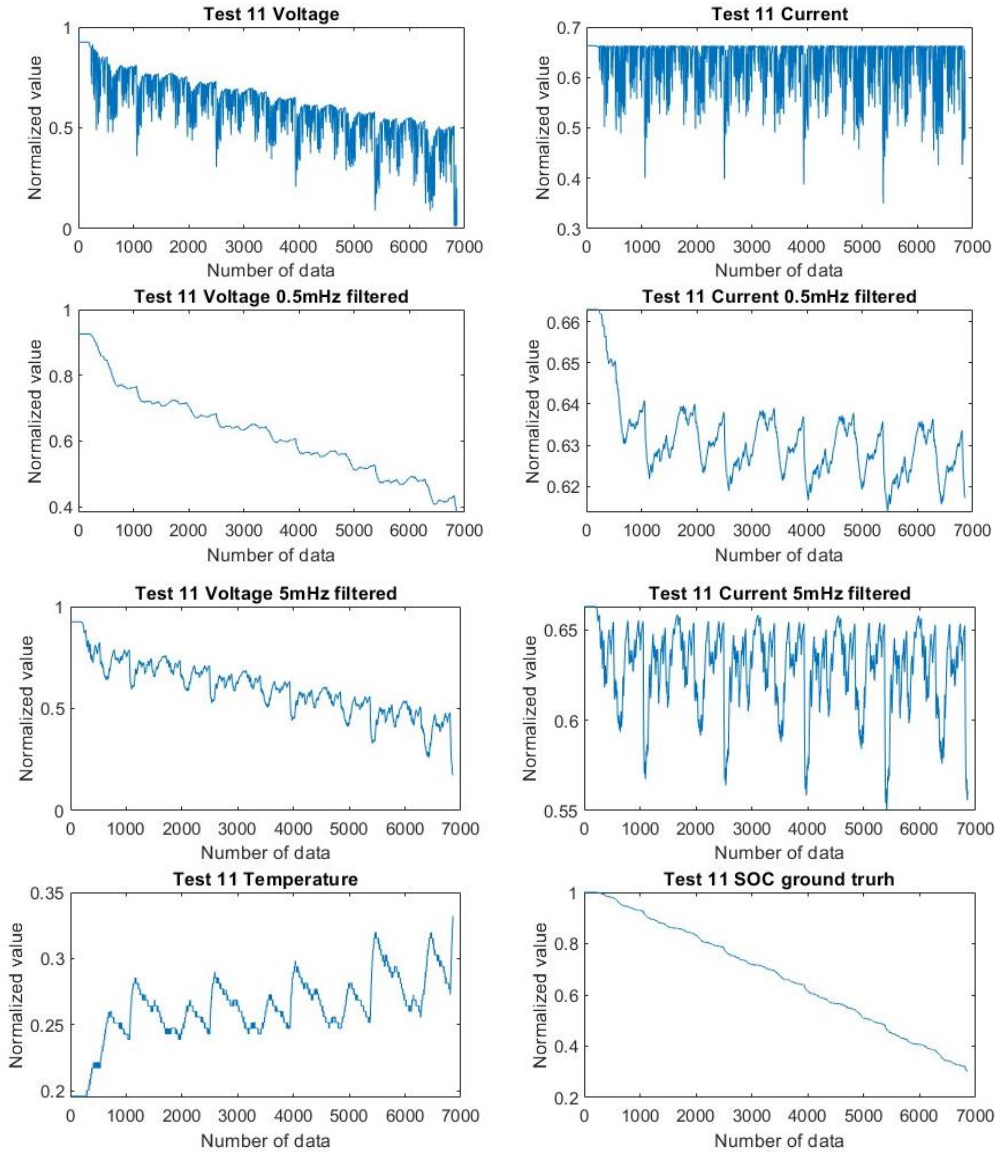
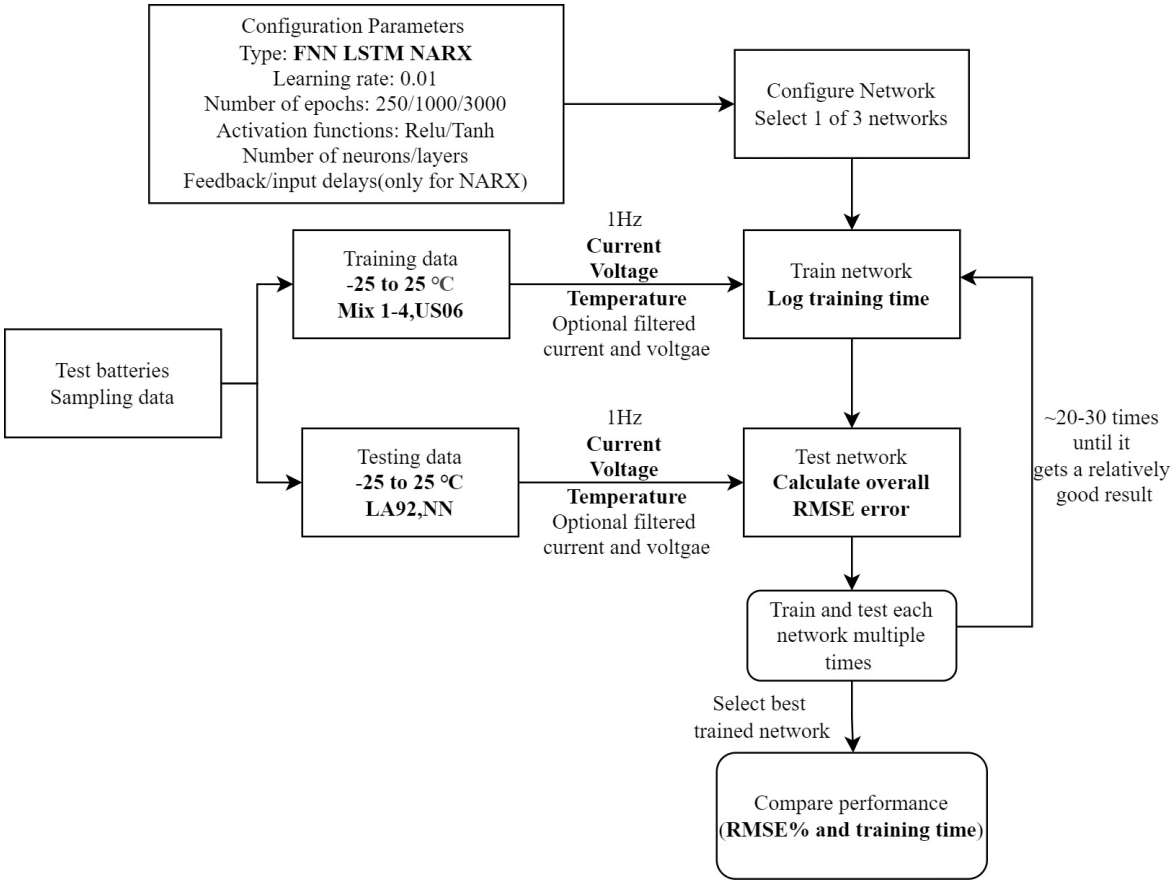


Fig.3.2.4 characteristics of 11th, -10 degree-testing dataset

3.2.2 Training and Testing

Fig.3.2.3 and Fig.3.2.4 show every input and SOC characteristics of selected two testing data, when the data preparation is ready, training process can be start, we first follow some training procedures and sequences as following (Summarized in a Flow chart.1 below):



Flow chart.1 : Training and testing algorithm

LSTM:

- 3inputs(V, I, T) and 250/1000/3000 epochs respectively.
- 5inputs{T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz} and 250/1000/3000 epochs respectively.

FNN:

- 3inputs(V, I, T) and 250/1000/3000 epochs respectively.
- 5inputs{T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz} and 250/1000/3000 epochs respectively.

NARX with an additional while loop(based on 25 and -10 degree/25 and 10 degree):

- 3inputs(V, I, T).
- 5inputs{T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz}.

Each network is trained for 20-30 times until it gets a relatively good result.

3.3 Results and performance comparison

In this section, the configurations and results of every training options based on different network have been analyzed, realize network performance comparison. Based on network characteristics, network optimization is realized, and a new round of performance comparison is performed for the optimized network.

3.3.1 Configurations and results

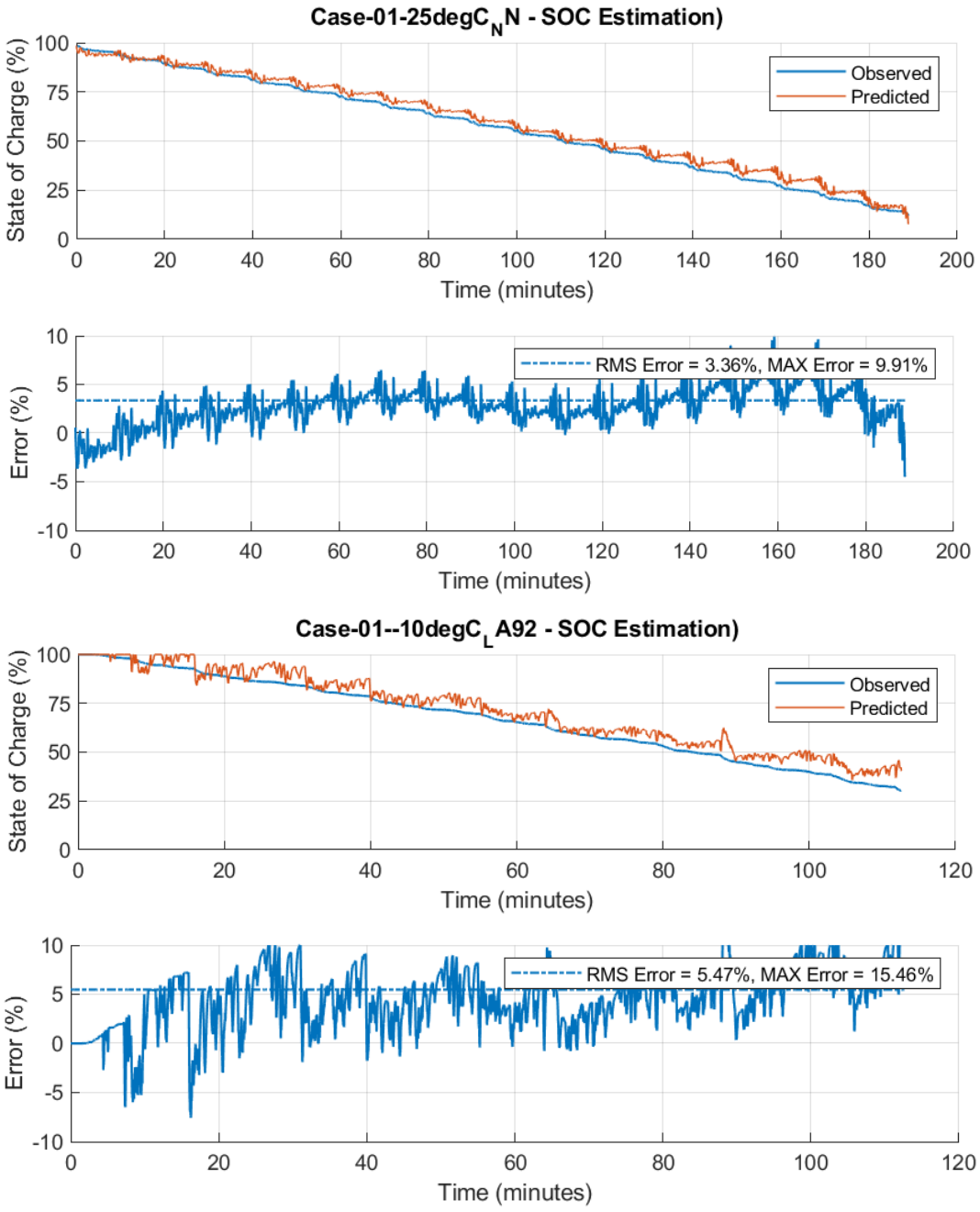
LSTM configurations:

```
numResponses = 1;  % number of output expected from the NN, in
this case the output is SOC
Epochs  =   250;   % maximum number of epochs, after reaching
this value the training will stop
                   % Epochs are set to 250/1000/3000
LearnRateDropPeriod = 2000;
InitialLearnRate=0.01;    %initial Learning rate value
LearnRateDropFactor=0.85; % after each "LearnRateDropPeriod", the
learning will be multiplied by "LearnRateDropFactor" number
validationFrequency = 3;

layers_LSTM = [sequenceInputLayer(numFeatures)
    lstmLayer(10,'OutputMode','sequence')
    fullyConnectedLayer(numResponses)
    clippedReluLayer(1)
     regressionLayer];
```

Results of LSTM (3 inputs):

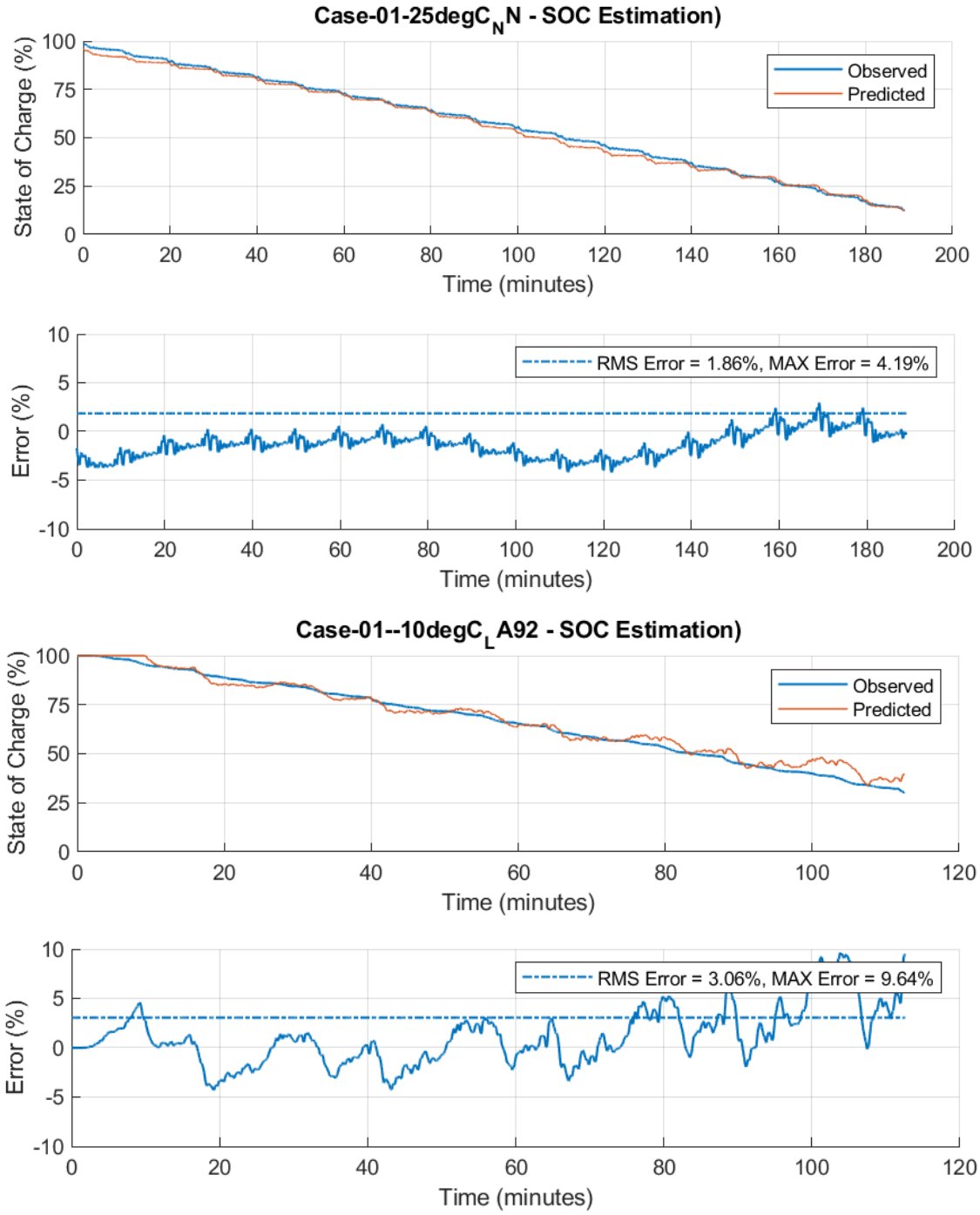
3000 Epochs: 25 degree testing (upper) and -10 degree testing (lower)



The results are shown from figures above, the LSTM network can work properly even there are just 3 inputs, but the estimated results have large oscillation and the RMSE% error behaves decreasing when the number of epochs goes from 250 to 3000. The network has best performance when the number of epoch equals to 3000 (shown in 3000 Epochs figures).

Results of LSTM (5 inputs):

3000 Epochs: 25 degree testing (upper) and -10 degree testing (lower)



LSTM performs much better when the inputs are 5: {T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz}, the oscillations are reduced and the network performs much stable, the root mean squared error percentage (RMSE%) is around 2%~3%.

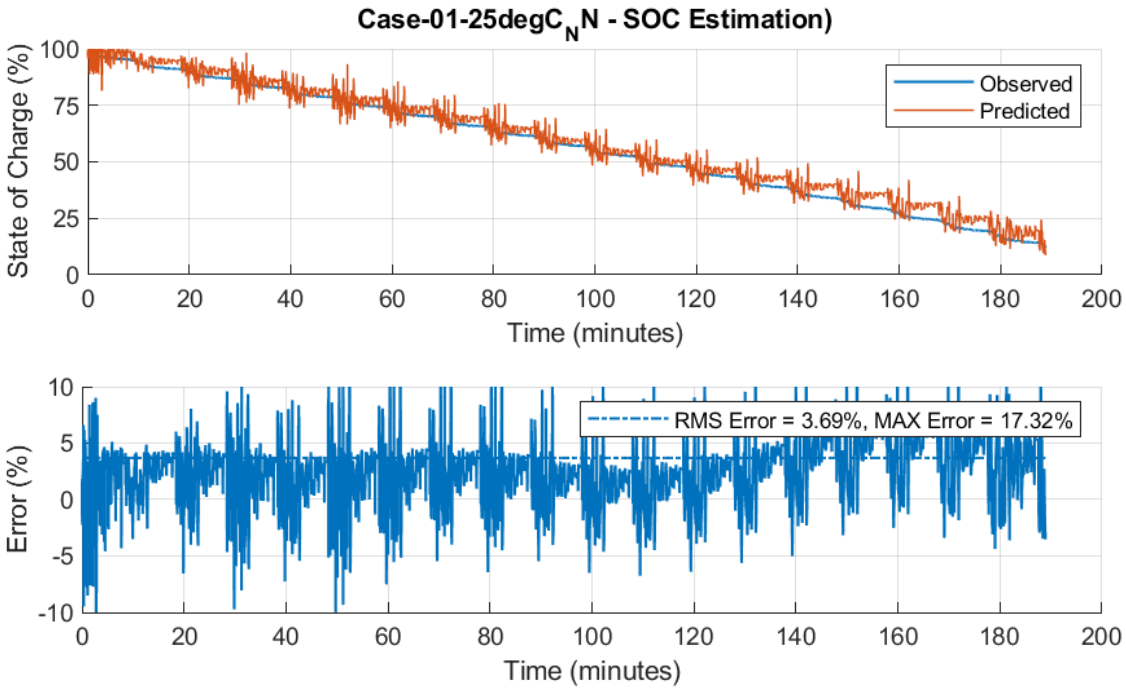
FNN Configuration:

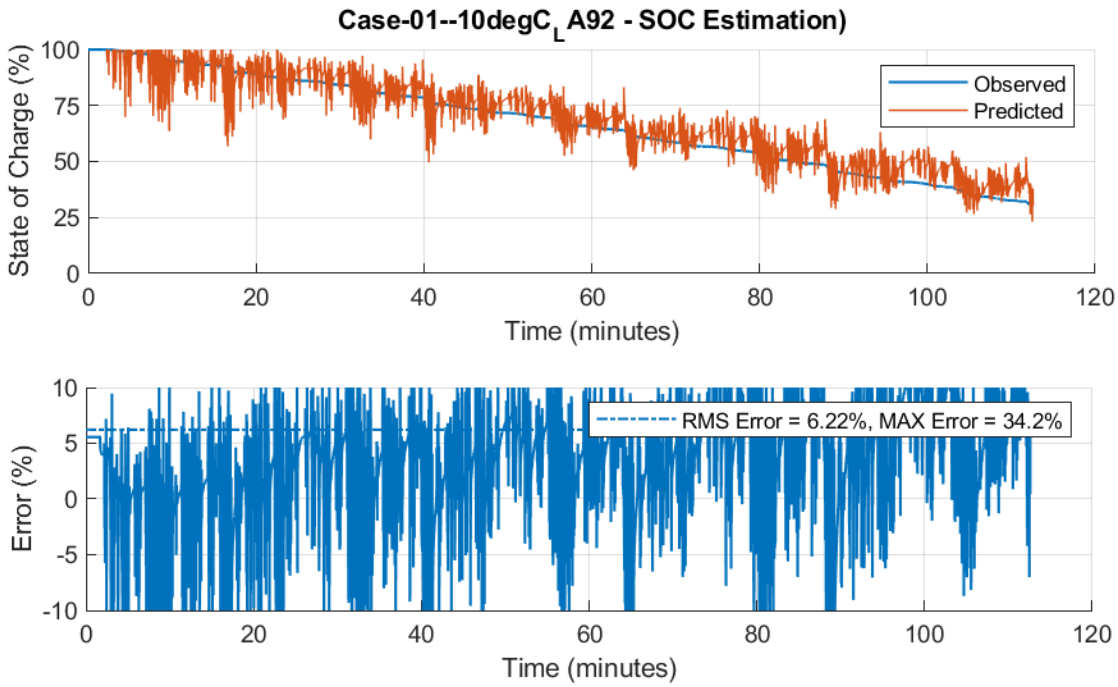
```
numResponses = 1;  % Number of output expected from the NN, in
this case the output is SOC
Epochs  =   250;   % Maximum number of epochs, after reaching
this value the training will stop
                   % Epochs are set to 250/1000/3000
LearnRateDropPeriod = 2000;
InitialLearnRate=0.01;    %Initial Learning rate value
LearnRateDropFactor=0.85; % After each "LearnRateDropPeriod", the
learning will be multiplied by "LearnRateDropFactor" number
validationFrequency = 3;

layers_FNN = [sequenceInputLayer(numFeatures)
    fullyConnectedLayer(21)
    reluLayer
    fullyConnectedLayer(19)
    reluLayer
    fullyConnectedLayer(numResponses)
    reluLayer
    regressionLayer];
```

Results of FNN (3 inputs):

3000 Epochs: 25 degree testing (upper) and -10 degree testing (lower)

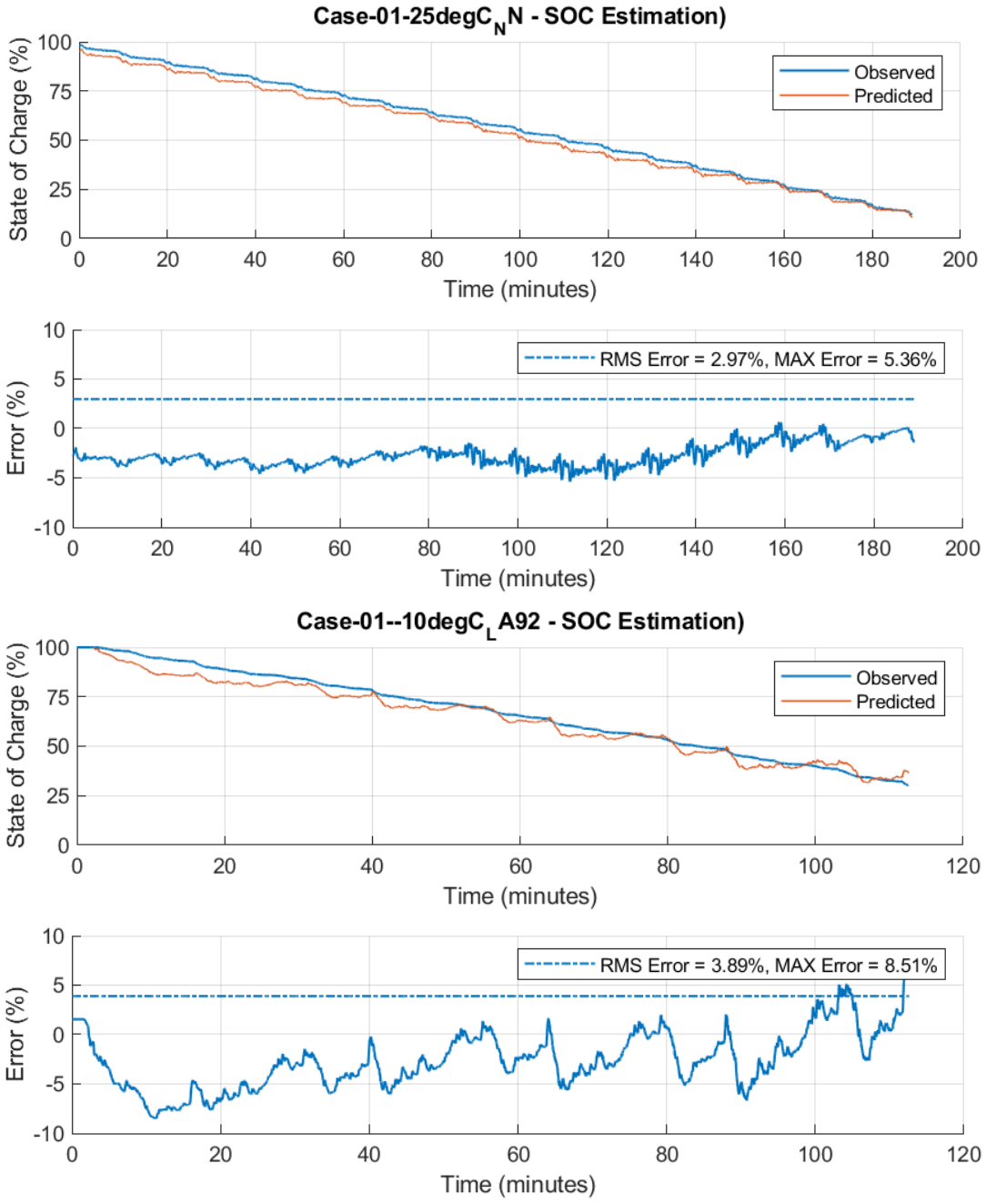




The training speed of FNN is much faster with compared to LSTM (4 times faster), but from the figures above, the estimated result performs large oscillation, as well as very poor performance in minus temperature. Considering the real application scenarios, the large oscillation SOC values (shown in Results of FNN (3 inputs)) are not acceptable even though the overall RMSE% error is not so high. Because the SOC is going to be monitored on the dashboard of the applied vehicle, the large oscillation results will lead to a very bad perception and annoying the driver and passengers, then lead to some wrong decision based on wrong SOC value.

