energies

MDPI

Article

Temperature-Based State-of-Charge Estimation Using Neural Networks, Gradient Boosting Machine and a Jetson Nano Device for Batteries

Donghun Wang, Jihwan Hwang, Jonghyun Lee, Minchan Kim and Insoo Lee *

School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
* Correspondence: insoolee@knu.ac.kr; Tel.: +82-10-5312-5324

Abstract: Lithium-ion batteries are commonly used in electric vehicles, mobile phones, and laptops because of their environmentally friendly nature, high energy density, and long lifespan. Despite these advantages, lithium-ion batteries may experience overcharging or discharging if they are not continuously monitored, leading to fire and explosion risks, in cases of overcharging, and decreased capacity and lifespan, in cases of overdischarging. Another factor that can decrease the capacity of these batteries is their internal resistance, which varies with temperature. This study proposes an estimation method for the state of charge (SOC) using a neural network (NN) model that is highly applicable to the external temperatures of batteries. Data from a vehicle-driving simulator were used to collect battery data at temperatures of 25 °C, 30 °C, 35 °C, and 40 °C, including voltage, current, temperature, and time data. These data were used as inputs to generate the NN models. The NNs used to generate the model included the multilayer neural network (MNN), long short-term memory (LSTM), gated recurrent unit (GRU), and gradient boosting machine (GBM). The SOC of the battery was estimated using the model generated with a suitable temperature parameter and another model generated using all the data, regardless of the temperature parameter. The performance of the proposed method was confirmed, and the SOC-estimation results demonstrated that the average absolute errors of the proposed method were superior to those of the conventional technique. In the estimation of the battery's state of charge in real time using a Jetson Nano device, an average error of 2.26% was obtained when using the GRU-based model. This method can optimize battery performance, extend battery life, and maintain a high level of safety. It is expected to have a considerable impact on multiple environments and industries, such as electric vehicles, mobile phones, and laptops, by taking advantage of the lightweight and miniaturized form of the Jetson Nano device.

Keywords: lithium-ion battery; state of charge; multilayer neural network; long short-term memory; gated recurrent unit; gradient boosting machine; vehicle-driving simulator; Jetson Nano device; real time

Citation: Wang, D.; Hwang, J.; Lee, J.; Kim, M.; Lee, I. Temperature-Based State-of-Charge Estimation Using Neural Networks, Gradient Boosting Machine and a Jetson Nano Device for Batteries. *Energies* **2023**, *16*, 2639. https://doi.org/10.3390/en16062639

Academic Editor: Jun Young Cheong

Received: 8 February 2023
Revised: 8 March 2023
Accepted: 9 March 2023
Published: 10 March 2023

1. Introduction

In modern times, environmental pollution has become a major problem. Therefore, the use of environmentally friendly energy is important. Lithium-ion batteries are the most popular energy resources in energy-storage systems, electric vehicles, mobile phones, and laptops [1–3]. They have certain advantages, such as environmental friendliness, high energy density, high-efficiency charge and discharge, and long life. However, these batteries may experience overcharging and overdischarging when they are not continuously monitored. Overcharging may cause a fire or explosion, and overdischarging may increase the internal resistance of the battery, thereby decreasing its capacity and lifespan [4,5]. Furthermore, the internal resistance of the battery varies with temperature. It increases with decreases in temperature and degrades the battery's capacity [6]. If the state of charge (SOC)

is estimated using acquired data without considering the temperature during data acquisition, the estimation may become inaccurate. Therefore, acquiring data based on appropriate temperatures and estimating the SOC of batteries can increase SOC-estimation accuracy. The SOC is an important concept that represents the remaining capacity of a battery. When the SOC is 100% and 0%, the battery is completely charged and discharged, respectively. The accurate estimation of the SOC to prevent overcharging and overdischarging would considerably help to prevent battery damage and accidents.

The SOC-estimation method comprises two main approaches: model-based and data-driven methods [7–17]. The model-based method involves generating a model that is suitable for the data and estimating the SOC using the generated model. This method has high accuracy but requires professional knowledge to generate a model that fits the battery characteristics. Furthermore, a significant amount of time is required for model design. However, the data-driven method does not require battery expertise because it does not involve designing models. Moreover, the data-driven method has a relatively shorter development time than the model-based method. Machine learning is an important data-driven method.

In this study, a model was generated using the temperature measured during a discharge experiment, and the SOC of the battery was estimated using the generated model. The discharge experiment was conducted using a vehicle-driving simulator that simulates the output of a real vehicle. The vehicle-driving scenario applied to the simulator was the highway-fuel-economy test (HWFET) cycle used to measure vehicle-fuel efficiency in the United States. The SOC estimation models implemented using the acquired data were multilayer neural network (MNN), long short-term memory (LSTM), gated recurrent unit (GRU), and gradient boosting machine (GBM). The MNN was used as the most basic model, and the LSTM was applied because the battery data were time-series data. Owing to the small size of the datasets acquired by the simulator, GRU was used because of its advantage for small datasets. In addition, GBM, which has high performance with structured data, was used. In GBM, the tree is trained up to the Nth tree by reducing the residuals, which improves accuracy by effectively reducing the bias of individual decision trees. The results of these models were compared. The experimental process was as follows. First, the data acquired by the vehicle-driving simulator were classified based on the measured temperature and used as inputs for the model learning. Subsequently, the SOC was estimated using the models obtained by the Jetson Nano device, and the results were transferred to the user. This study presents several contributions. First, the authors utilized the lightweight and miniaturized form of the Jetson Nano device without depending on desktop or laptop devices, which can be used in various environments. Second, the paper compares the estimation performance using various NNs. Third, the authors set up a vehicle-driving simulator to conduct actual vehicle-driving experiments. Finally, this study confirms the high accuracy of SOC-estimation performance based on temperature, which adds to its overall contribution.

2. Vehicle-Driving Simulator and Jetson Nano Device

A vehicle-driving simulator that simulates the actual output of a vehicle was developed to estimate the SOC of the battery. The configuration of the overall system based on the vehicle-driving simulator and Jetson Nano device is depicted in Figure 1.

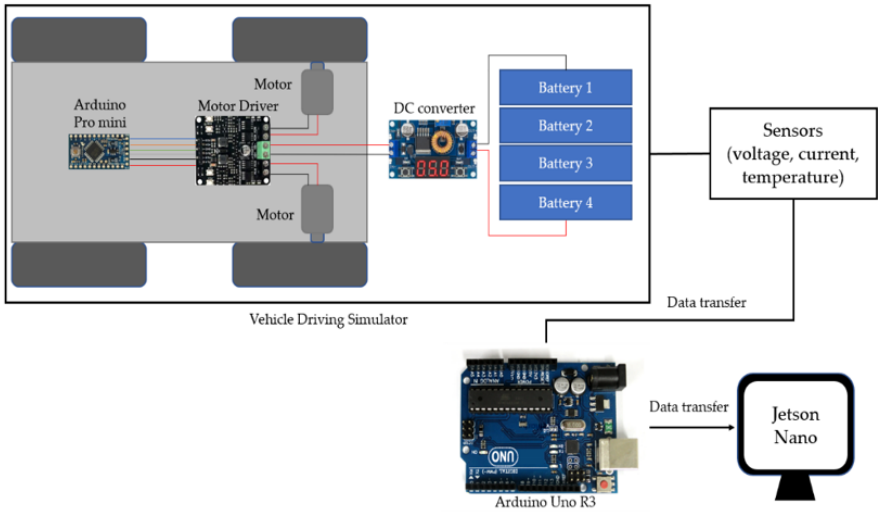


Figure 1. Configuration of the overall system based on the vehicle-driving simulator and Jetson Nano device.

The simulator comprised two direct-current (DC) motors (each with a speed of 6000 rpm and a rated voltage of 12 V), one MDD3A motor driver, one DC converter, one Arduino Pro Mini, one remote-controlled car frame, four batteries, and four tyres. Table 1 lists these items and their specifications. Each battery had a nominal voltage of 3.7 V and a rated capacity of 2000 mAh. The four batteries were connected in series and adjusted using a DC converter, and the adjusted voltage was used as the input voltage for the simulator.

Table 1. Specifications of items used for simulation and Jetson Nano Developer kit.