Results of FNN (5 inputs):

3000 Epochs: 25 degree testing (upper) and -10 degree testing (lower)

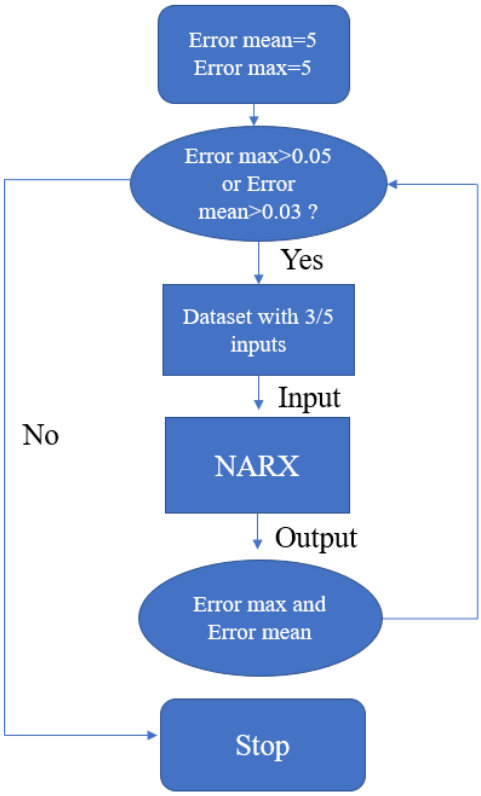


When the inputs are 5{T, V_0.5mHz, I_0.5mHz, V_5mHz, I_5mHz}, The performance becomes much better with compared to 3 inputs (V, I, T). While the training time remains nearly the same. The root mean squared error percentage (RMSE%) is around 3%~4%.

When filtered current and voltage are feasible, or the original data behaves as low oscillation, FNN network is recommended since its faster and it has relatively high performance.

NARX configuration and additional loop:

```
inputDelays = 1:2;
feedbackDelays = 1:2;
hiddenLayer = randi([12 20],1,1);
net = narxnet(inputDelays,feedbackDelays,hiddenLayer,
'open',trainFcn);
    net.trainParam.lr=0.01;
    net.divideParam.trainRatio = 70/100;
    net.divideParam.valRatio = 15/100;
    net.divideParam.testRatio = 15/100;
    net.trainParam.epochs = 3000;
    net.trainParam.min_grad = 9e-12;
    net.trainParam.max_fail = 200;
```

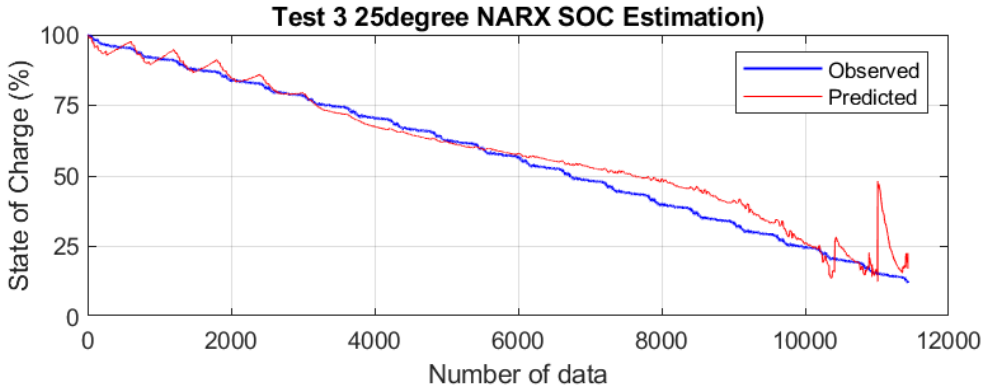


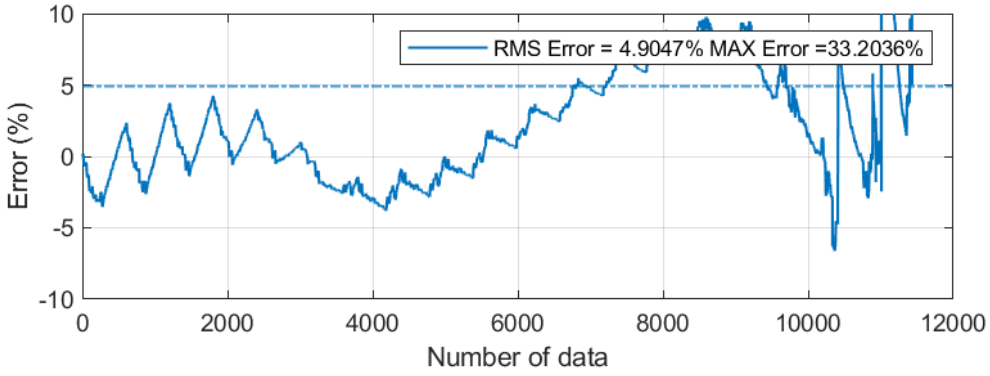
Flow chart.2: Logic of additional algorithm for NARX

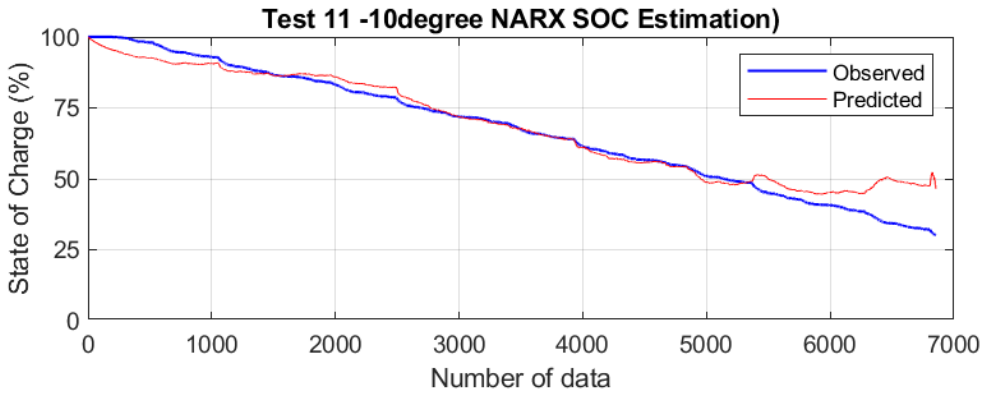
Since NARX net is trained based on another toolbox (Machine learning toolbox), an additional training loop is introduced to the code, in order to get a relatively stable and high performance network. Max error and min error are set by user in a while loop, number of hidden layers is obtained by a rand[5 20] code, and the net converges better when the number equals to 15-18.

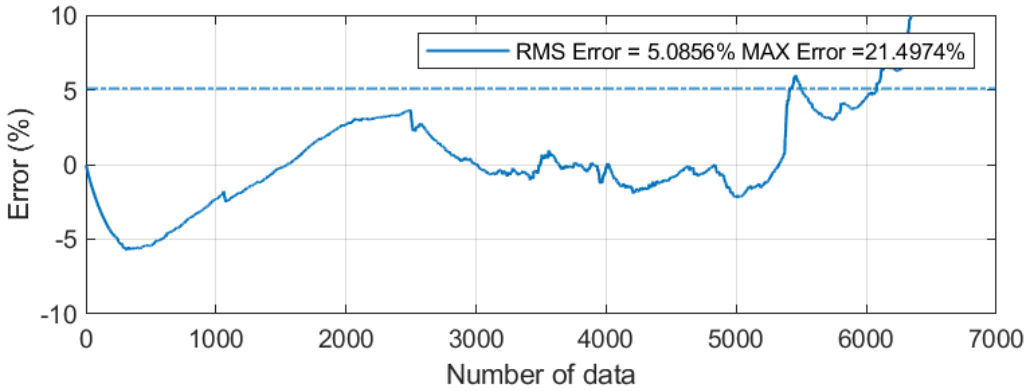
Results of NARX (3 inputs):

3000 Epochs: 25 degree testing (upper) and -10 degree testing (lower)









Since the training process stopped when the validation number reach the target value (shown in Fig.3.3.1.1):

Training Progress

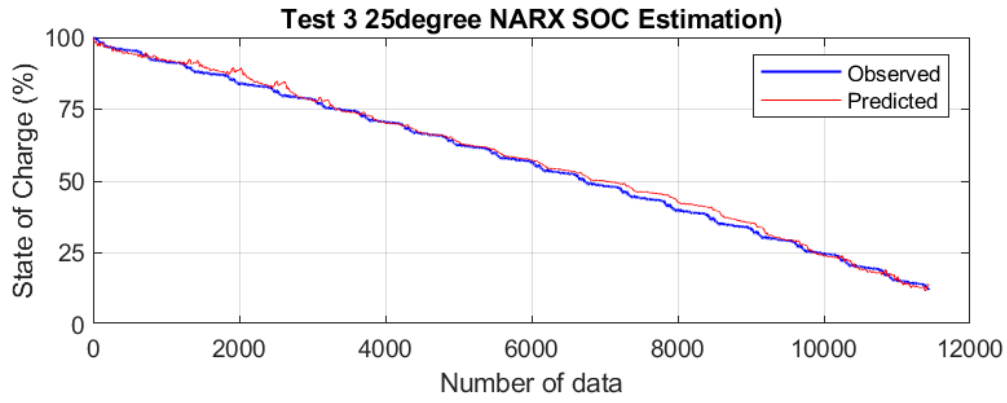
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	495	3000
Elapsed Time	-	00:03:45	-
Performance	0.759	5.96e-05	9e-12
Gradient	2.52	3.61e-05	9e-12
Mu	0.001	1e-08	1e+10
Validation Checks	0	200	200

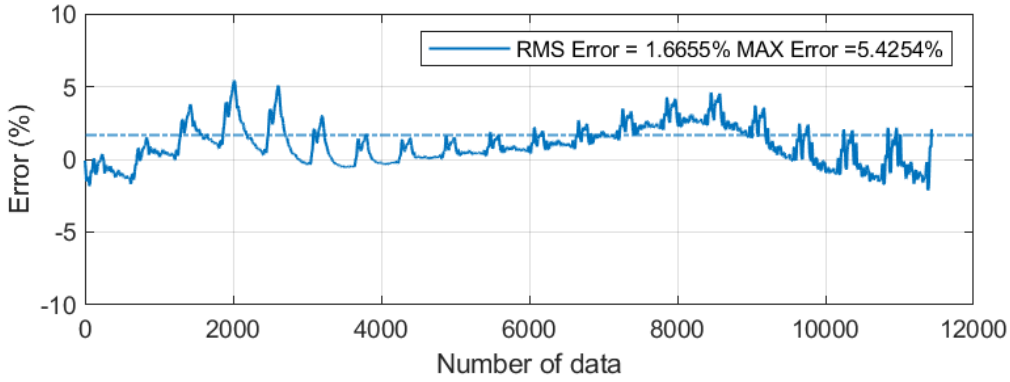
Fig.3.3.1.1 Process of training for NARX

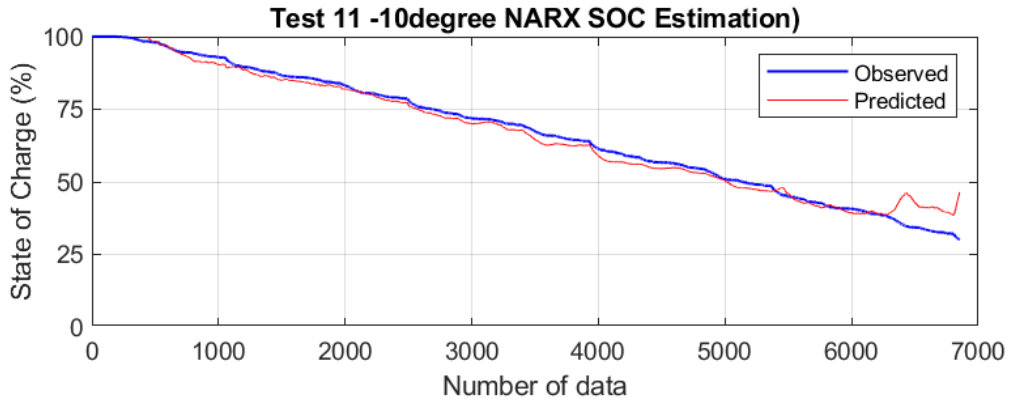
```
net.trainParam.max_fail = 200;
net.trainParam.epochs = 3000;
```

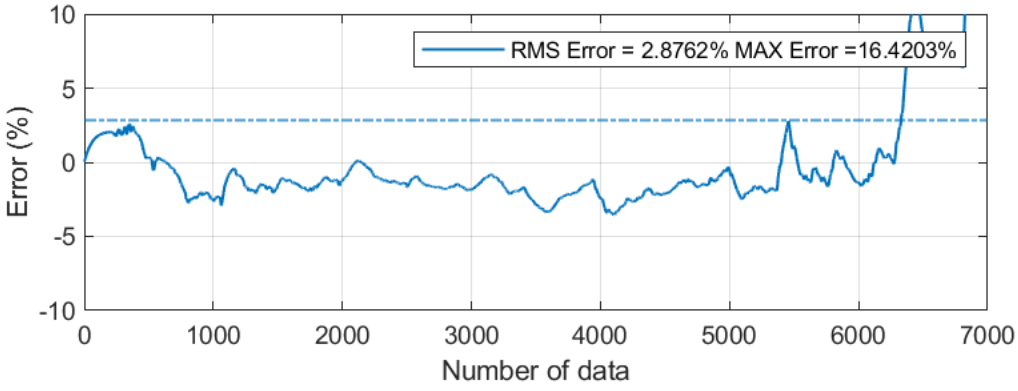
When the validation error increases for a specified number of iterations (net.trainParam.max_fail), the training is stopped, and the weights and biases at the minimum of the validation error are returned. In this way, the number of epochs and training time of every single training phase are not fixed (Training is finished before reaching the target number of epochs).

NARX with 5 inputs (Train based on 25 and -10 degrees):

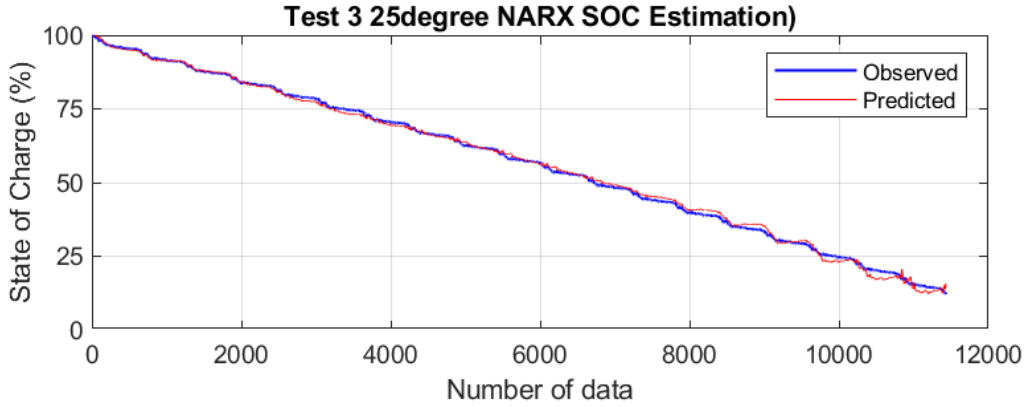


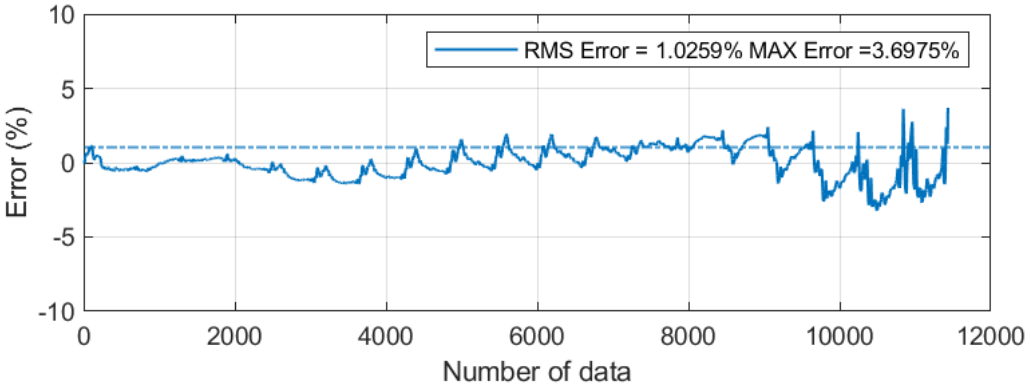


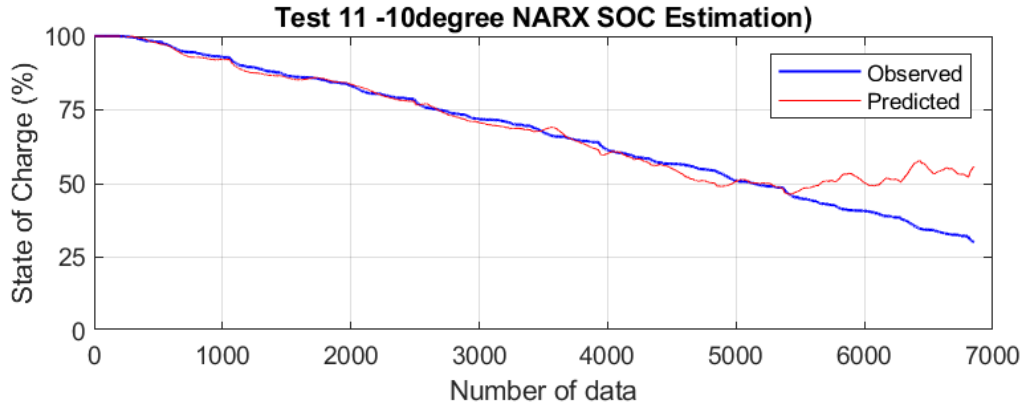


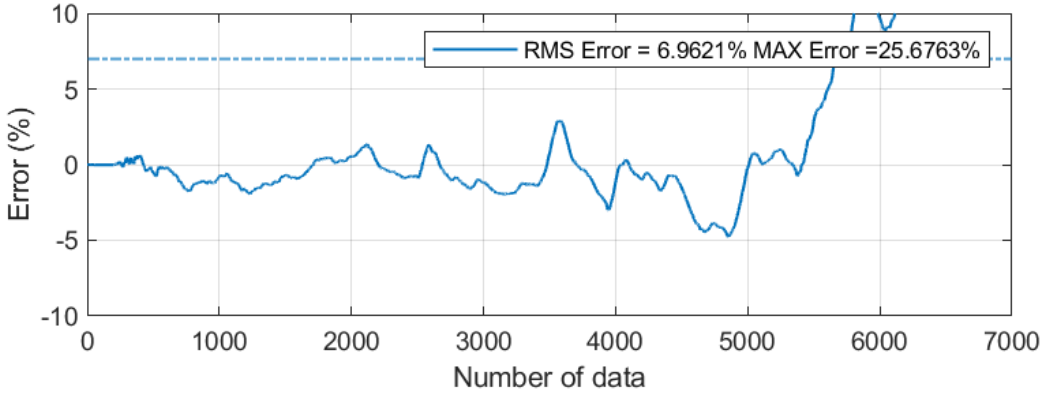


NARX with 5 inputs (Train based on 25 and 10 degrees):









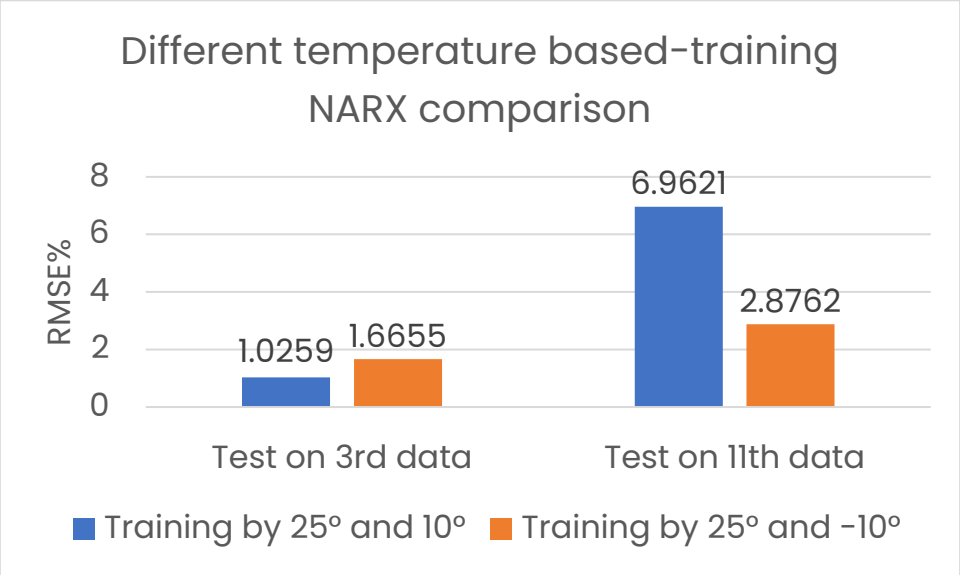


Fig.3.2.5 Different temperature based NARX comparison

Fig.3.2.5 implements the comparison between two different temperature data based training results. These results are taken from the above performance figures.

Training by 25° and 10° (better positive temperature performance): Training process is stopped when the test error on 25 degree and 10 degree reach the goal.