Item	**Motor Driver**	**Micro-Controller**	**Motor**	**Battery**
Specifications	MDD3A	Arduino Pro Mini	DC 12 V 6000 rpm 0.15 A	2000 mAh 3.7 V
Item	**GPU**	**CPU**	**Memory**	**Power**
Specifications	128-core maxwell	Quad-core ARM 1.43 GHz	4 GB 64-bit	5~10 W

The output of the simulator represents the Hyundai Avante Sports AD 1.6 model (Seoul, Republic of Korea) with 255/40/18 tires driving in the HWFET cycle. The HWFET is a highway-driving scenario defined by the United States Environmental Protection Agency to measure the fuel efficiency of a vehicle. The motor output simulated the third-gear ratio of the vehicle, and the speed of the simulator was controlled by the Arduino Pro Mini and motor driver. The HWFET is shown in Figure 2.

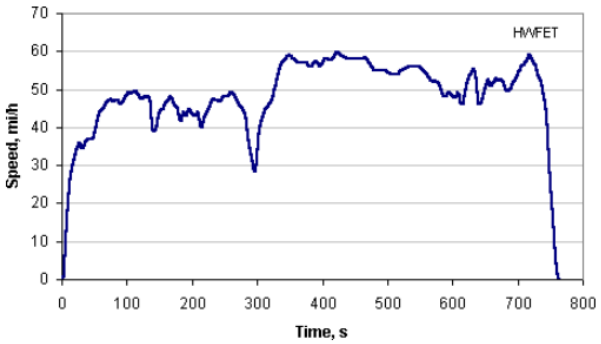


Figure 2. Highway-fuel-economy-test cycle.

In this method, Jetson Nano Device is used for performance of the SOC estimation. Jetson Nano is a single-board computer developed by NVIDIA and is one of the most popular devices for ML inference. It features heterogeneous CPU–GPU architecture, small form factor, light weight, and low power consumption. Moreover, it has a comprehensive development environment (JetPack SDK) and libraries developed for embedded applications, deep learning, and computer vision [18,19]. Table 1 shows the technical specifications of the Jetson Nano Developer Kit.

3. Proposed Temperature-Based SOC-Estimation Method

3.1. Temperature-Based Battery-SOC-Estimation Method

A SOC estimation method was proposed by selecting a suitable model for the measured temperature; it is shown in Figure 3.

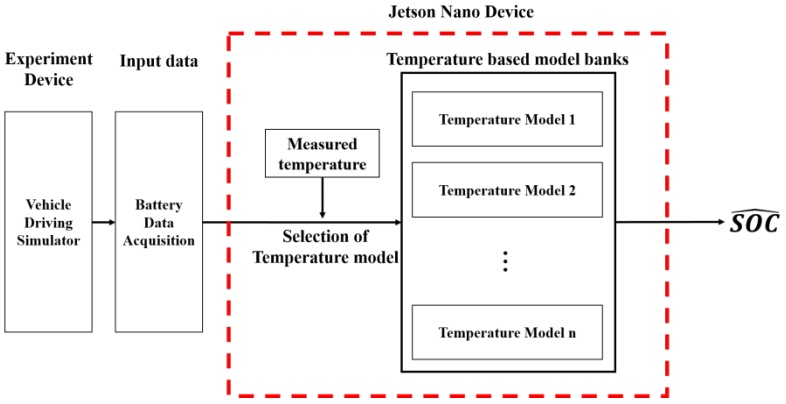


Figure 3. Proposed SOC-estimation method.

The proposed method in this study involves data categorization based on the operating temperature of the simulator and using it to develop models. In particular, the study-generated models for 25 °C, 30 °C, 35 °C, and 40 °C were based on their respective temperature datasets, and the SOC was estimated using the most appropriate model selected according to the temperature. To implement the proposed method, the authors used a vehicle-driving simulator to collect battery data. The estimated SOC was sent to the user after using the model. The Jetson Nano device followed the same process of applying the input data to the model for SOC estimation.

The SOC is the remaining capacity of the battery and is an important measure of the battery's state [18]. In this study, the Coulomb counting method was used to confirm the errors and results of the proposed SOC method. It was expressed as follows:

$$\mathrm{SOC}(t) = \mathrm{SOC}(0) - \int_0^t \frac{I(t)}{C_n} dt, \quad (1)$$

where SOC(0) denotes the initial measured capacity of the battery (%), C_n denotes the rated capacity of the battery (Ah), $I(t)$ denotes the current at time t (A), and SOC(t) denotes the SOC at time t (%).