Training by 25° and -10° (better generalization by wide temperature range): Training process is stopped when the test error on 25 degree and -10 degree reach the goal.

From these comparison we can see that in order to make the NARX net has better robustness in order to withstand the wider testing temperature, the second choice which is ‘Training by 25° and -10°’ is suggested. After finishing each training option and getting the results of three different networks, the first gross performance comparison can be done.

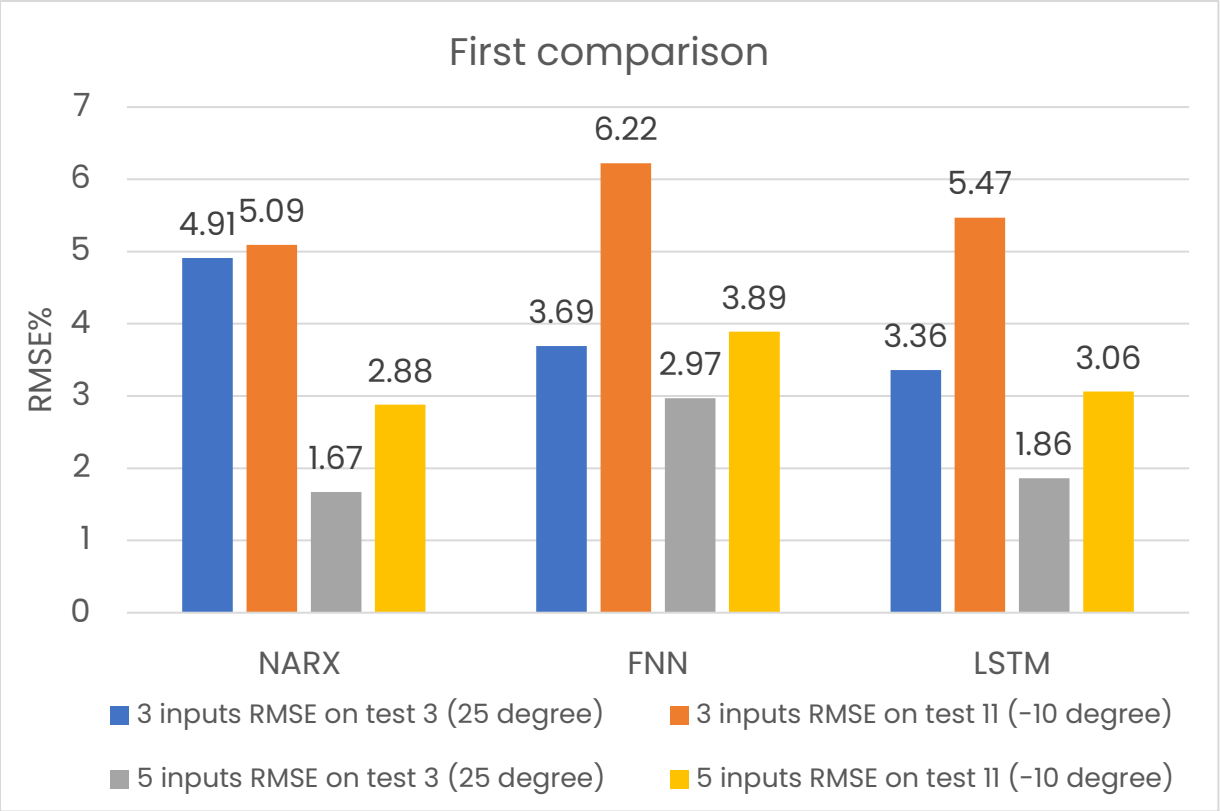


Fig.3.2.5 First comparison

From the first comparison (Fig.3.2.5) we can see that the NARX net is improved the most when the input number changes from 3 to 5. And the performance depends strongly on the testing data (different testing data and temperature) we put in the while loop, Considering the non-stable convergence behavior (not only the performance, but also the training time) of the previous NARX, the structure development is mandatory, in order to get a relatively stable NARX net when applying training every time.

3.3.2 Network optimization

Since we just focused on test3 (25 degree) and test11 (-10degree)/test3 (10degree), changing the number of inputs to see the results in a simple way, we can have a gross perception of each networks performance on effect of inputs and different temperatures.

In order to apply it on real scenarios, we have to make higher the networks' generalization, obtain higher robustness. So we are going to change the networks' structure and consider overall RMSE (mean RMSE between 15 different testing data) to obtain better performance.

FNN optimization:

Since FNN/LSTM and NARX are trained by different AI toolbox (machine learning toolbox and deep learning toolbox), and machine learning toolbox for NARX is feasible for training FNN, so an effective comparison between two different toolbox is necessary.

In this optimization step, an additional FNN is trained by machine learning toolbox, by the same data fed to NARX network, and the performance is recorded in the performance comparison section.

Transform FNN to machine learning toolbox and compare to other networks:

Put a same structure into machine learning toolbox, but the input data is the whole data without separating into minibatches (the same data used in NARX):

```
layers_FNN = [sequenceInputLayer(numFeatures)
    fullyConnectedLayer(21)
    reluLayer
    fullyConnectedLayer(19)
    reluLayer
    fullyConnectedLayer(numResponses)
    reluLayer
    regressionLayer];
```

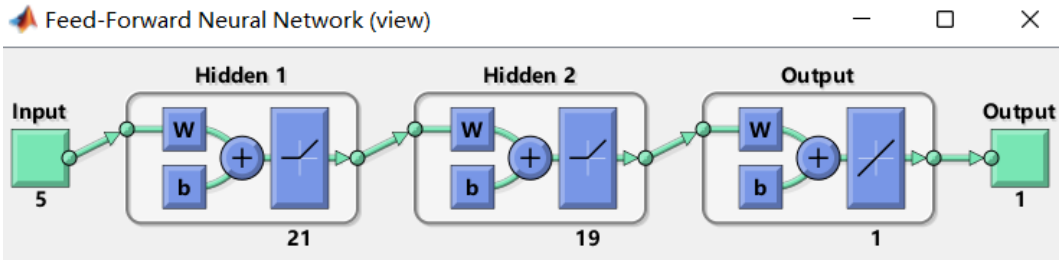
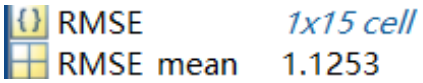


Fig.3.2.6 FNN in the machine learning tool box

Change activation functions of FNN from 'tanh' into 'Relu', change some other hyperparameters:

```
net.trainParam.lr=0.01;
net.trainParam.epochs=3000;
net.layers{1}.transferFcn='poslin'
net.layers{2}.transferFcn='poslin'
```

Test the network on all the testing data, and compute an overall RMSE% (RMSE% mean value), to see if the model has a robust generalization:



RMSE

1x15 cell

	1	2	3	4	5	6	7	8	9	10	11	
1	0.6892	0.2579	0.3572	1.0128	0.6319	0.4937	0.8775	0.6671	1.0998	0.6749	1.5009	1.55

Fig.3.2.7 RMSE% for each different testing data and RMSE% mean

Same procedure applied to LSTM and FNN in deep learning toolbox

Hyperparameters and results for FNN:

```
numResponses = 1;  % number of output expected from the NN, in
this case the output is SOC
Epochs  =   250;   % maximum number of epochs, after reaching
this value the training will stop
                   % Epochs are set to 250/1000/3000
LearnRateDropPeriod = 2000;
InitialLearnRate=0.02;    %initial Learning rate value
LearnRateDropFactor=0.85; % after each "LearnRateDropPeriod", the
learning will be multiplied by "LearnRateDropFactor" number
validationFrequency = 3;
layers_FNN = [sequenceInputLayer(numFeatures)
    fullyConnectedLayer(21)
    reluLayer
    fullyConnectedLayer(19)
    reluLayer
    fullyConnectedLayer(numResponses)
    reluLayer
    regressionLayer];
```

Test the network on all the testing data, and compute an overall RMSE% (RMSE% mean value), to see if the model has a robust generalization:

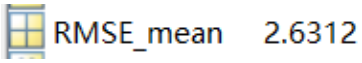


Fig.3.2.8 RMSE% mean of FNN

Hyperparameters and results for LSTM:

```
numResponses = 1;  % number of output expected from the NN, in this
case the output is SOC
Epochs  =   250;   % maximum number of epochs, after reaching this
value the training will stop
                   % Epochs are set to 250/1000/3000
LearnRateDropPeriod = 2000;
InitialLearnRate=0.02;    %initial Learning rate value
LearnRateDropFactor=0.85; % after each "LearnRateDropPeriod", the
learning will be multiplied by "LearnRateDropFactor" number
validationFrequency = 3;

layers_LSTM = [sequenceInputLayer(numFeatures)
    lstmLayer(10,'OutputMode','sequence')
    fullyConnectedLayer(numResponses)
    clippedReluLayer(1)
    regressionLayer];
```

Test the network on all the testing data, and compute an overall RMSE% (RMSE% mean value), to see if the model has a robust generalization:

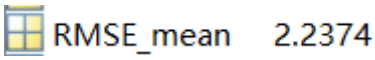


Fig.3.2.9 RMSE% mean of LSTM

NARX optimization

Since there are many noises and oscillations inside each input data, too long series input time delays may introduce the noise and oscillations into the network again, this causes the negative effect for parameters inside the network and it may not perform very well.

For this reason, we have to remove the input delay and change it to '0:1', and just take one output value to feedback, so change feedback delay to '1'.

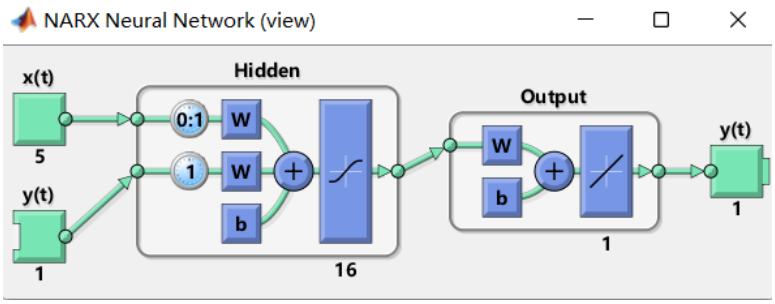


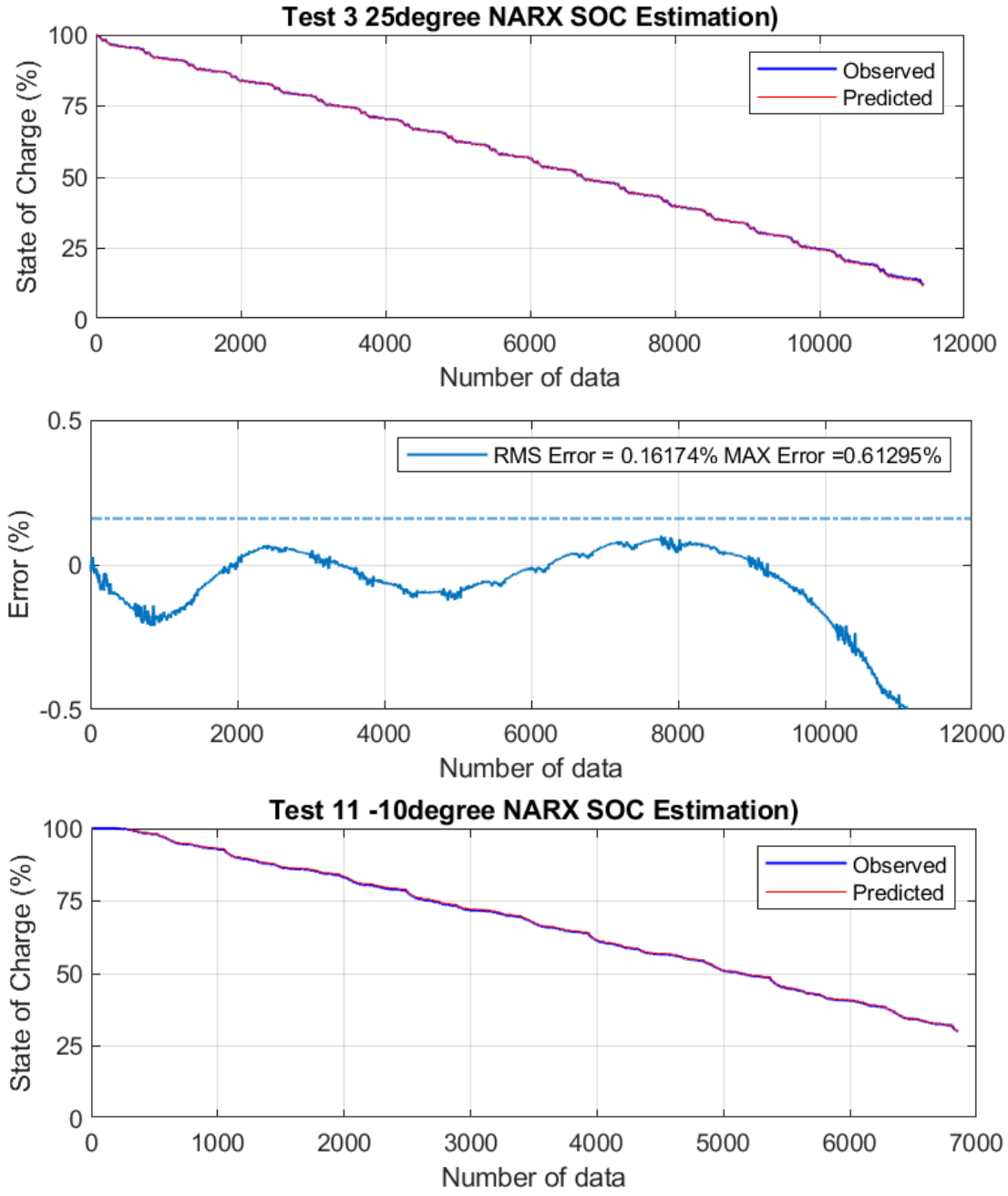
Fig.3.2.8 NARX with '0:1' input delay and '1' feedback delay

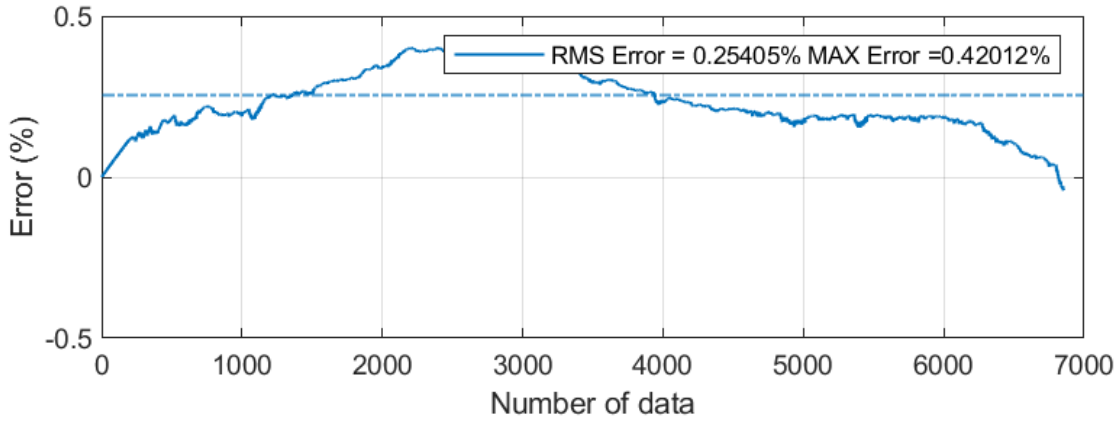
In order to optimize the network, we put RMSE_mean into the while loop, in order to get a better global generalization network.

RMSE	*1x15 cell*
RMSE_mean	0.2995

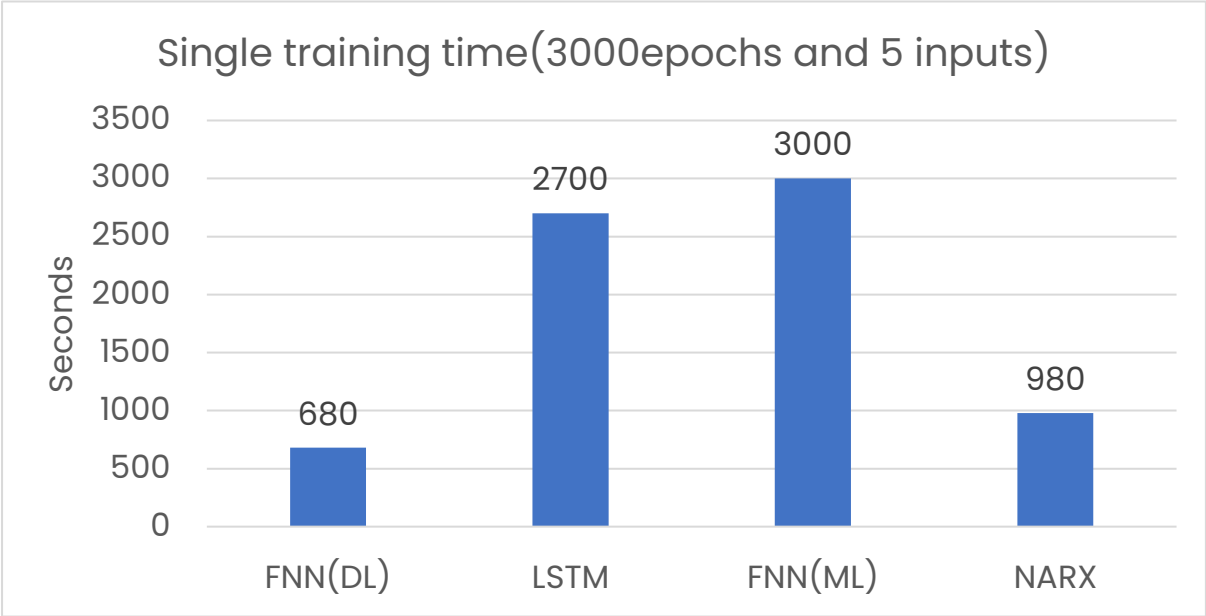
Fig.3.2.7 RMSE% for each different testing data and RMSE% mean

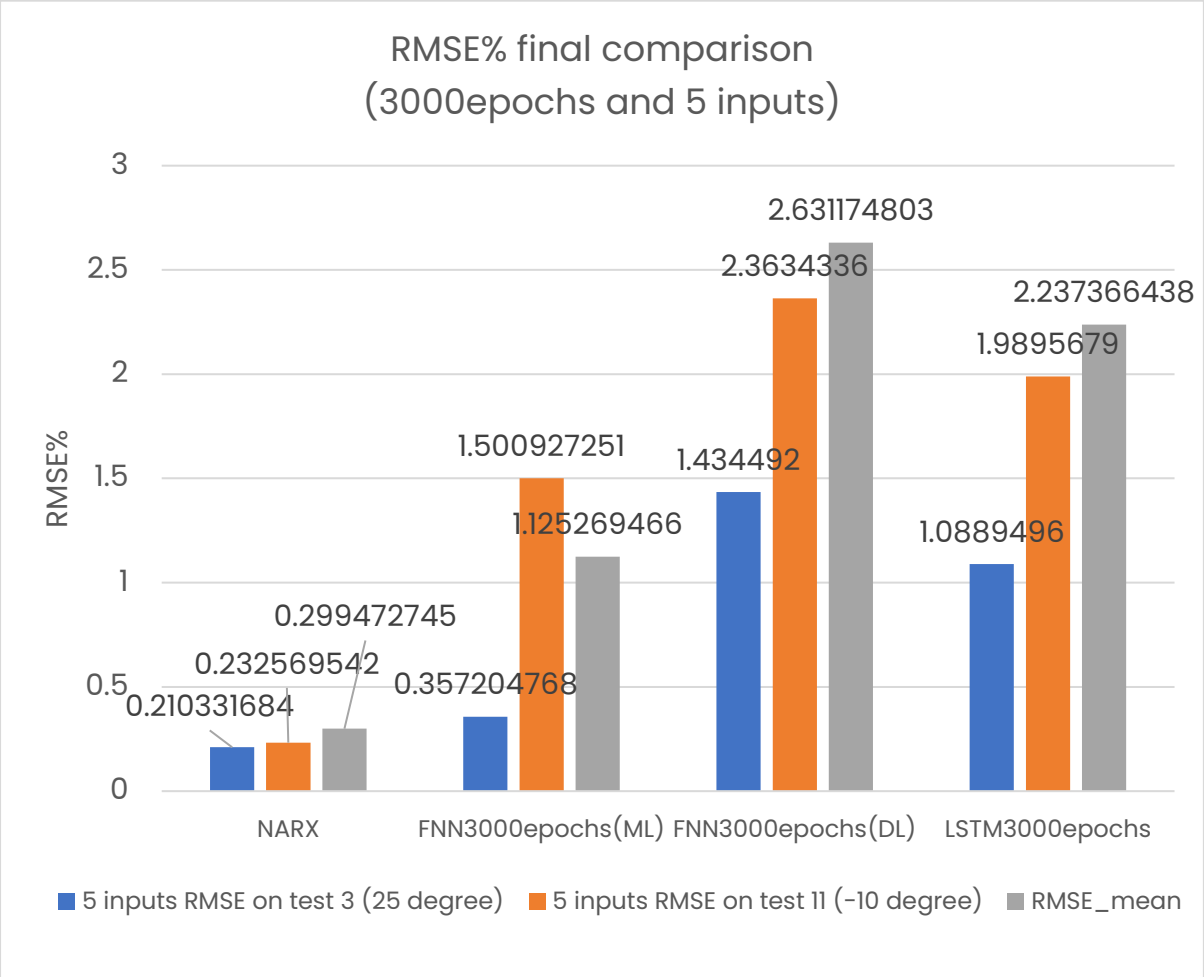
Testing result for datasets corresponding to different temperature:





3.3.3 Second performance comparison





NARX network has the dominating performance among 4 networks (two FNN networks), the mean RMSE% ~0.3%, the single training time~1000 seconds (for 3000 epochs). It has not only really low error, but also fast speed (lightweight). For further exploration, NARX network is selected as the optimal SOC estimation network.

4 SOC estimation considering SOH

Based on the SOC estimation comparison from Chapter 3, a high performance network for estimating SOC is obtained. While if we consider the real-time and realistic application for Electric or hybrid electric vehicles, the parameter SOH should not be ignored. Fig.3 shows the non-linear relationship between SOC and SOH. In fact, if SOH is considered for SOC networks, the estimation could be more precise (SOC network only work on its dedicated SOH). In the Chapter.4, a new specific dataset is used, which include many different battery sampling phases. Some of the phases are created for accelerating battery aging, the intention is to take advantage of some of the characteristics after aging. In this way, this dataset is suitable for both SOC and SOH networks, and actually they are two parts that cannot be separated from each other.

4.1 Specific dataset introduction and selection

The data used for the experiments were acquired from an in-house developed test bench. The battery module which connected to the test bench is consists of six battery cells connected in series is. And the cell voltage is measured using Elithion battery plates (Equipment showed in Fig.4.1.1).

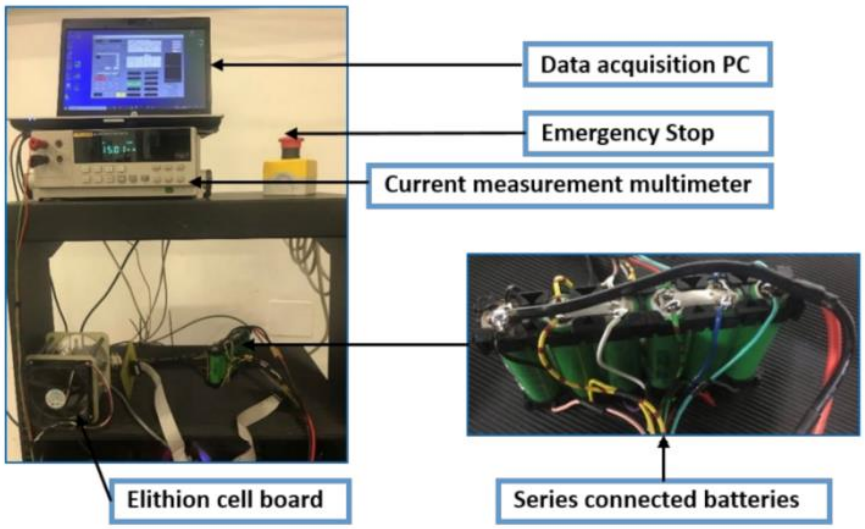


Fig.4.1.1 Experimental setup for data acquisition from lithium-ion batteries connected in series [20]

Two LM35 Texas Instrument temperature sensors are used to measure the cell surface temperature. The Elithion (Lithulmate) BMS is mounted on the test bench to increase the security of the acquisition process. An Arduino Mega board is connected via LAN to a dedicated PC, which is then used to acquire the measurement data. From the safety point of view, the system is equipped with an emergency stop device as an additional safety measure.

The battery cell used in this section is Sony Murata US18650VTC6 (Fig.4.1.2):

Parameters	Value	Unit
Nominal Capacity	3120	mAh
Rated Capacity	3000	mAh
Nomial Voltage	3.6	V
Internal Impedance	13	mΩ
Standard Charge Condition		
Charge Up Voltage	4.2 ± 0.05	V
Charge Current	3	A
Charge Time	2.3	h
Ambient Temperature	23	℃

Fig.4.1.2 Sony Murata US18650VTC6

The composition of the acquired data is shown in the flow chart of the following figure (Fig 4.1.4). The entire test plan is repeated 55 times to observe the battery aging phenomenon. It means in these 43 different phases, each of which will be repeated 55 times.

The battery test plan can be simplified into repeatedly Charging-Discharging phases (green block represents Charging phase, while red block represents Discharging phase, shown in Fig.4.1.3) The load profiles are either charge or discharge phase or a sequence of charge and discharge as in the dynamic load profile. In the later stages of phase, a special algorithm called Random Walk is applied to two dedicated phases, which are 37: discharging polarized random walk and 41: charging random walk.

Random Walk (RW), also known as random wandering or random walking, is a mathematical statistical model consisting of a sequence of trajectories, each of which is random. It can be used to represent irregular forms of variation, as in the case of a random process record formed by a person's drunken and disorderly walk. Therefore, it is the basic statistical model for recording random activities. The data which random walk algorithm is applied is shown in (Fig.4.1.3).

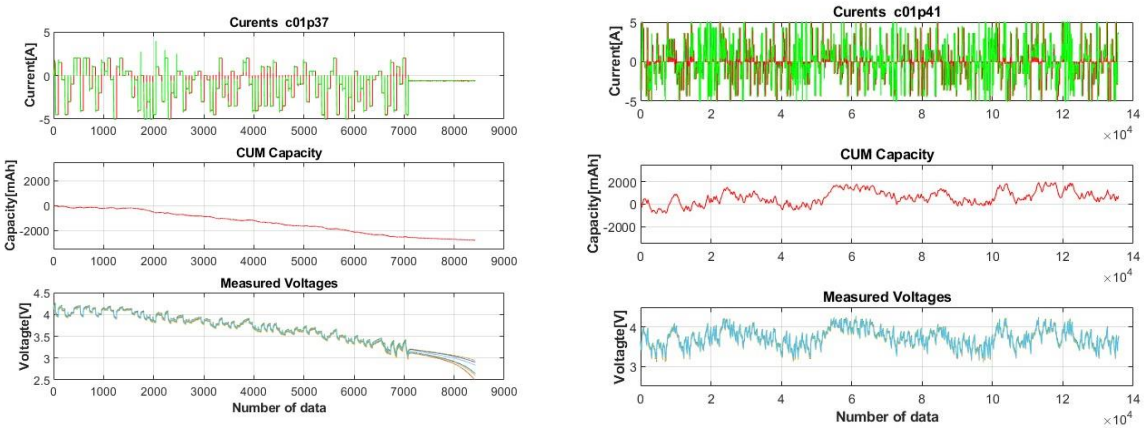


Fig.4.1.3 Phase 37: PRW (left) and Phase 41: RW (right)

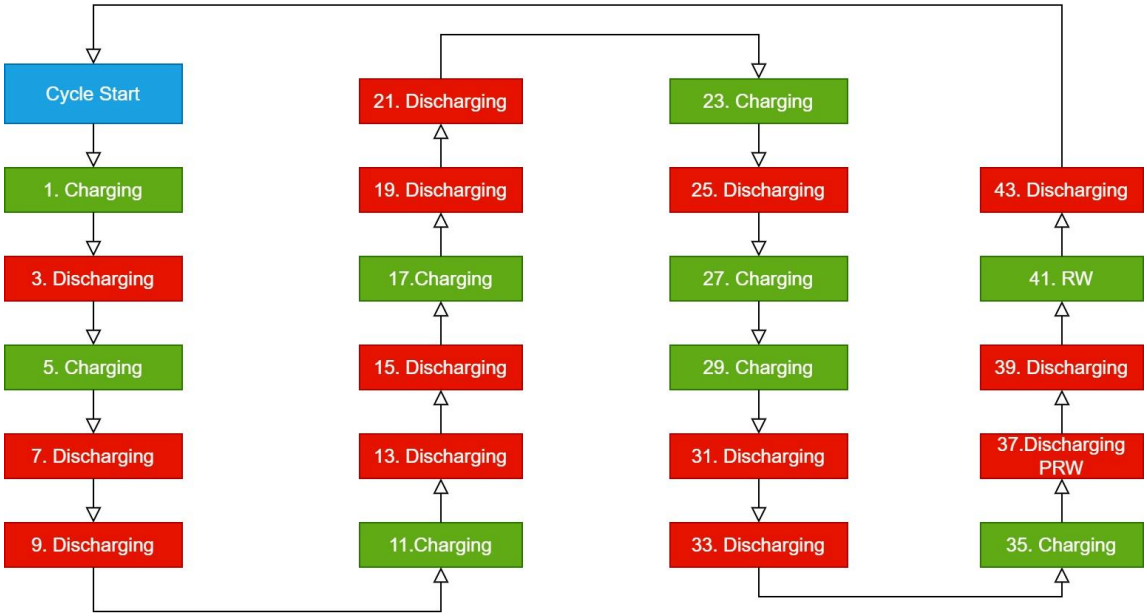


Fig.4.1.4 Sony Murata US18650VTC6 simplified test plan

When we get into details of the test plan (Fig.4.1.5), the profiles include different charging/discharging rate(5/C,2/C,C), constant CC (1.675A) -CV (0.35A) charging current, constant discharging voltage (2.5V). The specific parameter for Phase 37-PRW is: 5A for 60 seconds, 4.2V/3V to 2.5V. And for Phase 41-RW is: 5A current for 300 seconds, 4.2V/3V.

From the network performance and generalization point of view, the SOC estimation network is required to estimate both charging and discharging profiles.

From the realistic use of vehicles, during the charging, vehicles are usually parked in fixed locations, such as charging stations, garages at home. So the charging phase is usually a stable process, without any discharging profiles or any rapid-change/pulse of currents/voltage. In this way, a pure charging phase (CC-CV) will be selected for training the network, in order

to let the network identify and estimate the charging process, from 0% to 100% SOC.

In the opposite way, the discharging phase of a vehicle is usually an uncertainty process, Complex working conditions and unexpected factors, such as sudden acceleration and sudden braking, may occur at any time. Also for electric vehicles, regenerative braking (Current flows in the opposite direction, SOC increases corresponds to Fig.1 and coulomb counting equation) is an important function that cannot be ignored. So the discharging process, or the working process of an EV, is definitely not a pure discharging process. For this consideration, a complex discharging phase should be selected for network.

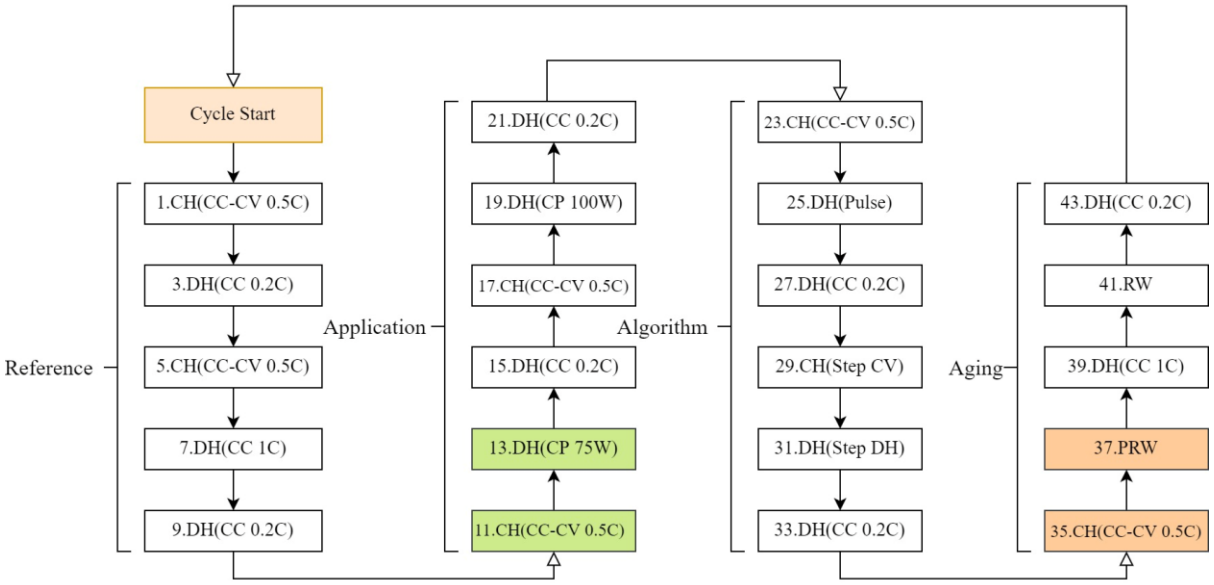


Fig.4.1.5 Sony Murata US18650VTC6 test plan

In the author's continuous experiments, and considering robustness/generalization of the NARX net, the training dataset, as well as the testing dataset are selected (marked in figure 4.1.5, green block represents training data, orange block represents testing data). Each data's profile are shown below(Fig.4.1.6 and Fig.4.1.7).

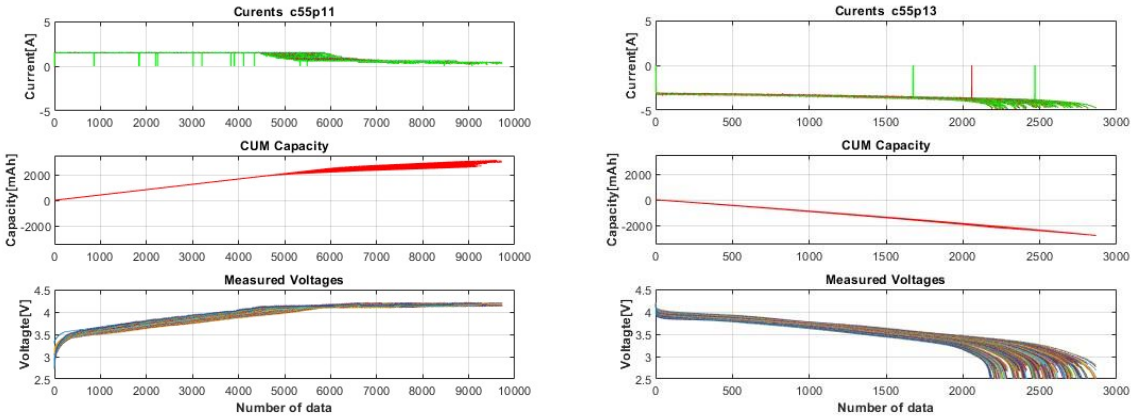


Fig.4.1.6 Phase 11(left) and Phase 13(right) used for training process

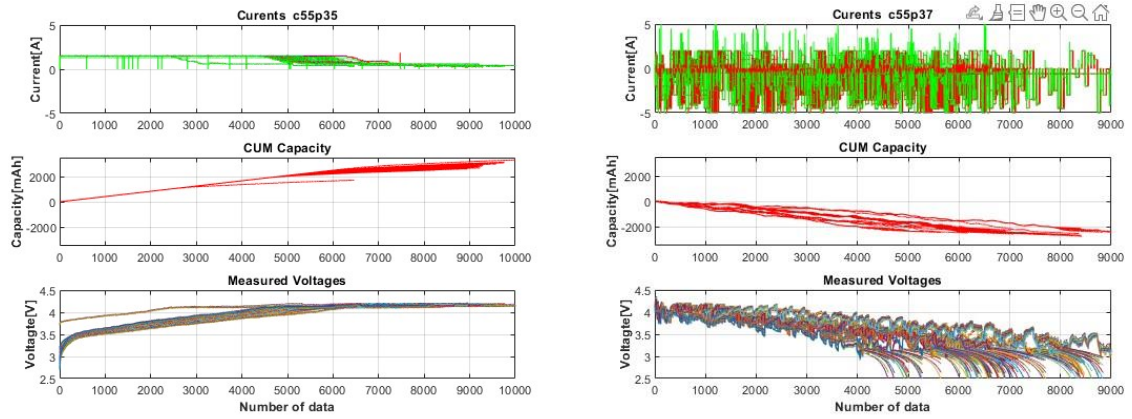


Fig.4.1.7 Phase 35 (left) and Phase 37 (right) used for testing process

The selection criterion is that to ensure that the data between training and testing does not overlap, and increase the challenge to the network as much as possible.

Since there are six battery cells sampled in series, based on this criterion, the training data is taken from the first battery cell, then the testing data is taken from the third battery cell.

While Phase 11 (CC-CV pure charging) and phase 13 (CP pure discharging) dataset from the first battery cell are combined together for training. And Phase 35 (CC-CV pure charging) and phase 37 (Polarized random walk discharging) dataset from the third battery cell are combined together for testing.

The reason that phases used for training/testing are close to each other (11-13, 35-37) is to avoid relatively large battery capacity variation because of capacity decay after running multiple phases. This will cause errors and lead to poor training results. For example, the first battery's capacity after finishing phase 1 is not equal to the first battery's capacity after finishing phase 41.

The other reason for combined charging-discharging phase together is to avoid the rapid change of the dataset (the vertical lines appear in the Fig 4.1.8), due to the characteristic of NARX net's time series memory, it will affect the network's performance. At the same time, to avoid the rapid change of data, the combined charging-discharging profile is adjusted to 0%-100%-0%. Even though some of the cycles' SOC (end SOC for charging, start SOC for discharging) are not equals to 100%(For example: 1.02% or 99.8%).

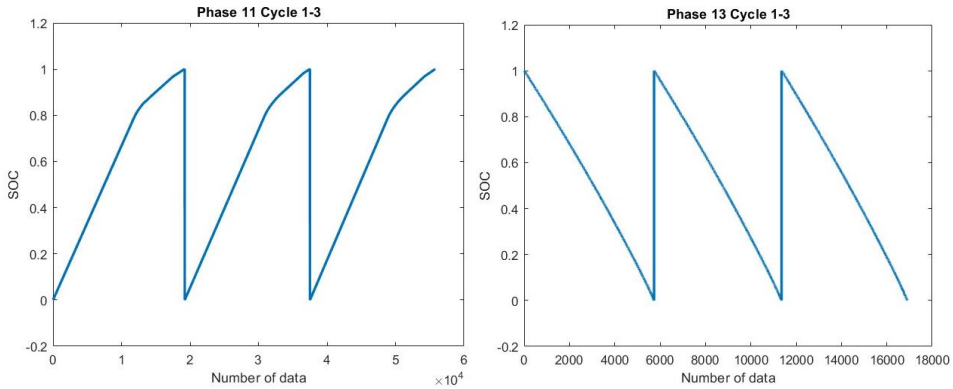


Fig 4.1.8 SOC ground truth profile of Phase 11 Cycle 1-3 (left) and Phase 13 Cycle 1-3 (right)

In this way, Continuous charging-discharging profiles are built up for training and testing. The simplified dataset combination schematic is shown in Fig 4.1.9 below:

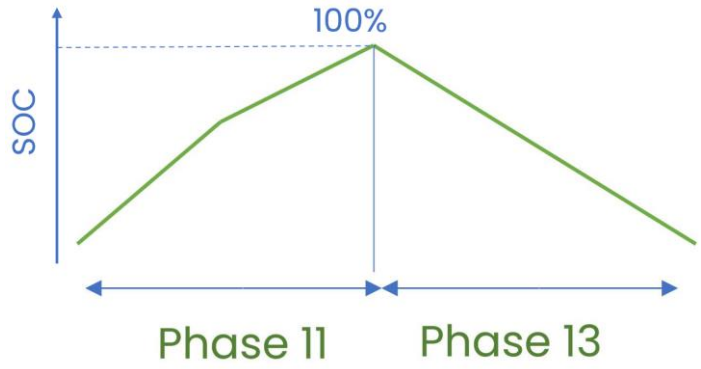


Fig.4.1.9 Phase 11 and Phase 13 combined for training process

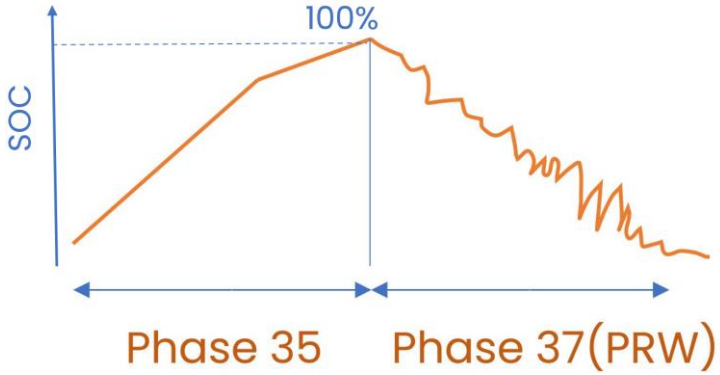


Fig.4.1.10 Phase 35 and Phase 37 combined for testing process

4.2 Training and testing methodologies

The updated training and testing procedures (shown in Flow chart.3) are very similar to Flow chart.1 in Chapter.3.2.2, but considering the best NARX configuration (Fig.4.2.1) and new battery's training/testing data (green and orange block discussed in 4.1):

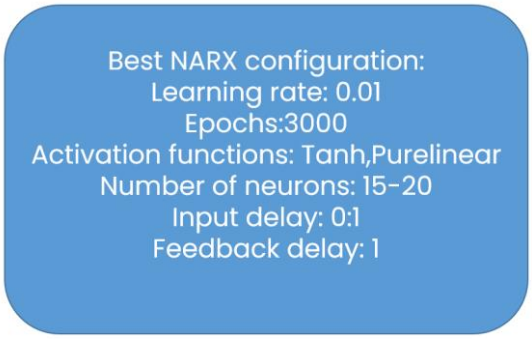
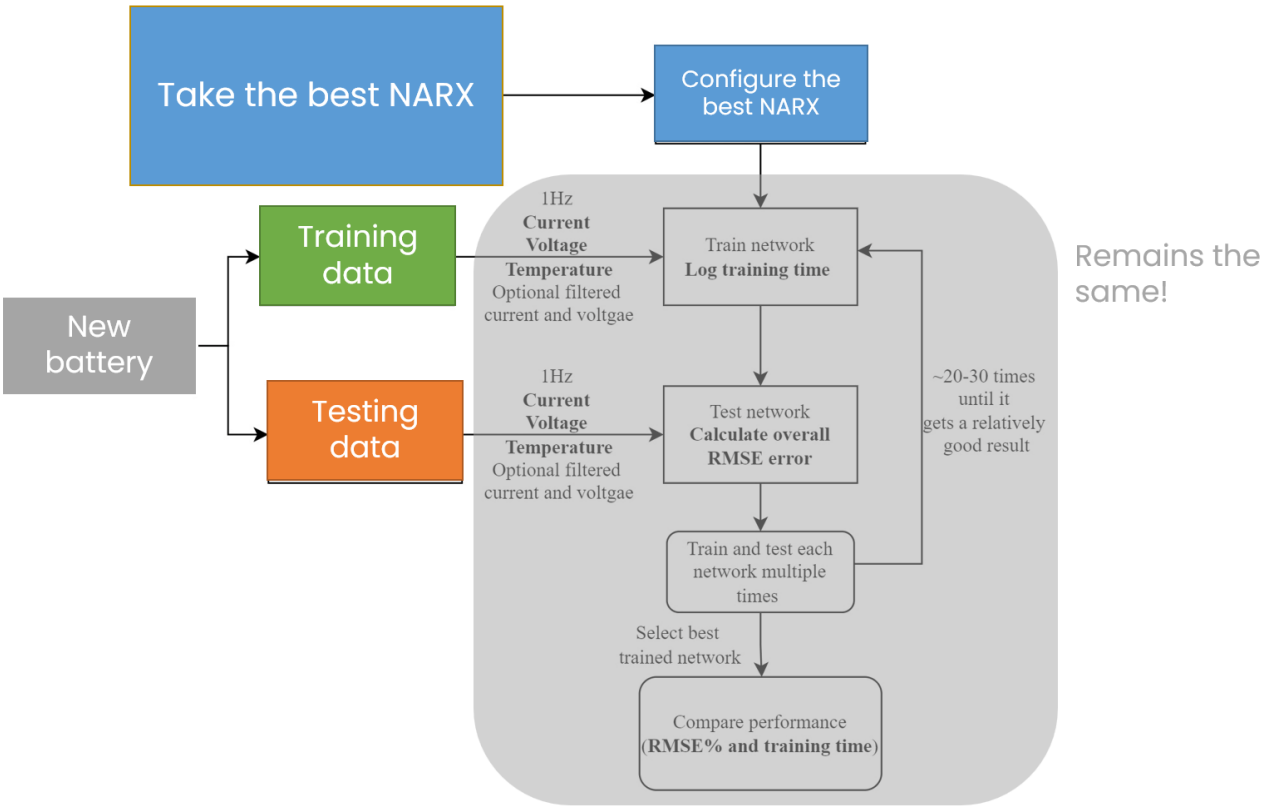


Fig.4.2.1 Best NARX network configuration



Flow chart.3 : Training and testing algorithm for the specific battery module

4.3 SOH interval assignment options and results

As we mentioned before, if SOH is considered for SOC networks, the estimation could be more precise (SOC network only work on its dedicated SOH). After knowing the training and testing methodologies, in this section, we would like to separate the SOH ground truth (Fig.4.3.1) into many different classes, and then training and testing the network's performance only on its dedicated SOH class range.

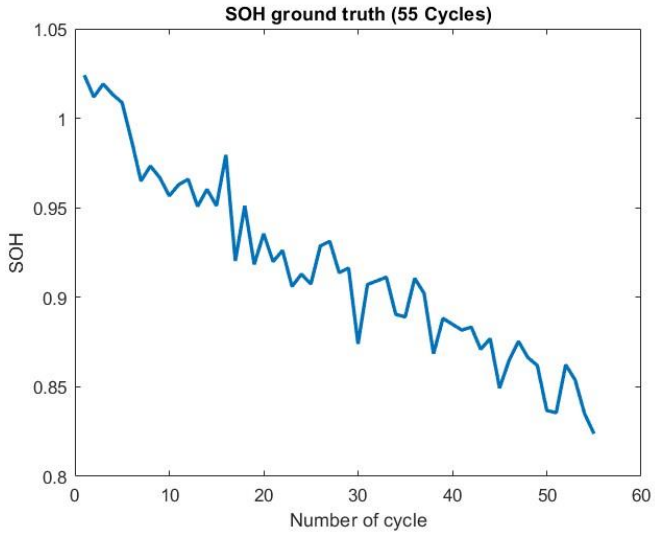


Fig.4.3.1 SOH ground truth of 55 cycles

From Fig.4.3.1 we can see SOH decrease from 1.0241 (Cycle 1) to 0.8238 (Cycle 55). Due to some uncontrollable sampling errors and the characteristics of the battery itself, the SOH profile is not pure linear, SOH value will also increase a little bit in some cycles. Since the overall decreasing trend is correct, we will not filter the SOH line to make it pure decreasing, because the SOH is computed from the cumulative capacity and maximum capacity, where the cumulative capacity is strongly depend on the current corresponding to its time discrete:

```
for i=1:52
   Capacity{i,1} = cumtrapz(meas{i,1}.time(:,1),meas{i,1}.Current(:,1))/3.6;
end
```

So a slightly change of SOH may lead to very poor training and testing performance of the network. Due to these characteristics, the interval assignment is mainly based on approximate SOH range and the number of cycles, three different options are considered: 3-Class, 5-Class and 1-Class with an additional SOH input (V, I, T and SOH).