3.2. Deep Neural Network Algorithms

3.2.1. Multilayer Neural Network

The MNN is a NN in which one or more hidden layers are added to a single perceptron. Because the perceptron has a problem that cannot be nonlinearly classified, the MNN can be used to solve the problem [20]. The structure of the MNN is illustrated in Figure 4.

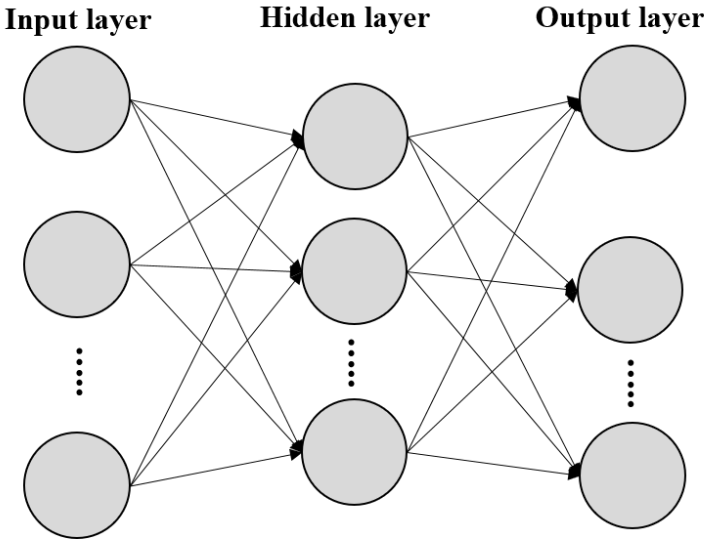


Figure 4. Structure of the MNN.

The MNN uses feedforward and backpropagation for learning, calculates the output value through feedforward, and corrects the error through backpropagation. In this study, the SOC-estimation model employed used voltage, current, temperature, and time parameters as input data and the SOC result as output data. The rectified linear unit (ReLU) function was used as the activation function in the MNN. Compared with the sigmoid function, ReLU has the advantages of nonvanishing gradient and fast convergence [21]. The equation for ReLU is as follows:

$$f(x) = \begin{cases} x, \ for \ x > 0 \\ 0, \ otherwise \end{cases}. \tag{2}$$

An Adam optimizer was used. It is a first-order gradient-based optimizer algorithm that combines momentum and root-mean-squared propagation (RMSprop); it is easy to implement and efficient because of its small amount of calculation [22]. The equation for Adam is as follows:

$$m_0 = 0, \ v_0 = 0, \ t = 0, \tag{3}$$

$$g_t = \nabla_\theta f_t(w_{t-1}), \tag{4}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{5}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{6}$$

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)}, \ \hat{v}_t = \frac{v_t}{(1 - \beta_2^t)}, \tag{7}$$

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{8}$$

where m_0, v_0, and t are initialized to 0; g_t denotes the gradient of the network; m_t denotes the first moment vector; v_t denotes the second moment vector; and β_1 and β_2 are the exponential decay rates for the moment estimates and have the following values: $\beta_1 = 0.9$ and $\beta_2 = 0.999$. At the start of the learning process, m_t and v_t were close to 0. Bias correction was applied to $\hat{m}$ and $\hat{v}$ to render them unbiased. Herein, w_t denotes the update of the weight, and α denotes the learning rate. The value of α is 0.001, and that of ε is 10^{-8}.

3.2.2. Long Short-Term Memory

The LSTM is a RNN in which a past output value affects a current input value [23]. The RNN has advantages in predicting time-series data. However, the problem of the vanishing gradient occurs with increases in learning time. The LSTM solves the problem by

adding the cell state and three gates (forget, input, and output) to the RNN. The structure of the LSTM is shown in Figure 5.