3-Class assignment:

Class 1 : 1-17 Cycles SOH: 100%~95%

Class 2 : 18-34 Cycles SOH: 95%-90%

Class 3 : 35-55 Cycles SOH: 90%-82%

5-Class assignment:

Class 1 : 1-11 Cycles SOH: 100%~96%

Class 2 : 12-22 Cycles SOH: 96%-93%

Class 3 : 23-33 Cycles SOH: 93%-90%

Class 4 : 34-44 Cycles SOH: 90%-87%

Class 5 : 45-55 Cycles SOH: 87%-82%

1-Class assignment:

1-55 Cycles SOH: 100%-82%

Based on these classes and cycles assignment, the training and testing process (section 4.2) will be done to each of them. The original data (55 cycles) will be separated and then merged into three/four columns (or three/four rows). Fig.4.3.2 shows an example of 3-class training and testing data.

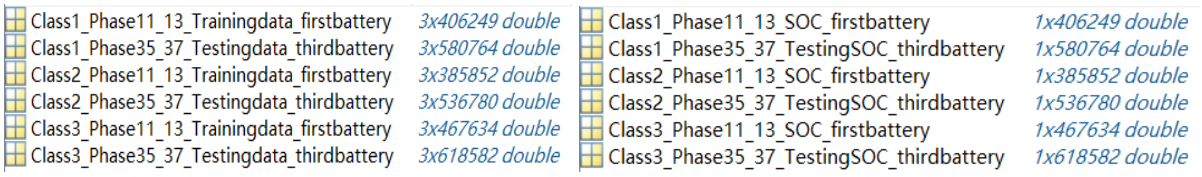


Fig.4.3.2 Training and testing data for the first class of 3-class option

In Fig.4.3.2, phase11_13 means phase 11 and phase 13 are combined and used for training , while phase35_37 means phase 35 and phase 37 are combined and used for testing (Correspond to section 4.1). [Current, Voltage , Temperature] are placed as three rows, each line has been merged 17-cycle's data. The ground truth SOC value has been calculate and extract as training and testing data. Then the output of NARX net will be compared to this ground truth during training and testing phase. The same procedure is applied to 5-class option.

For 1-Class option, the fourth SOH input is needed. Since SOH value of each cycle is a single ground truth number, the SOH input data should be set to the same dimension as the other

inputs (Fig.4.3.3), in order to implement training and testing properly(All the SOH values from the same cycle are set to the same value correspond to SOH ground truth, Fig.4.3.1. Testing SOH data is applied by the same procedure, since the SOH estimation network is not available yet).

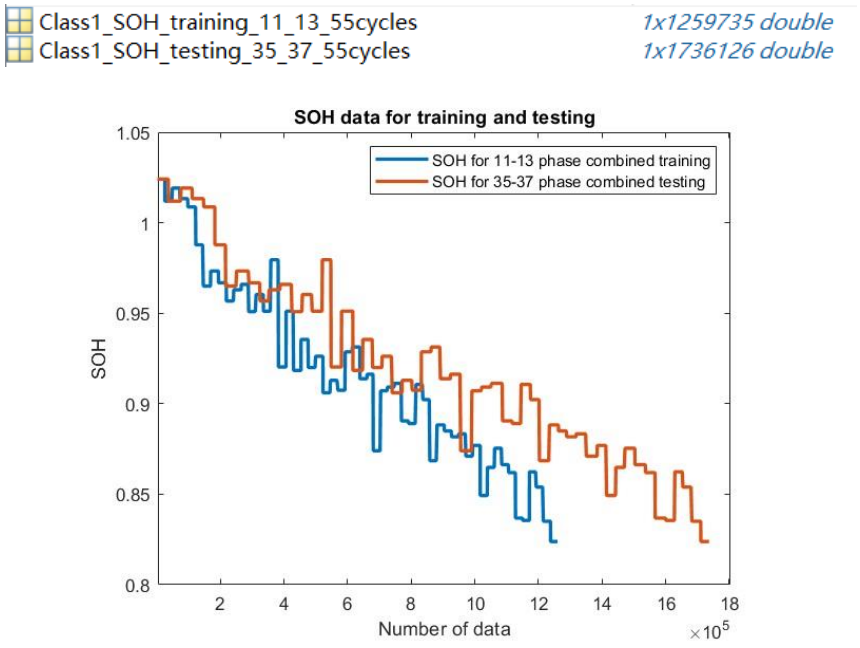


Fig.4.3.3 SOH training and testing data used for 1-Class option

After data organization is finished, training and testing are implemented. Then results and single training times for each option are shown below(Fig.4.3.4 and Fig.4.3.5):

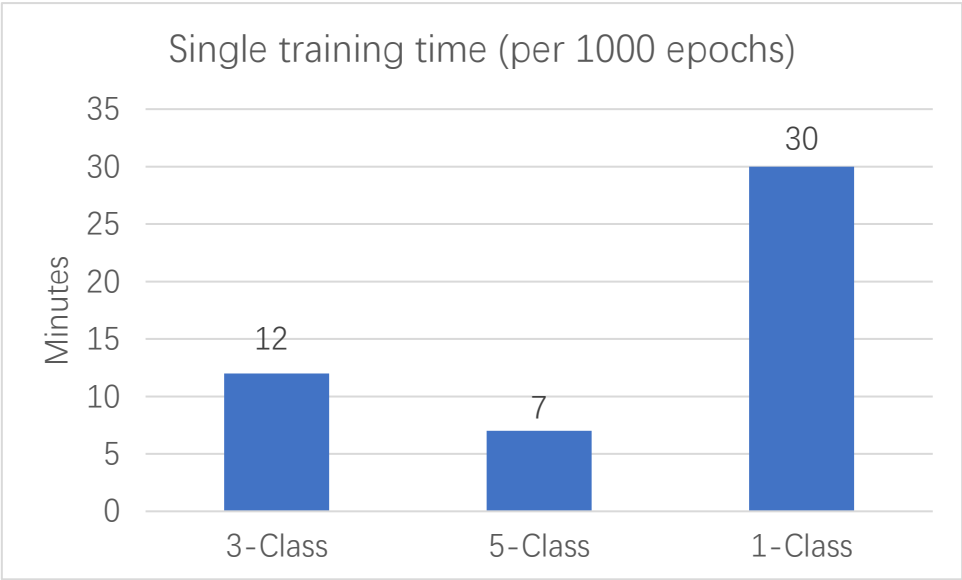


Fig.4.3.4 Single training time(per 1000 epochs) for each class from different option

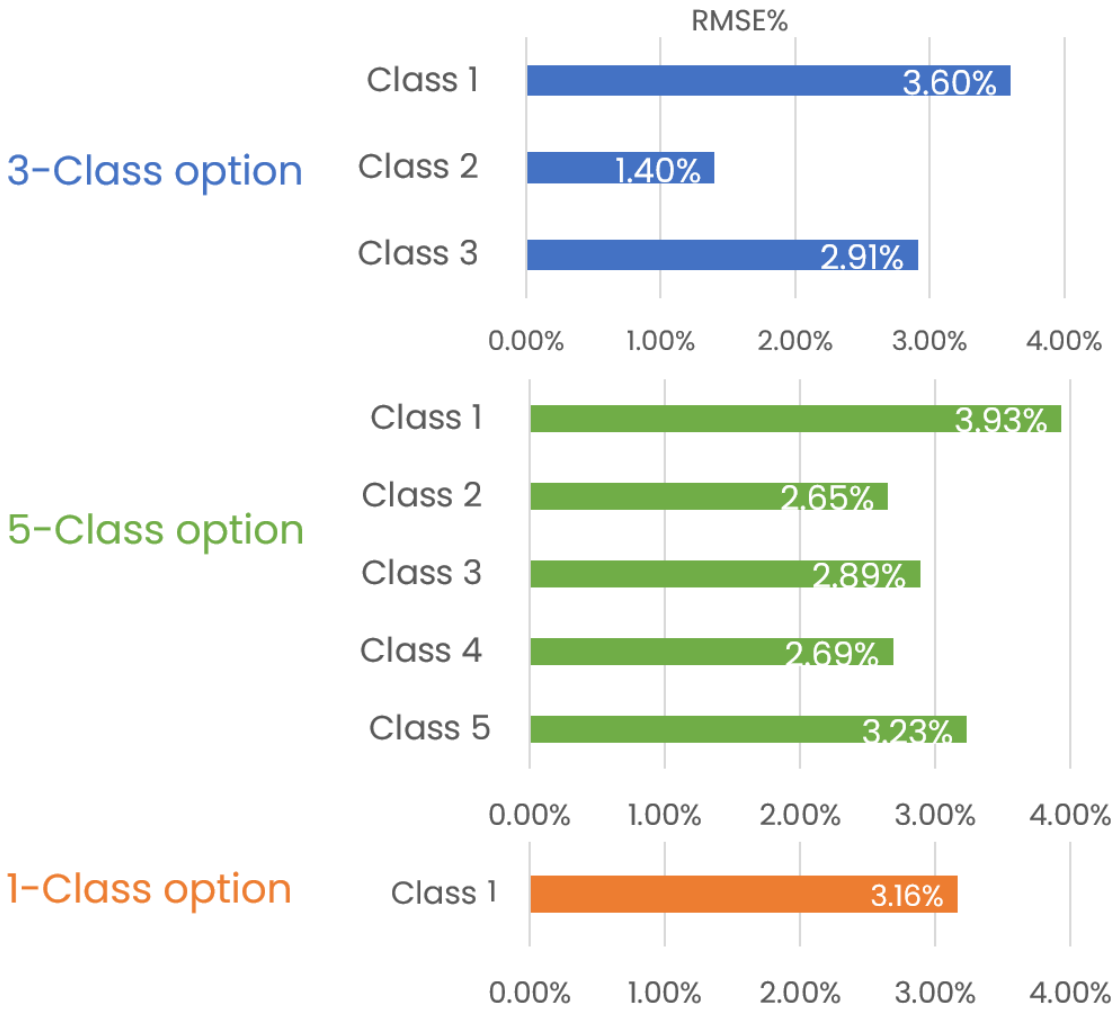


Fig 4.3.5 Results of three options (expressed in RMSE%)

From the results (Fig.4.3.5), we can see that the errors are relatively close to each other. (~3% error), from (Fig.4.3.4) we can see that the single training time of each class from different option is corresponding to the number of training data. And the time is recorded for each single class. (For example, the training time for the first class from 3-Class option is 12mins (1000 epochs), the total training time for 3-Class option could be $12x3 = 36mins$, 5-Class option could be $7x5 = 35mins$).

From the gross comparison of Fig.4.3.4 and Fig.4.3.5 we cannot say which option is better or faster. In fact, these gross comparison are done without the SOH estimation network, in other words, without combination system. So the further exploration of network performance of three options will be done in the next chapter.

4.4 Extract networks into Simulink models

Before the effective network combination, the model pre-trained in Matlab needs to be exported to Simulink, so as to implement the combination and application of the two system (SOC estimation system and SOH estimation system) in a more intuitive block diagram.

The code 'genism' is used to initialize the SOC network:

```
[sysName,netName] = gensim(net,'InputMode','Workspace',...
'OutputMode','WorkSpace','SolverMode','Discrete');
```

Before the code 'genism' is used, a proper testing data should be fed to the network, in order to give the NARX network a 'initial memory', otherwise the network will not work in the correct way as it should be. For example:

```
load("first _class_3_class_option.mat")
load("Testing35_37_Cycle_3.mat")
testingdata= Testing35_37_Cycle_3; %Cycle3 from Phase 35 and 37
combined data
Utest(1,:) = testingdata(1,:);     %Testing current
Utest(2,:) = testingdata(2,:);     %Testing voltage
Utest(3,:) = testingdata(3,:);     %Testing temperature
SOC= testingdata(4,:);             %Testing SOC
Y1 = SOC;
x = tonndata(Utest,true,false);
t = tonndata(Y1,true,false);
net = closeloop(net)            %Close the NARX feedback delay
loop for testing
[xs,xi,ai,ts] = preparets(net,x,{},t);
y=net(xs,xi,ai);                %Prediction SOC ouput
[sysName,netName] = gensim(net,'InputMode','Workspace',...
'OutputMode','WorkSpace','SolverMode','Discrete');
x1 = nndata2sim(x,1,1);
```

The NARX network 'first _class_3_class_option' is the pre-trained network for estimating SOC when SOH varies between 100%~95% (The range assignment is done based on section 4.3). While the testing data 'Testing35_37_Cycle_3' is phase 35-37 combined data with only the third cycle. The purpose is to remind the network to start estimating SOC from '0%', it

then follows the trend corresponding to the dataset used to train the network. Which goes like 0%-100%-0%.

Actually the network initialization should be down based on the estimation strategy, because the network was trained so that it could estimate both charging and discharging phases. When it is used to estimate only the discharging phase, the loaded dataset should only include the data corresponds to SOC decreases from 100% to 0%. For example, the dataset should be loaded as below:

```
load("first _class_3_class_option.mat")
load("Testing37_Cycle_3.mat")
testingdata= Testing37_Cycle_3;   %Cycle3 from Phase 37 data
Utest(1,:) = testingdata(1,:);    %Testing current
Utest(2,:) = testingdata(2,:);    %Testing voltage
Utest(3,:) = testingdata(3,:);    %Testing temperature
SOC= testingdata(4,:);            %Testing temperature
Y1 = SOC;
x = tonndata(Utest,true,false);
t = tonndata(Y1,true,false);
net = closeloop(net);            %Close the NARX feedback delay
loop for testing
[xs,xi,ai,ts] = preparets(net,x,{},t);
y = net(xs,xi,ai);               %Prediction SOC ouput
[sysName,netName] = gensim(net,'InputMode','Workspace',...
'OutputMode','WorkSpace','SolverMode','Discrete');
x1 = nndata2sim(x,1,1);
```

The loaded dataset ‘Testing37_Cycle_3’ is the dataset of phase 37, the third cycle. It is only consisting of PRW discharging data. In this way, the network is initialized by reminding it start from 100% SOC to 0%. Then it can be used for estimating only the discharging phase.

The code ‘net = closeloop(net)’ have to be applied to close the NARX network’s feedback loop, which means there will be no supervised SOC ground truth value, the feedback SOC value is the SOC estimated output value from the last time discrete.

After extracting all the corresponding networks and initializing them correctly, we build them in Simulink according to the previous interval assignment (Section 4.3), the schematic shown in Fig.4.4.1, Fig.4.4.2 and Fig.4.4.3 below:

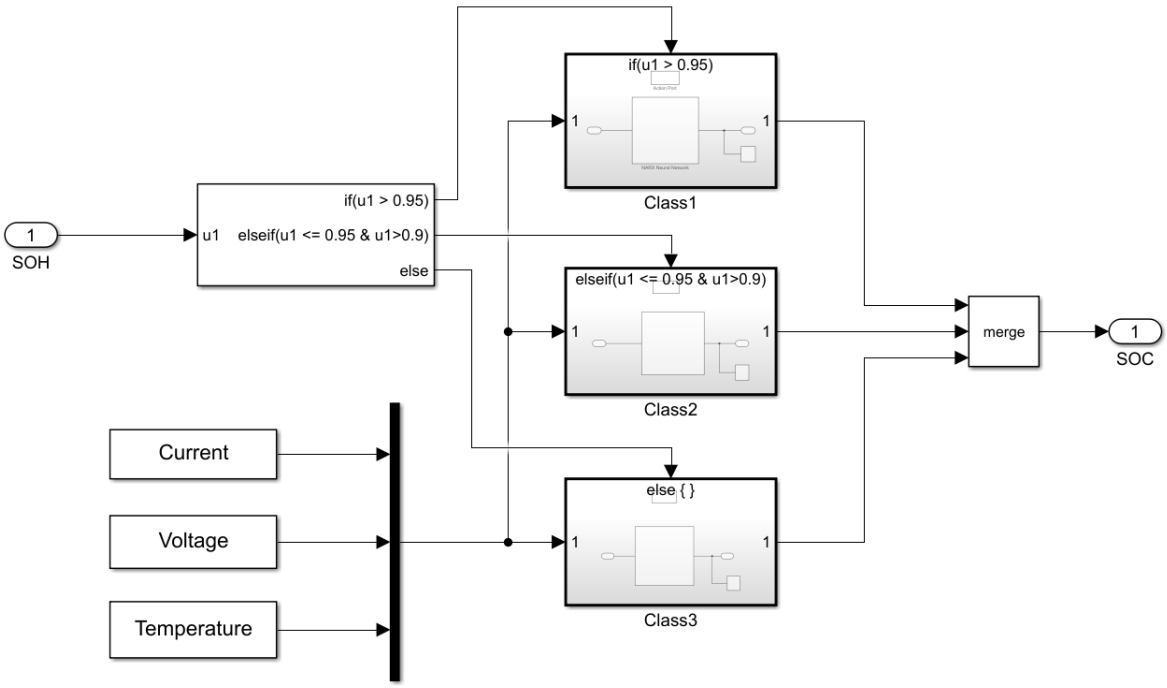


Fig.4.4.1 3-Class NARX network option

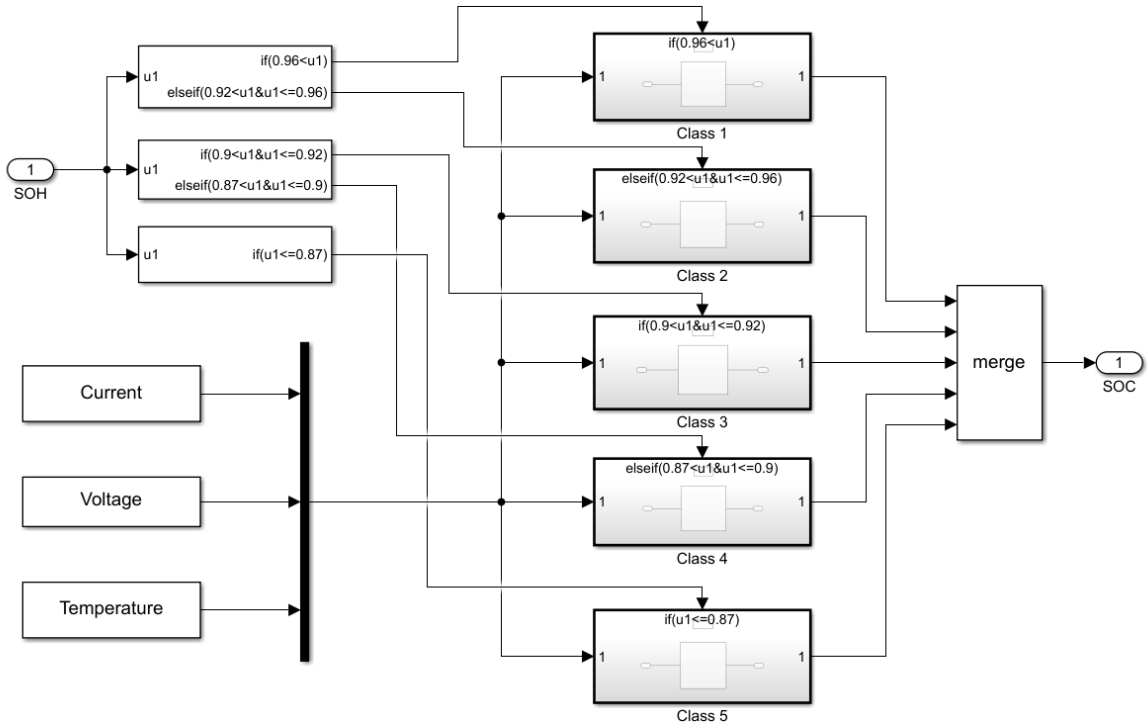


Fig.4.4.2 5-Class NARX network option

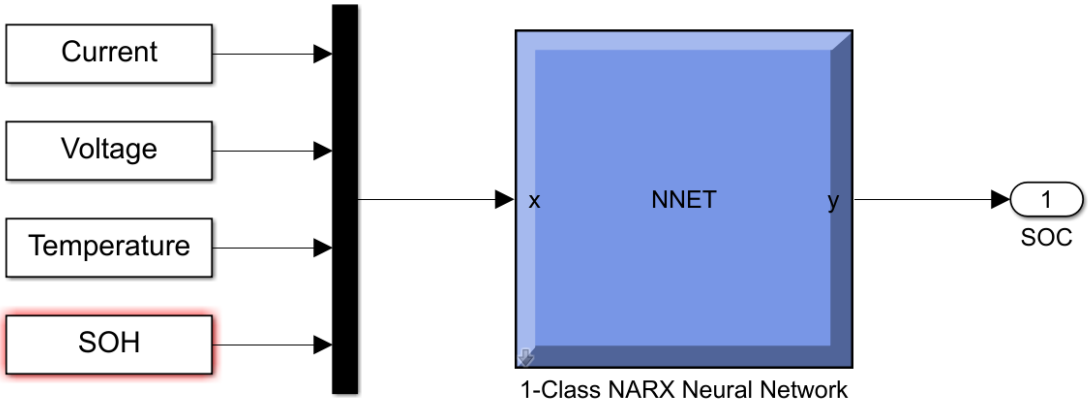


Fig.4.4.3 1-Class NARX network option

The 'if-action subsystem' block is used to trigger the corresponding NARX network, for 1-Class NARX network option, an additional SOH should be added, in this case, the SOH value should be the same as the SOH value used for training, since the real-time SOH estimation is not available yet, the ground truth SOH is used as the estimated SOH.

In this testing situation, the SOC estimation result behaves better than pure testing without any SOH ground truth, the further exploration without SOH ground truth will be shown in Chapter 5.

5 SOC-SOH combination estimation system

After getting good performance networks for each class and each option, while the other part "SOH estimation system by LSTM network" is ready. The SOC part and SOH part can be combined, based on the basic concept published by [1], the first trial of combination system is built up, then the simulation for the combined system is implemented.

5.1 Concept build up

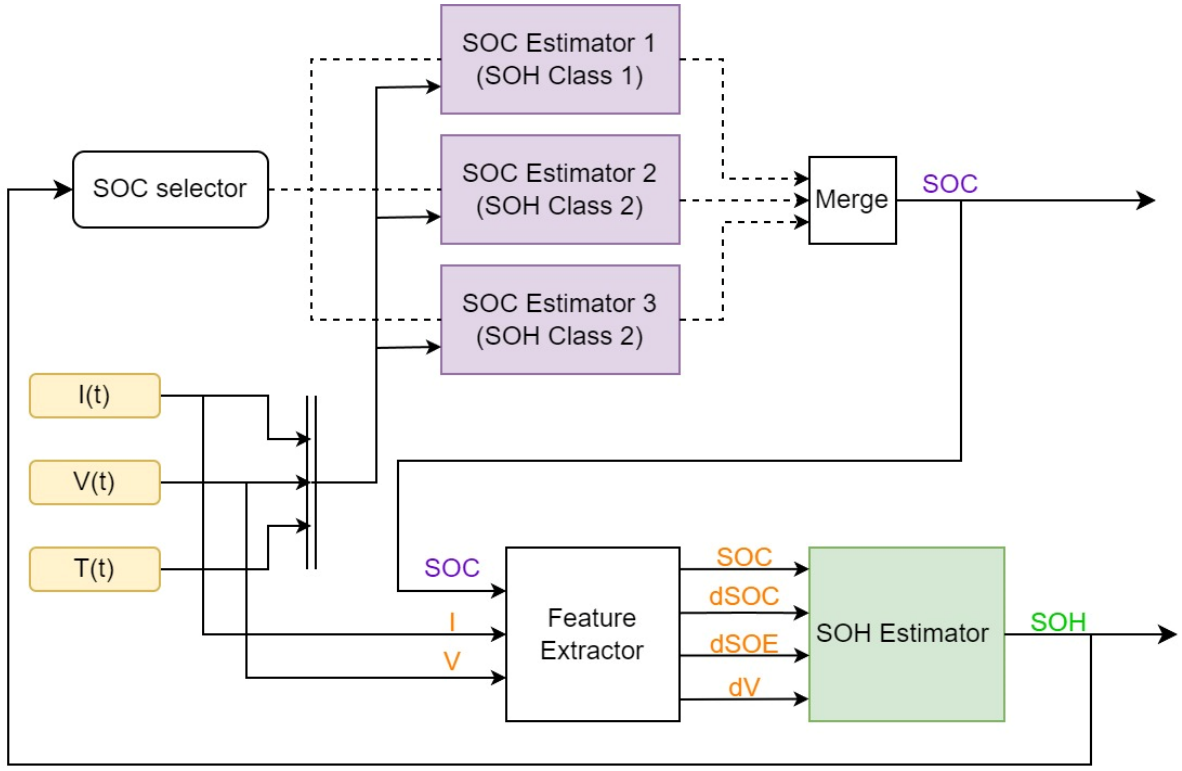


Fig.5.1.1 Concept of combination system (3-class example)

The updated concept is shown in Fig.5.1.1. At first, an initial SOH value should be given to the system, to let the estimation loop work properly. The first SOH initial value could be '1', which means the battery start at full health condition. If the system starts at other specific conditions, the precise SOH value as initial condition is needed, in order to trigger the system in the right way.

SOC selector is the 'if-action subsystem' mentioned above in section 4.4, the dedicated range used to select the corresponding SOC network can be set by the user, in the default setting, the range is the same as in section 4.3. In the further trials, the range can be changed to an appropriate value to guarantee the highest performance of the estimation system.

When the SOC selector has chosen the right SOC estimator (NARX network), at the same time current, voltage and temperature are fed into the chosen estimator, the real-time SOC will be the output. The block 'merge' is used to set output signal into one dimensional time series instead of three. The non-zero signal will be chosen and fed into 'feature extractor' block.

The ‘feature extractor’ is the main functional subsystem that extract four important features to describe the battery aging phenomenon. The subsystem includes some buffers to cumulate input data, some Matlab functions to extract the needed parameters.

The ‘feature extractor’ process the inputs[SOC, I, V] to [SOC, dSOC, dSOE, dV]. Which are the needed input features of SOH estimator (LSTM network). Inside the SOH estimator, there can be two LSTM networks, which responsible for CC phase and CV phase respectively (Since the pure charging phase is consisted of CC-CV process). Due to some performance and combined functional reason, the network which only estimate the CC phase is chosen. Further details will be discussed in the following section.

After getting the SOH estimation value, we can monitor both the SOC and SOH value, at the same time, the estimated SOH value will be fed back to the ‘SOC selector’, then the following SOH become the estimated value without any given ground truth (initial condition SOH value). The loop works properly until the end of the simulation time.

5.2 Combination simulation trial

After understanding the system concept , the Simulink models for different options are built(Fig.5.2.1, Fig.5.2.2, Fig.5.2.3):

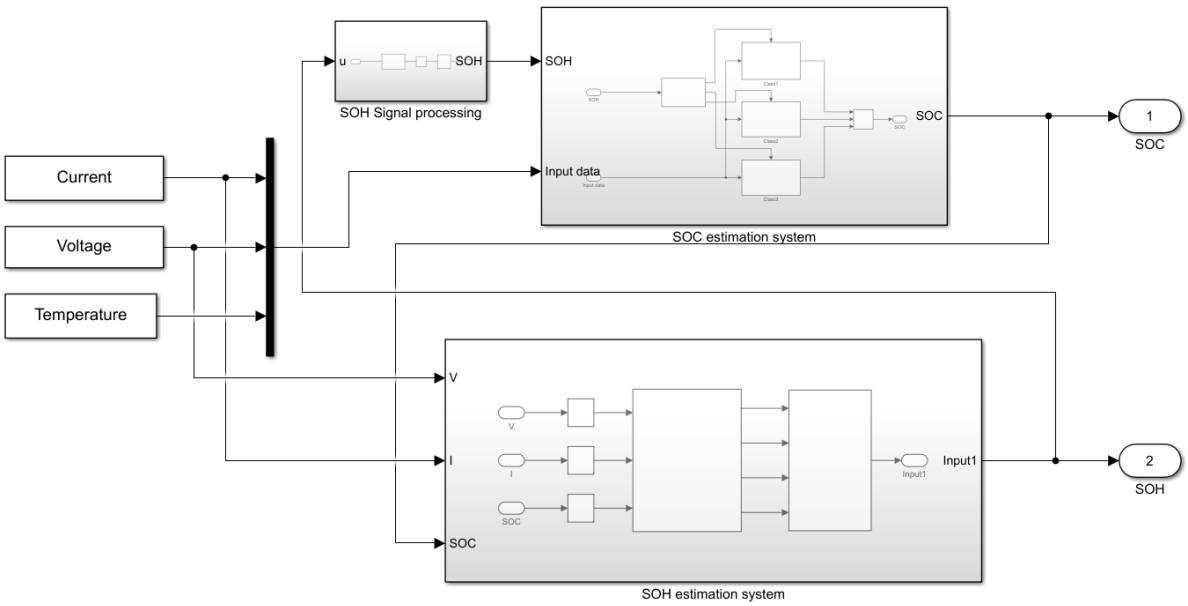


Fig.5.2.1 Combination system for 3-Class option

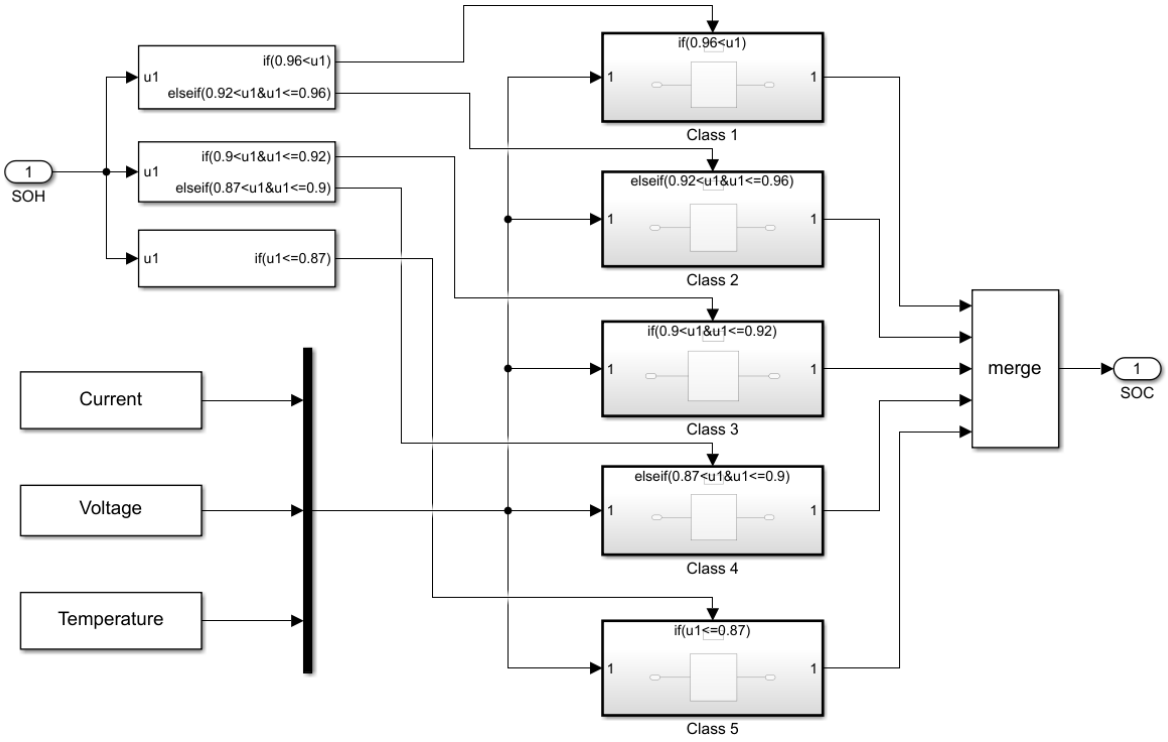


Fig.5.2.2 SOC subsystem of combination system for 5-Class option

The whole system remains nearly the same, while the SOC estimation system being more complicated, more if-action subsystems are used to cover 100%-80% SOH value

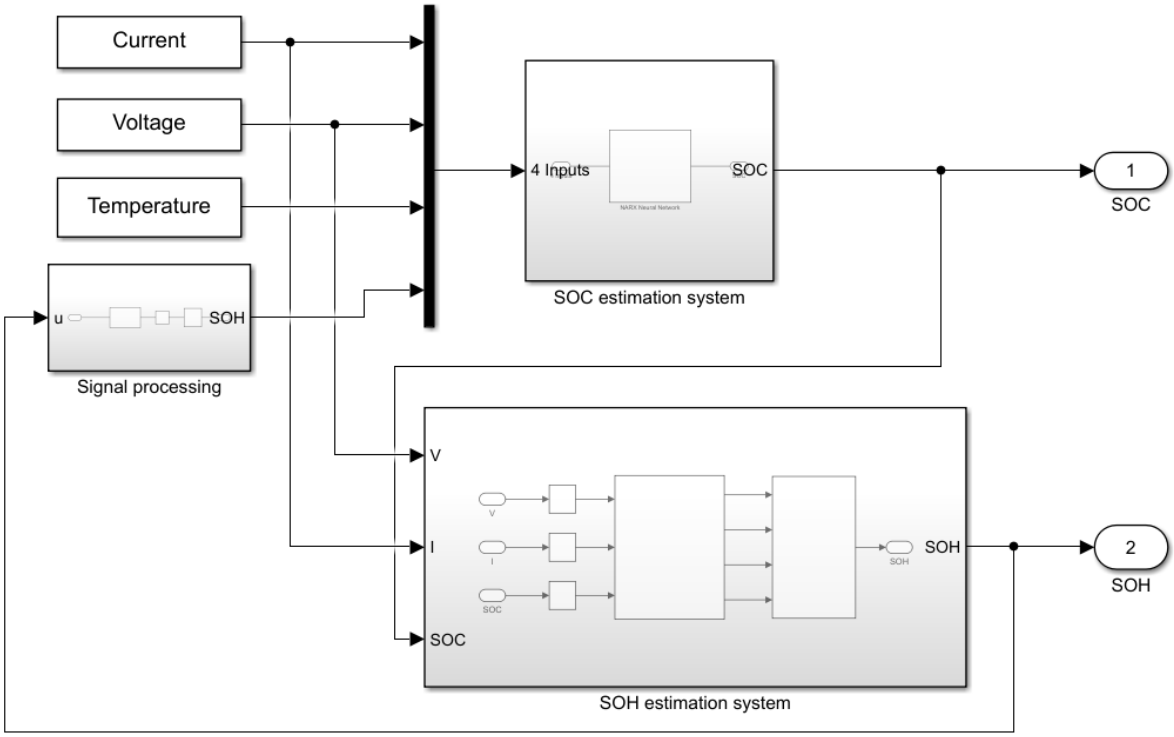


Fig.5.2.3 Combination system for 1-Class option

A SOH signal processing subsystem is needed when the output of SOH estimation system be the input of SOC estimation system, in order to get the right SOH signal value. (Shown in Fig.5.2.4)

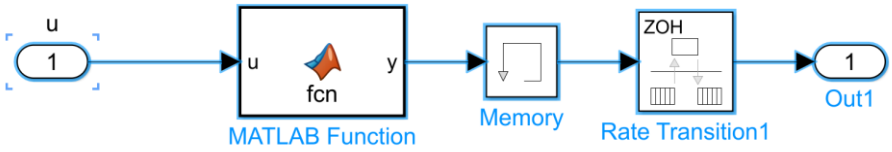


Fig.5.2.4 Details of signal processing subsystem

The Matlab function block changes the SOH value to 1 when its zero at the beginning. Because the buffers in the SOH estimation systems need to cumulate the first 120 seconds to output the first group of features. These zero values will lead to wrong SOC network selection. (Code shown below).

```
function y = fcn(u)
y=double(u);
if u==0
   y=double(1);
end
```

There should also be a 'memory' block to give an initial value '1' to active SOC estimation system first, and then the SOC-SOH loop will work properly (Mentioned in section 5.1).

'Rate transition' block is used to align if-act subsystem function and SOH output sampling rate, which is '1Hz'.

Inside the SOH estimation system, the schematic shows like Fig.5.2.5, inside the LSTM network, there are two networks which estimate CC part and CV part respectively. In the actual simulation, there needs to be a judgment condition here to make the network switch from CC to CV, but in fact this condition depends on the actual application conditions (charging control of the charging pile), so we actually don't know when to switch from CC to CV. In addition, it has been proved by experiments that there is a possibility of a step SOH value in the conversion process of CC and CV, and we cannot guarantee that the last value estimated by the CC network is the same as the first value estimated by the CV network.

Based on this situation, we decided to use only the CC part as the estimation network for

SOH (Shown in Fig.5.2.5).

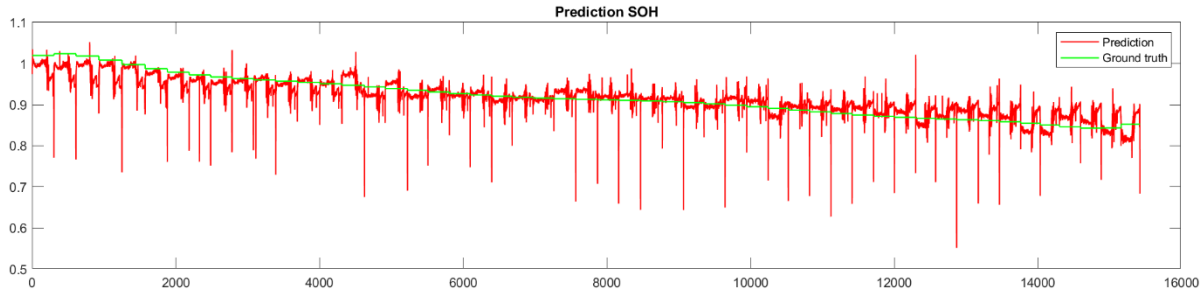


Fig.5.2.5 SOH step variation during prediction

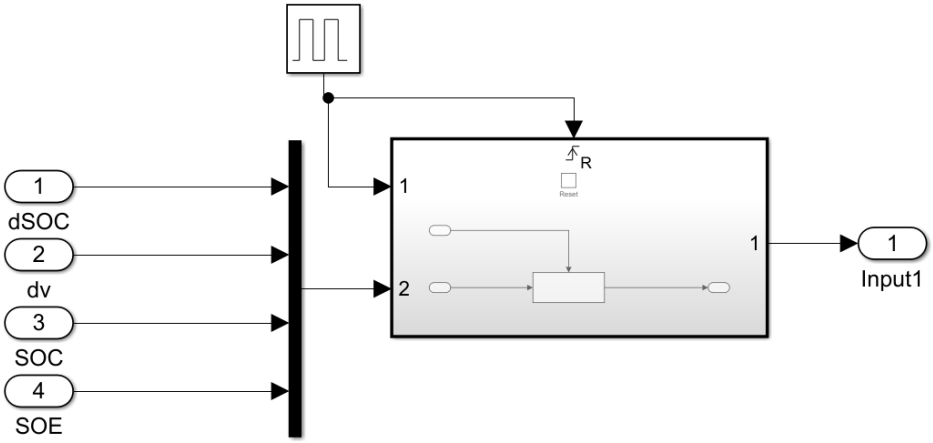


Fig.5.2.6 Subsystem with only CC part

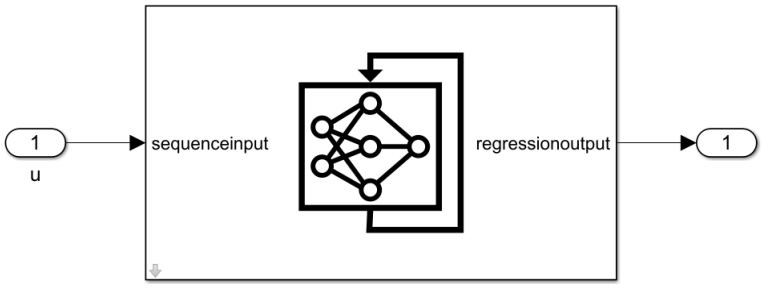


Fig.5.2.7 CC network inside the subsystem

As shown in Fig.5.2.6 and Fig.5.2.7, the SOH network is consist of only CC part. When both SOC estimation system and SOH estimation system are ready, the simulation can be started. The system is tested by two different testing data, which are Phase 1-Cycle 3 and Phase 1-Cycle 38, the purpose of choosing two different cycle is to test if it can work properly under different SOH conditions. For cycle 3, the initial SOH is equal to 1.0191. For cycle 38, the initial SOH is equal to 0.8817. Both of these SOH values are taken from the SOH ground truth, in order to trigger the right SOC network at the beginning of the simulation.

Results are shown in Fig.5.2.8 and Fig.5.2.9:

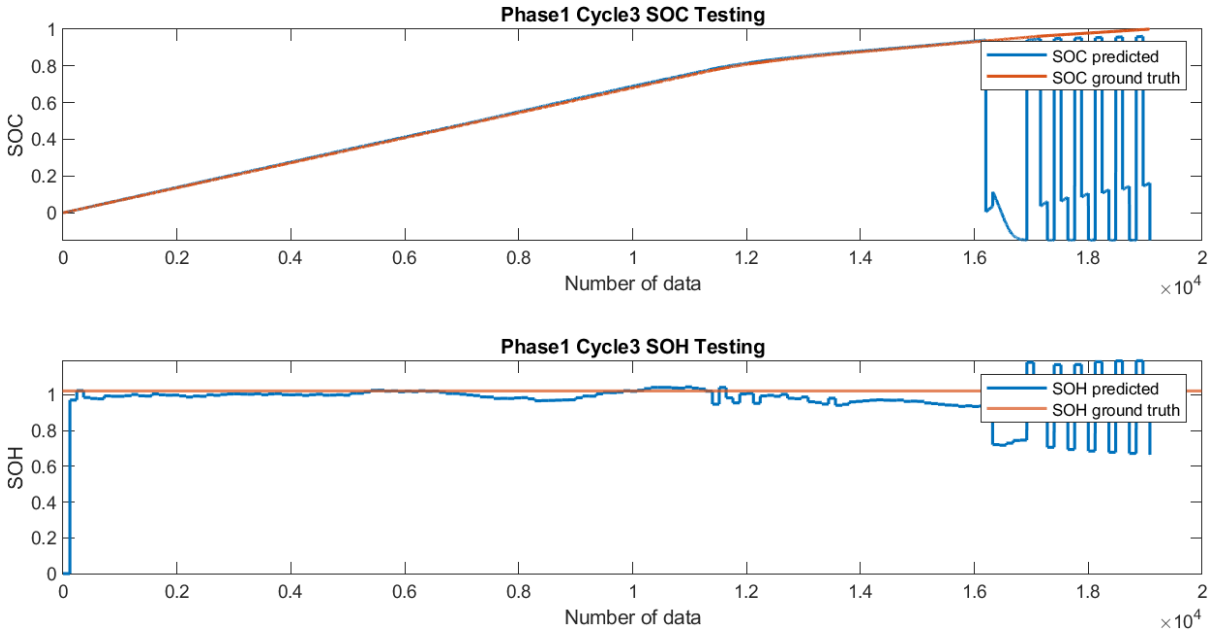


Fig.5.2.8 Phase1 Cycle 3 simulation results

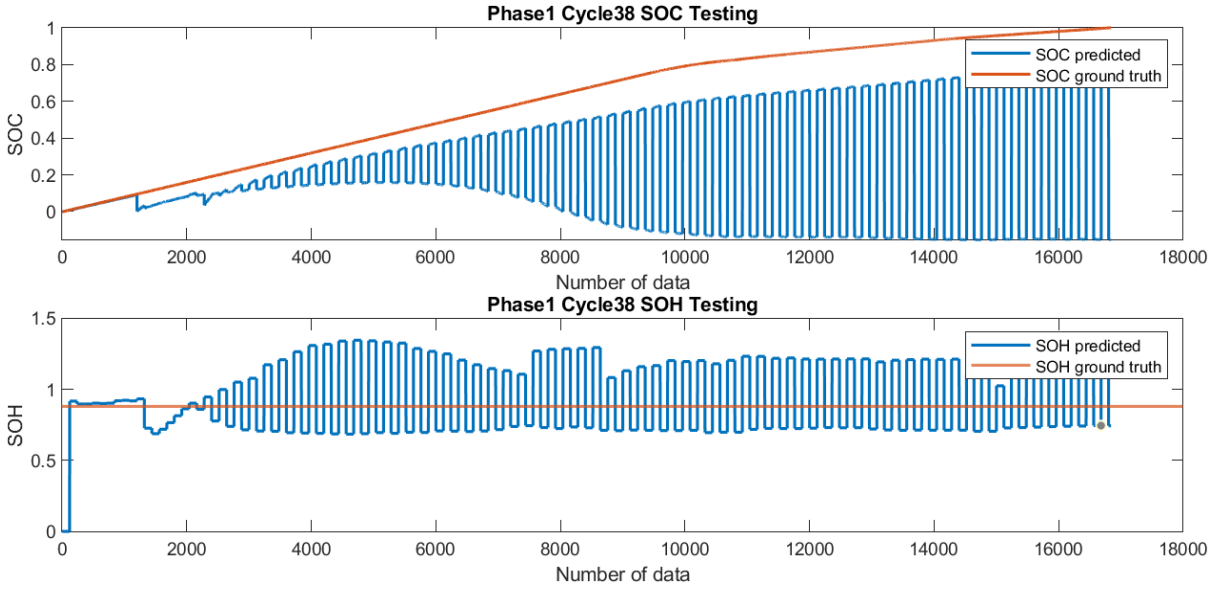


Fig.5.2.3 Phase1 Cycle 38 simulation results

From these result we can see, the stability of SOH prediction process is strongly dependent on the prediction of SOC. As shown in Fig.5.2.8, before about 16000 seconds (the time before shifting to CV part), both SOC and SOH predictions are stable and precise, once the charging phase exceed CC part, SOH estimation network becomes oscillating since this SOH network is only suitable for CC. In this situation, we need to add a judgment block in order to let the SOH network stop estimating when it nearly reach the CV zone. (For example, SOH prediction stops when SOC>0.7).

The other problem should be solved is shown in Fig.5.2.9. We can see both SOC and SOH becomes oscillating at about 1000 seconds, and the SOC value restart from about 0 at this point. This is mainly caused by the characteristics of SOC estimation network (NARX network). Since NARX has timeseries memory functions, in our application , it is set to start from 0 during charging phase. When the real-time estimated SOH value exceeds the assigned range of each SOC networks, the SOC estimation network will be changed from the current one to the other one (such as from Class 1 to Class 2). In this case, since the newly selected network has no previous timeseries memory, it re-estimates from 0. Although the input does not match the current estimate value, NARX has a tendency to correct toward the correct value, but it still has a huge impact on accuracy. Thus causes the poor performance of the whole system. Some corresponding solutions are mandatory to solve the problem described.

5.3 Combination method update

Due to the situation described in section 5.2, based on the characteristics of two parts during the first combination trial, the update estimation strategy is developed shown as Fig.5.3.1:

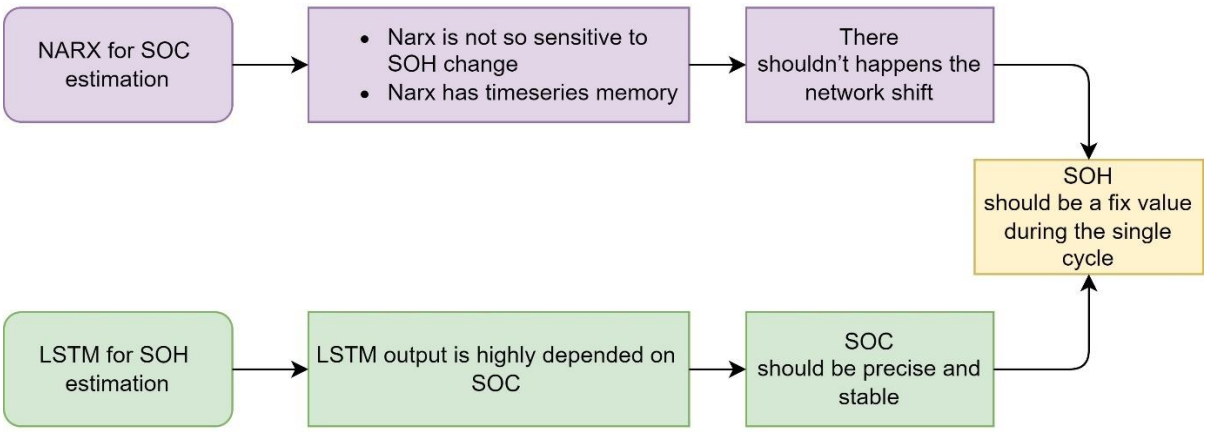


Fig.5.3.1 Estimation strategy

Based on the ground truth SOH value (shown in Fig.4.3.1), the SOH ground truth value only decreases a very small value(~0.5%) between two cycle. And we have approximated the SOH values of all phases in the same cycle to be equal. So the SOH value in a cycle, no matter which phase is applied, we can approximate it as constant. In this condition, the dynamic SOH output during a single cycle (synchronized with the output frequency of the SOC) is not needed, not only for stabilize the SOC estimation network, but also for obtaining a precise

SOH estimation value (since SOH estimating process needs buffers to accumulative data). However, the change of SOC is a very fast process (compared to SOH), and SOC needs to be used as an input of the SOH estimation network, so SOC is still a dynamic output value, and it needs to be very accurate to prevent the SOH estimation network from doing wrong judgment.

The new strategy is done by obtaining a fix SOH value at the end of each cycle, this SOH value is the trigger SOH value which used for SOC network selector in the next estimation cycle. At beginning an initial SOH ground truth value should be given in order to start the simulation cycle, the default value is equal to 1, which means it's a full health battery at the beginning. If we want the cycle start at the other SOH value, a precise SOH value should be measured, in order to make the estimation system running in a proper way (Shown in Fig.5.3.2).

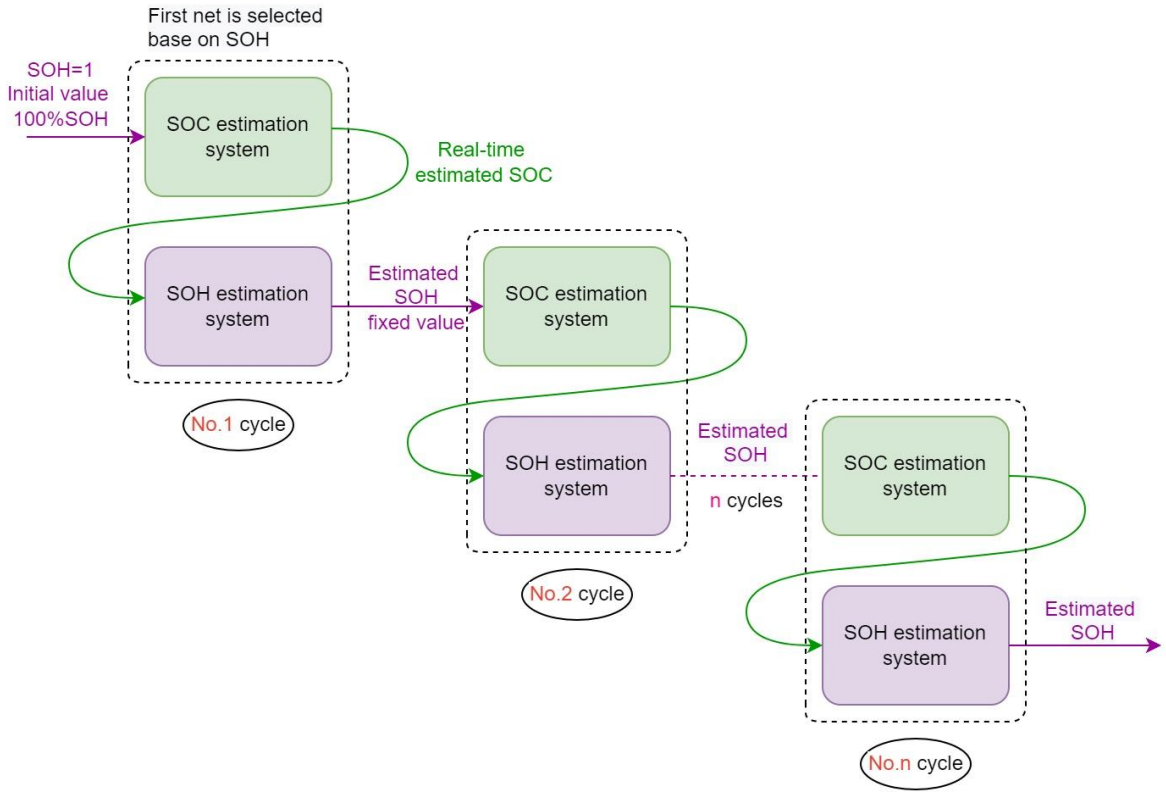


Fig.5.3.2 Charging phase estimation flow

The code for implementing multiple cycle simulations is shown as follows:

```
%% Charging
for i=1:length(meas)
    Current_og{i, 1}=meas{i, 1}.Current;
    Voltage_og{i, 1}=meas{i, 1}.Voltage(:,3);
    Temperature_og{i, 1}=meas{i, 1}.Temperature(:,3);
    simtime_og{i, 1}=length(meas{i, 1}.Current);
     SOC_GT_og{i,1}=SOC{i, 1};
%Extract Current, Voltage , Temperature, simulation time and SOC
ground truth of each cycle from the original data.
end
%% One charging cycle each time
SOH_predict=1;  % Give an initial SOH data ground truth, default
value =1
i = 1
for j=1:52      % The number of simulation cycle can be modified
   Current=timeseries(Current_og{i, 1});
   Voltage=timeseries(Voltage_og{i, 1});
   Temperature=timeseries(Temperature_og{i, 1});
   SOC_GT=timeseries(SOC_GT_og{i, 1});
   simtime=cell2mat(simtime_og(i)); % Assign the original data to
corresponding cycles
   sim('Combined_net_3Class')  % Modify the model name:1,3,5Class
   SOH_predict=double(ans.SOHout(simtime)); % Assign the
predicted SOH value to the next cycle
   SOH_saveresult(i,1)=SOH_predict;         % Save predicted SOH
value of each cycle
   error(i,1)=SOH_predict-SOH_p01(i,1);
   SOC_abserror{i, 1}=ans.SOC_abserror;     % Compute estimation
errors
   i=i+1
end
```

A filtering operation is required for the SOH estimation network to estimate the PRW discharge phase, since the data has strong oscillation behavior (shown in Fig.4.1.7), it needs to be filtered to find the corresponding features .The filter used for this step is 'Butterworth' filter (same as section 3.2).

Since Butterworth needs to filter the global data, in order to train a very high-performance SOH estimation network, but in our practical application, the cycle should be split and considered individually, rather than the global overall data (pure testing condition without

ground truth) , if we apply each cycle through Butterworth filtering, the features obtained from each cycle are different, which will cause the performance of the SOH estimation network to be particularly poor. Because we cannot guarantee that the estimation methods of the two parts during the PRW discharging phase in the same way. So for the PRW discharging phase, we use the SOH estimated from the charging phase as the ground truth, for estimating the discharging SOC. In order to obtain stable and accurate SOC estimation results (Estimation flow shown in Fig.5.3.3).

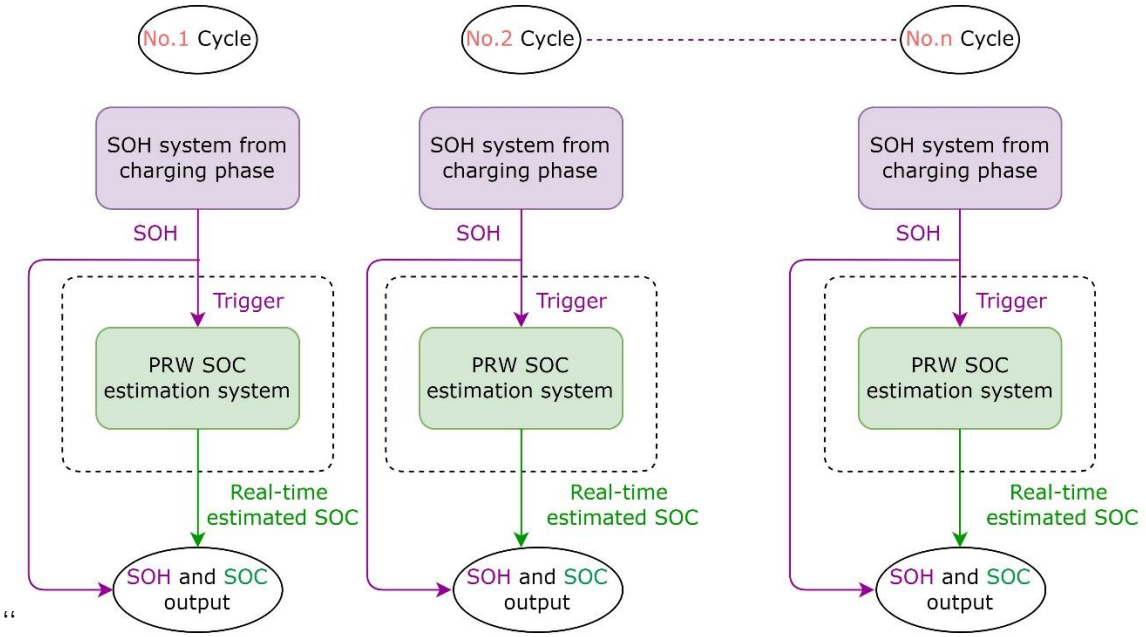


Fig.5.3.3 PRW discharging estimation flow

5.4 Simulink update

After knowing the updated system clearly, the simulation can be started, the updated Simulink model shown in Fig.5.4.1:

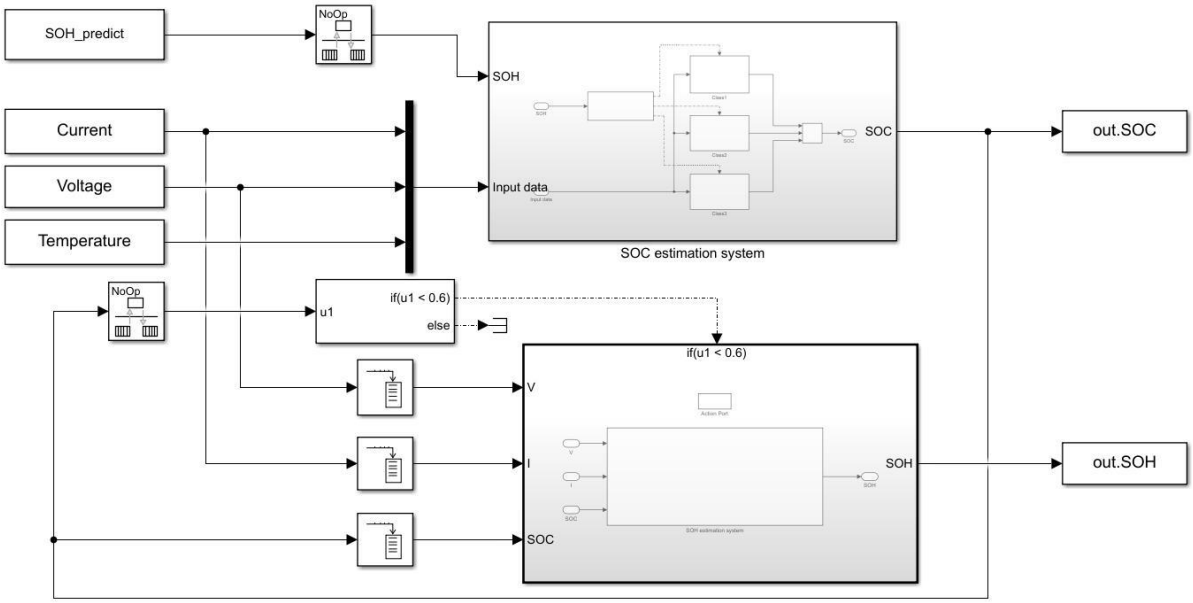


Fig.5.4.1 The updated combined system (3-Class option)

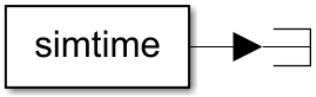


Fig.5.4.2 Introduce simulation time into Simulink

The above figure shows the 3-Class option of updated system. Since the simulation time of each cycle is not constant, the stop time which equals to simulation time should be introduce into Simulink models, in order to let the stop time change with the changing of cycles (Shown in Fig.5.3.4).

For the 5-Class option, only the SOC estimation subsystem is replaced by the 5-Class NARX network subsystems (Same as Fig.4.4.2). The other parts remain the same. For the 1-Class option, the system is shown in Fig.5.3.5.

For the polarized random walk discharging estimation, the model is built based on the concept of Fig.5.3.3. Predicted SOH value is taken from the SOH estimation output of charging phase, and be input as the ground truth SOH in order to trigger the right SOC estimation networks (In Fig.5.3.3, SOH ground truth is 'GT'). The rest part of the model is nearly the same with Fig.5.4.1, but without the SOH estimation system. It can be thought as the continue estimation for discharging when the charging phase is finished (The SOH value during the current phase is estimated).

In this case, we can decouple the charging and discharging phases, so that the charging and discharging process does not have to be continuous. In the real application, it is like the

vehicle has been fully charged, but it has been connected to the charging pile and has not moved. When the user wants to drive, based on the estimated SOH during the charging period, the SOC state during the discharging period can be accurately estimated, and the SOH value (fixed value) will also be monitored on the screen. This SOH value will not change until the next charging phase.

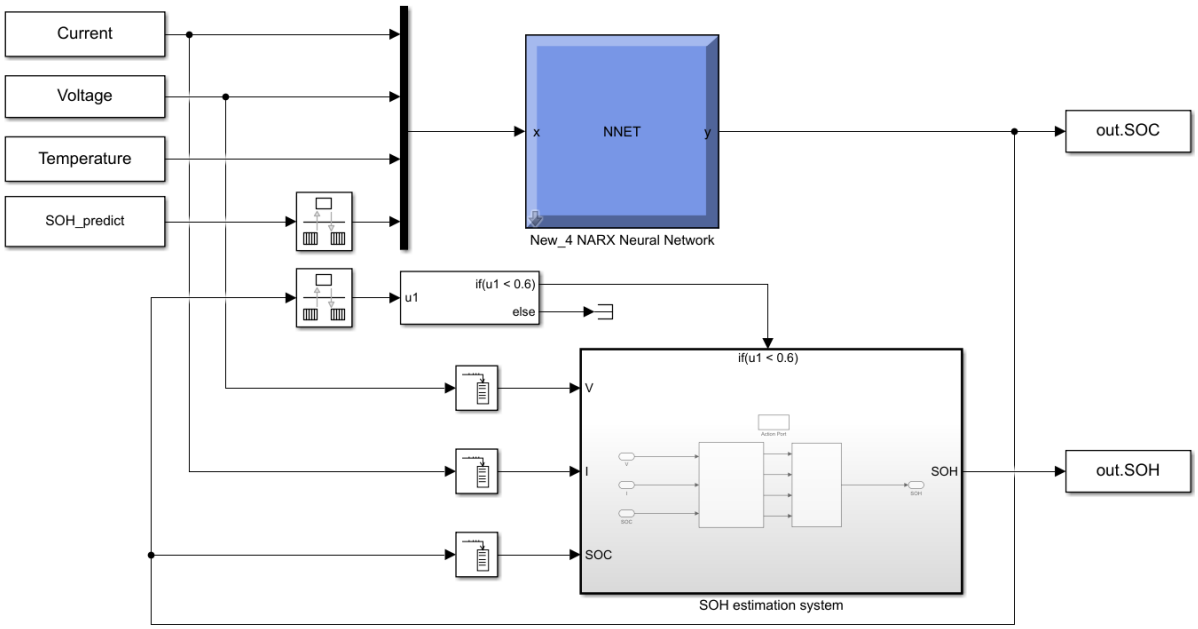


Fig.5.4.3 The updated combined system (1-Class option)

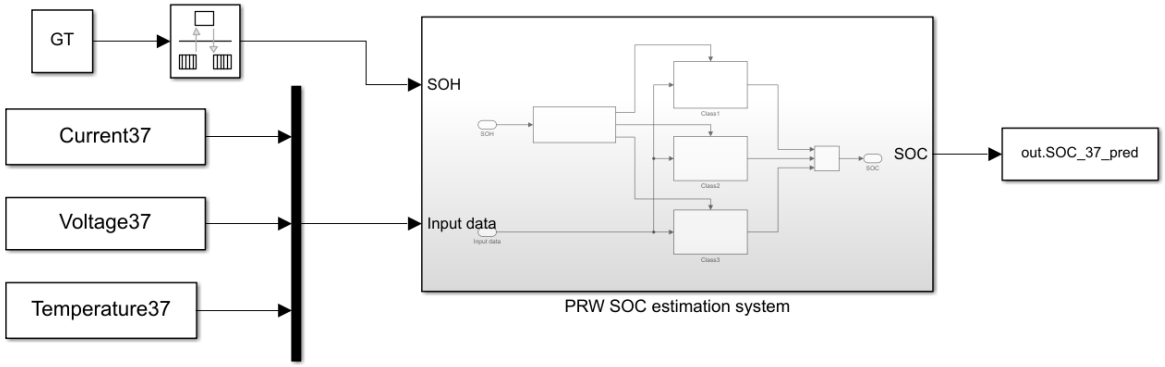
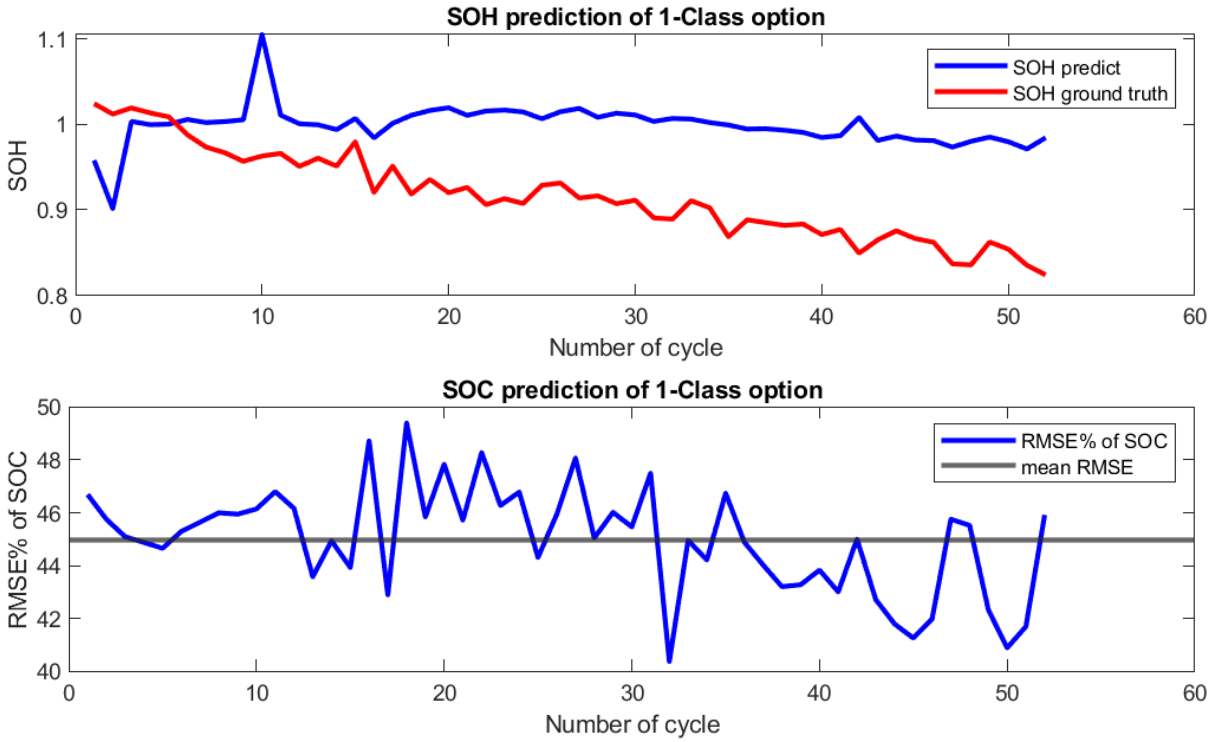


Fig.5.4.4 SOC estimation system for PRW discharging

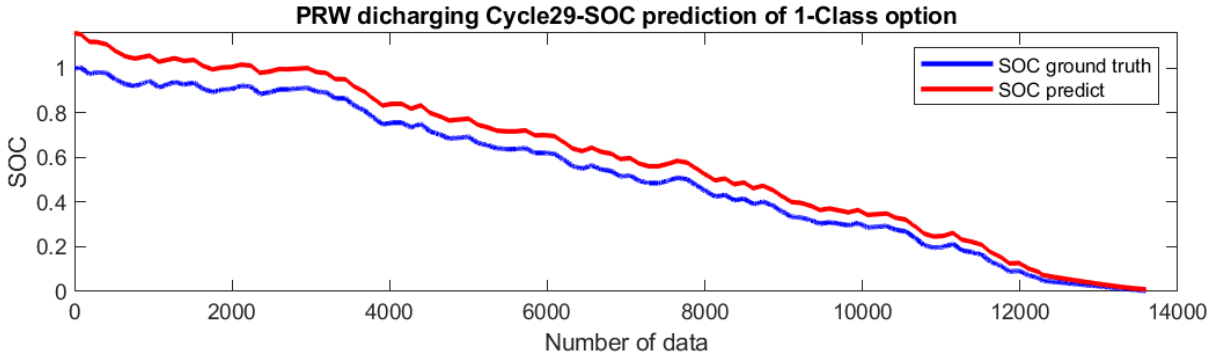
5.5 Result comparison

The simulation of both charging and PRW-discharging of three options are done, SOC and SOH prediction results of all the cycles are shown below, while for the PRW-discharging prediction, a random cycle (here cycle 29 is selected, the selected cycle can be changed) is displayed in order to have a rough idea of how the discharge phase behaves.

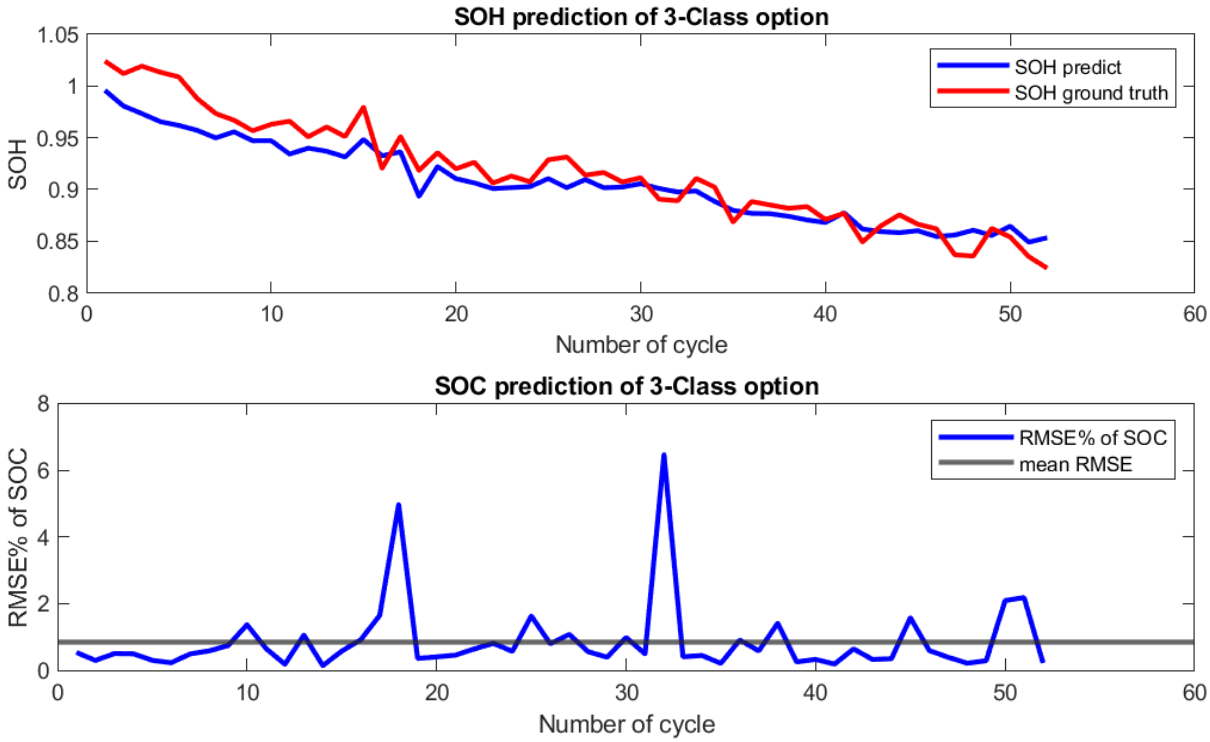
SOC and SOH prediction results of 1-Class option (Charging)



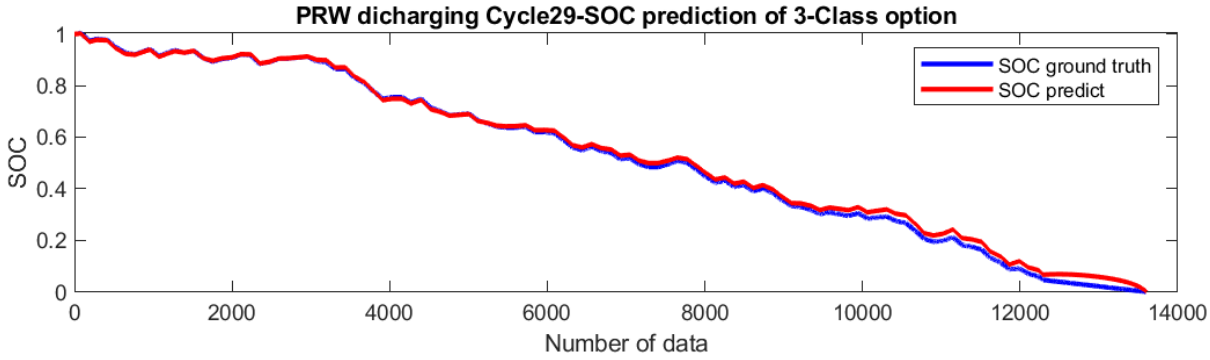
Cycle 29-SOC prediction results of 1-Class option (PRW discharging)



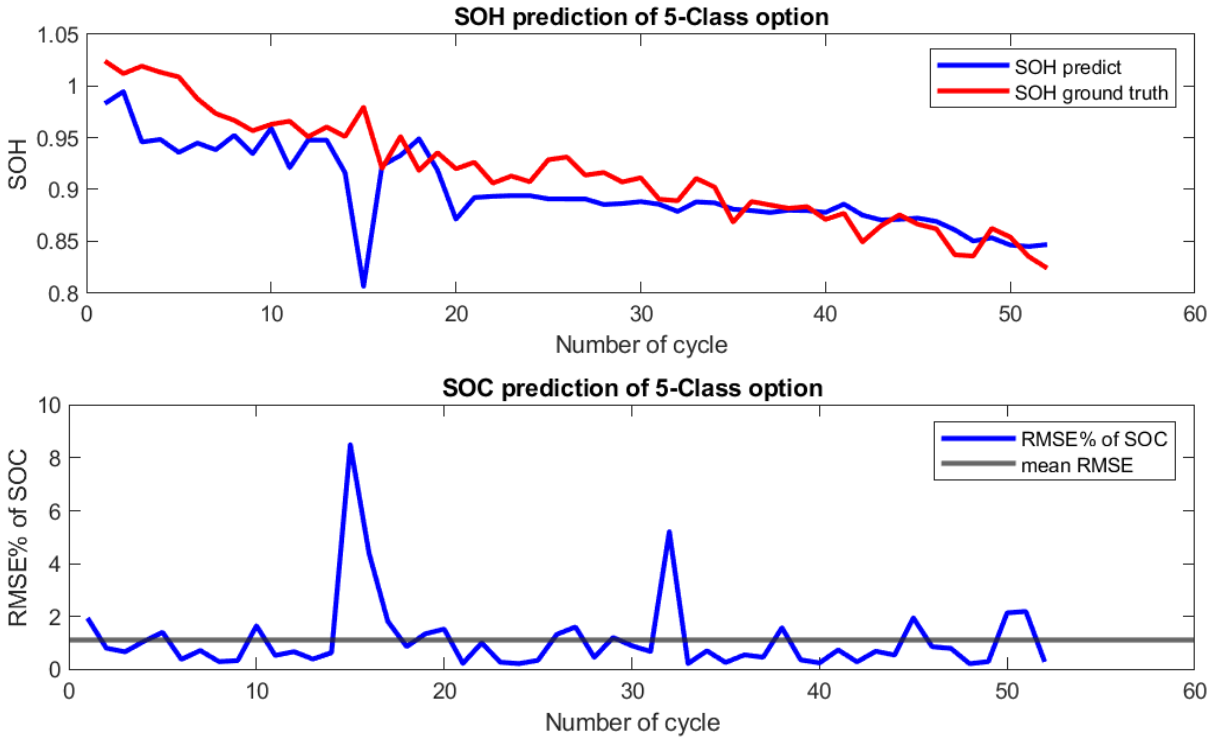
SOC and SOH prediction results of 3-Class option (Charging)



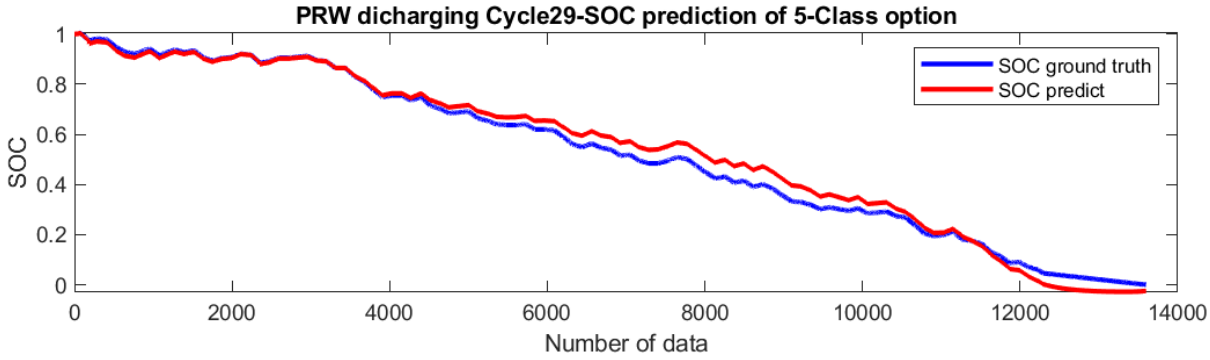
Cycle 29-SOC prediction results of 3-Class option (PRW discharging)



SOC and SOH prediction results of 5-Class option (Charging)



Cycle 29-SOC prediction results of 5-Class option (PRW discharging)



In fact the combined system is composed of multiple parts (only for the case of 3-Class and 5-Class).

The performance of each individual sub-network also has an impact on the global performance because each network has an opportunity to estimate within its own SOH working range. Another factor is the SOC network selector assigned to each sub-network based on the input SOH value. A final influencing factor is the choice of cutoff value for SOH on the SOC estimation interval. Since the estimate of SOH is also fluctuating, but we chose the last cut-off estimate as the output, so the output SOH is obtained as:

$$SOH_{\ OUPUT} = SOH_{estimate}(SOC_{cut-off})$$

The cutoff value of SOC also has a great influence on the final global result.

For example, in 3-Class option, the final interval selection changes from Fig.5.5.1 to Fig.5.5.3, and the corresponding result also changes:

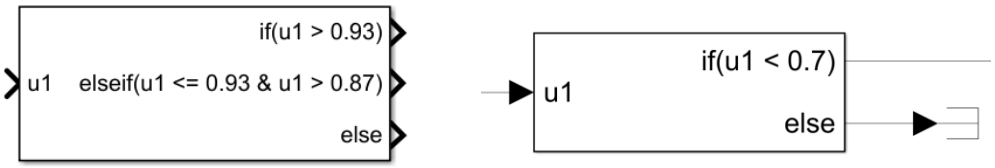


Fig.5.5.1 Original conditional judgment from 3-Class NARX combination option

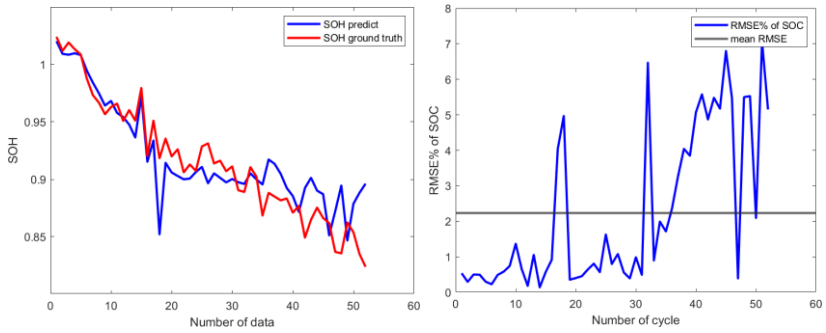


Fig.5.5.1 Original conditional judgment result SOH RMSE%=2.41% SOC mean_RMSE%=2.23%

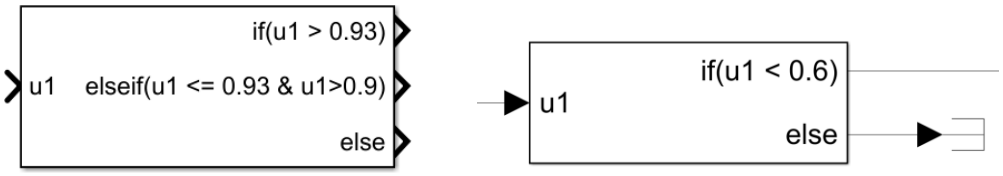


Fig.5.5.3 Updated conditional judgment from 3-Class NARX combination option

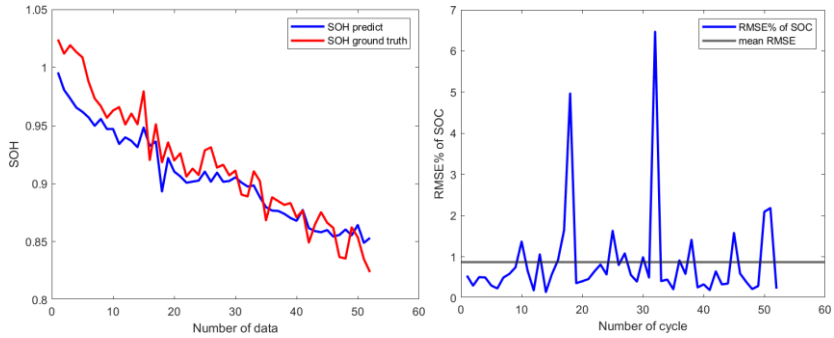


Fig.5.5.4 Updated conditional judgment result SOH RMSE%=1.99% SOC mean_RMSE%=0.87%

In the end, the comparison of both simulation speed and performance is implemented. Shown in Fig.5.5.5 and Fig.5.5.6:

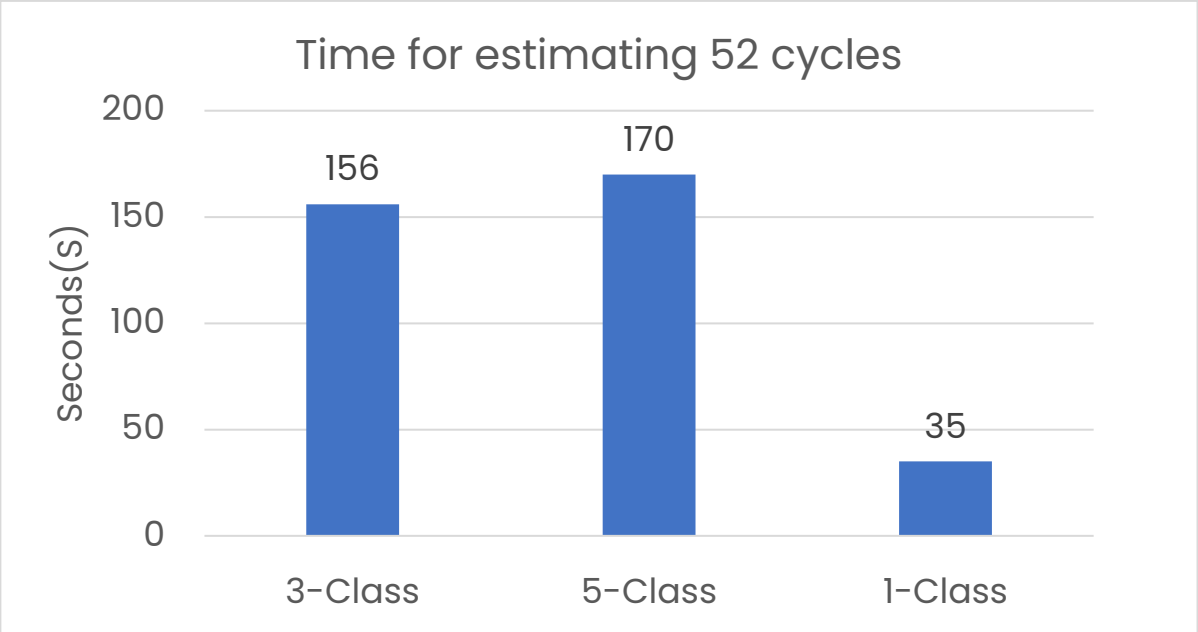


Fig.5.5.5 Estimation time comparison of three options

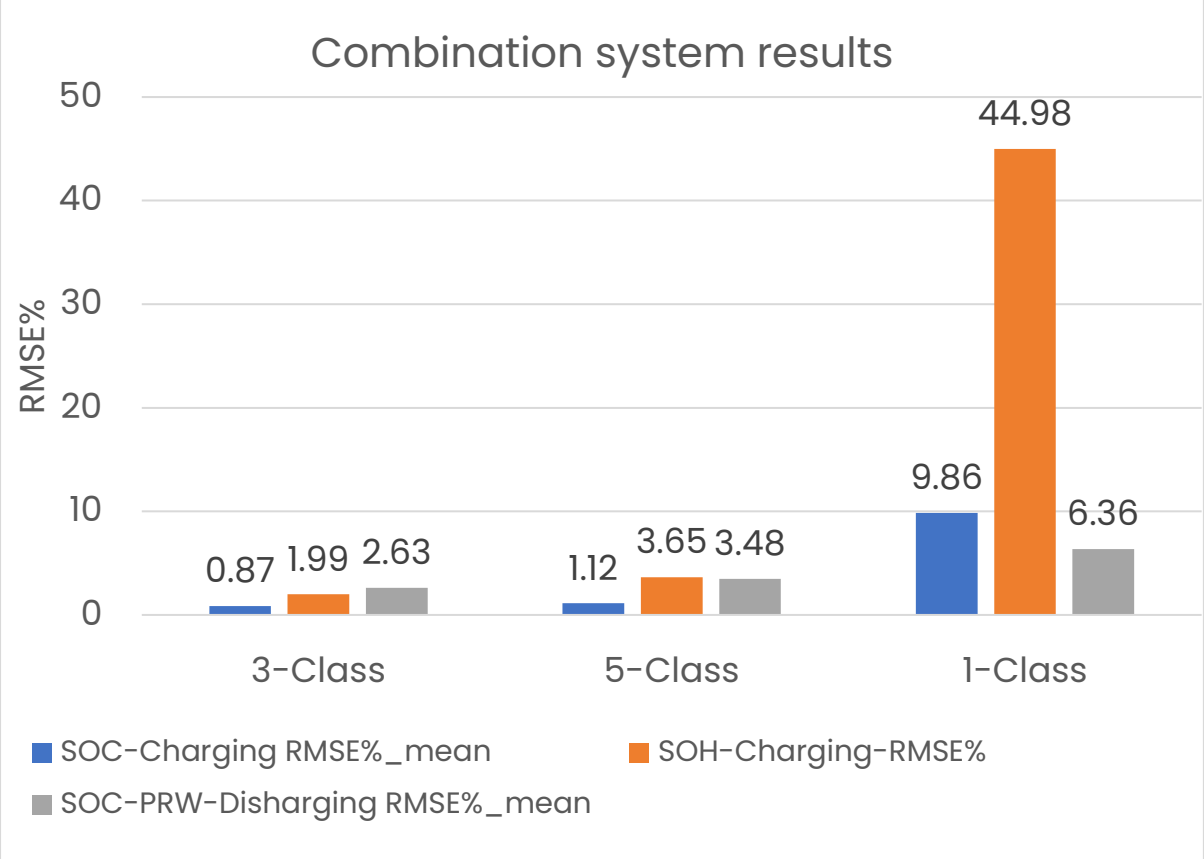


Fig.5.5.6 Estimation performance comparison of three options

6 Conclusion

The central idea of this thesis is to develop a lightweight and accurate network for estimating the SOC of electric vehicle batteries, considering real application scenario and real-time estimation functionality. First, the SOC estimation method of the network is analyzed one by one, and the latest neural network method is adopted.

Then three different kinds of neural networks are introduced and then trained separately. According to the performance and characteristics of the network itself, they are optimized respectively and then compared. The SOC estimation which performs best is finally selected for the next specified dataset. NARX has the best performance in this step and the corresponding mean RMSE% of all the testing data for the specific dataset is 0.29%.

After the best network and optimal structure are determined, a new and more complex dataset is introduced, which can also be used for training the SOH network (LSTM), since it contains a series of features that can describe the aging phenomenon of the battery. After picking out the data most suitable for training SOC estimation, a training process similar to the previous one is implemented. In this step, the classification of the SOC estimation network for the SOH interval is also arranged at the end in order to achieve the best network performance to deal with the complex nonlinear aging characteristics. There are three classification options, among which is RMSE% ranging from 1.40% to 3.93%.

Finally, the author's part (SOC estimation network) and the co-author's part (SOH estimation network) are put together. As a result, a combined estimation system based on the classification concept is built and then transferred to Simulink for construction. After overcoming the respective matching problems of the two parts, the combined system can simultaneously estimate SOC and SOH, two important parameters of BMS. 3-Class NARX combined with CC part LSTM has the best performance, SOC estimation RMSE% during charging phase is 87%, SOH estimation RMSE% during charging phase is 1.99%, SOC estimation RMSE% during PRW discharging phase is 2.63%.

During this research process, some shortcomings and problems were also found, and there is potential for improvement in further follow-up research.

Possible improvement

Temperature range:

In SOC estimation networks performance comparison (Chapter 3), the temperature range of this dataset varies from -20° to 25°, it covers a wide range of temperatures in order to improve the networks' robustness. But such type of dataset does not show any aging features or phenomena, so it's not enough for extracting the needed features. In other words, it cannot be used for training the LSTM network for SOH estimation. While in SOC estimation considering SOH in Chapter 4, another dataset is used instead of the previous one. The new dataset consists of a large number of different cycles which has very detailed description of some characteristic changes due to oxidation. It can not only be used for SOC network training, but also for SOH networks training. But the temperature variation of this specific dataset is limited. The temperature varies from about 17° to 28°, even if it has been considered the temperature variation corresponding to other parameters, it does not cover a lower temperature range like zero or the minus temperature. For further exploration, in order to achieve more robust performance of the SOC-SOH combined system and cover more corner cases, the lower temperature related data augmentation of this specific dataset is needed.

Data sampling accuracy:

Comparing the dataset used for Chapter 3 and that for Chapter 4, we can see the difference in accuracy between the two datasets. The dataset used for Chapter 3 has the overall accuracy of about 0.1% (from Table 1), while for the dataset used for Chapter 4, the accuracy is not mentioned. But from the dataset, such as SOH ground truth (Fig.4.3.1), the SOH is not linearly decreased, it increased a little bit in some specific cycles. This can cause a little inaccuracy when assigning the SOH classes for SOC estimation network. And the measurement bias of other parameters will also cause training fluctuations.

SOC-SOH combination estimation for PRW discharging phase:

Due to limited exploration time, the author of the SOC section and the author of the SOH section did not make further upgrades to the combined system for both SOC and SOH estimates for the PRW discharging phase. (In section 5.3, the system is done by feeding the

SOH value estimated from charging phase, and only the SOC estimation network is working during the PRW discharging phase, because of the cycle matching problem of two networks due to filter application).

Hardware testing

These combined system models can be deployed and transferred into the hardware platform. The hardware in the loop (HIL) architecture can be connected to the original battery module, or the other module made by the same battery cells. The further HIL exploration can be done, in order to see the real application ability and performance. [22]

Reference

[1] Angelo Bonfitto. “A Method for the Combined Estimation of Battery State of Charge and State of Health Based on Artificial Neural Networks”, energies-13-02548,2020.

[2] ZHONGXIAOI L, LI Z, ZHANG J B. Alternate adaptive extended Kalman filter and ampere-hour counting method to estimate the state of charge[C]//2018 IEEE International Power Electronics and Application Conference and Exposition (PEAC). November 4- 7, 2018, Shenzhen, China. IEEE, 2018: 1-4

[3] Fu Shiyi, Lv Taolin:“Overview of SOC estimation methods for lithium-ion batteries for electric vehicle” , Energy Storage Science and Technology 2021,May.

[4] https://en.wikipedia.org/wiki/Open-circuit_voltage

[5] Daehyun Kim,Keunhwi Koo,Jae Jin Jeong :“Second-Order Discrete-Time Sliding Mode Observer for State of Charge Determination Based on a Dynamic Resistance Li-Ion Battery Model”.

[6] Yang J, Wang T, Du C Y, et al. Overview of the modeling of lithium-ion batteries[J]. Energy Storage Science and Technology, 2019, 8(1): 58-64

[7] Slides of “Electrical drives for e-mobility” by prof.Bojoi, Politecnico di Torino.

[8] https://en.wikipedia.org/wiki/Kalman_filter

[9]https://it.mathworks.com/videos/understanding-kalman-filters-part-4-optimal-state-estimator-algorithm--1493129749201.html

[10] https://www.kalmanfilter.net/multiSummary.html

[11] https://blog.csdn.net/fengbingchun/article/details/50274471

[12] Zhenzhu Meng,Yating Hu, Christophe Ancey. “Using a Data Driven Approach to Predict Waves Generated by Gravity Driven Mass Flows”. February 2020,Water 12(2)

[13] Domenico Carlucci, “Battery state of health and state of charge estimation: comparison between classical and machine learning techniques”, Politecnico di Torino.

[14] B. Yegnanarayana, Artificial neural networks. PHI Learning Pvt. Ltd., 2009.

[15] W. He, N. Williard, C. Chen, and M. Pecht, “State of charge estimation for Li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation,” Int. J. Electr. Power Energy Syst., vol. 62, pp. 783–791, 2014.

[16] W. Zhang et al. “Deep learning-based prognostic approach for lithium-ion batteries with adaptive time-series prediction and on-line validation”/ Measurement 164 (2020) 108052.

[17] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] https://it.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html

[19] Carlos Vidal, Pawel Malysz, Mina Naguib, Ali Emadi, Phillip J. Kollmeyer, “Estimating battery state of charge using recurrent and non-recurrent neural networks,” Journal of Energy Storage, 2021 (see https://www.sciencedirect.com/ for complete citation information).

[20] Ethelbert Ezemobi , Mario Silvagni, Ahmad Mozaffari, Andrea Tonoli and Amir Khajepour : “State of Health Estimation of Lithium-Ion Batteries in Electric Vehicles under Dynamic Load Conditions”

[21] Muhammad Umair Ali, Amad Zafar, Sarvar Hussain Nengroo, Sadam Hussain, Muhammad Junaid Alvi, and Hee-Je Kim. Towards a smarter battery management system for electric vehicle applications: A critical review of lithium-ion 131 Bibliography battery state of charge estimation. Energies, 12(3):446, 2019.

[22] Hardware-in-the-Loop Assessment of a Data-Driven State of Charge Estimation Method for Lithium-Ion Batteries in Hybrid Vehicles / Luciani, Sara; Feraco, Stefano; Bonfitto, Angelo; Tonoli, Andrea. - In: ELECTRONICS. - ISSN 2079- 9292. - 10:22(2021). [10.3390/electronics10222828]