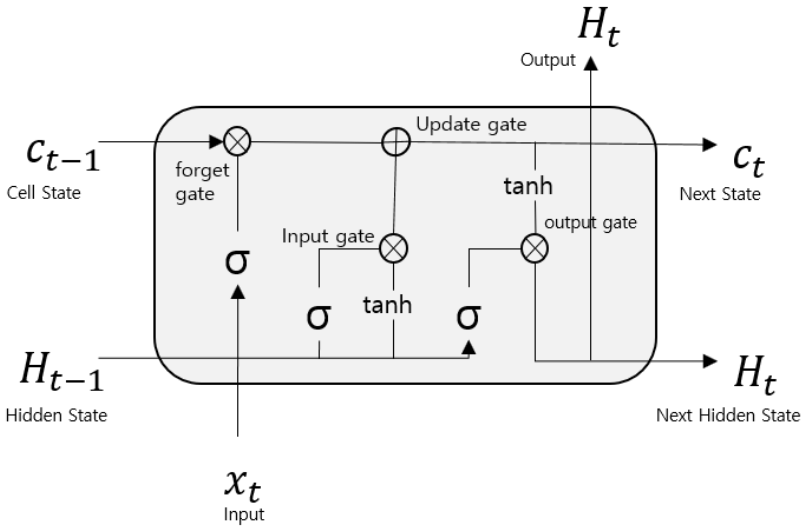


Figure 5. Structure of the LSTM.

The equations of the LSTM are as follows:

Step (1) Forget gate

$$f_t = \sigma\left(w_f \cdot [H_{t-1}, x_t]\right) + b_f; \tag{9}$$

Step (2) Input gate

$$i_t = \sigma(i \cdot [H_{t-1}, x_t]) + b_i, \tag{10}$$

$$\widetilde{C}_t = tanh(w_C \cdot [H_{t-1}, x_t]) + b_C; \tag{11}$$

Step (3) Cell-state update

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t; \tag{12}$$

Step (4) Output gate

$$O_t = \sigma(w_O \cdot [H_{t-1}, x_t]) + b_O, \tag{13}$$

$$H_t = O_t \cdot \tanh(C_t); \tag{14}$$

where H_{t-1} denotes the data of the previous cell; x_t denotes the current input data; w denotes the weight; b denotes the bias; f_t denotes the forget-gate value; $\widetilde{C}_t$ denotes the value of the previous cell calculated using tanh; C_t denotes the updated value of the cell state; O_t denotes the value of the output gate; and H_t denotes the output.

For the LSTM model employed for the SOC estimation, the Adam optimizer and tanh activation function were used. Compared with the sigmoid function, tanh demonstrates better performance for the gradient-vanishing problem, and when using the ReLU in the LSTM, the data diverge as the output value of the previous cell increases. The tanh function is expressed as follows:

$$tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}}. \tag{15}$$

3.2.3. Gated Recurrent Unit (GRU)

The GRU is a RNN that simplifies the LSTM. The GRU improves the shortcomings associated with long learning times due to the complex structure of the LSTM and demonstrates excellent performance for small datasets [24,25]. The GRU uses update and reset gates to determine the data of previous cells. The structure of the GRU is illustrated in Figure 6.

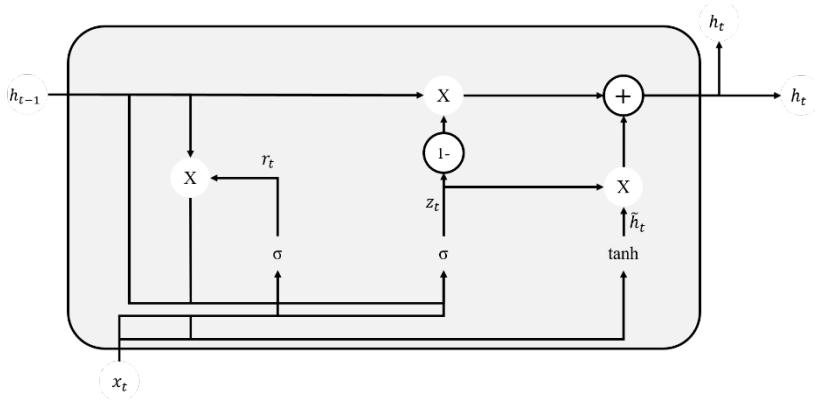


Figure 6. Structure of the GRU.

The equations of the GRU are as follows.
Step (1) Reset gate

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right); \tag{16}$$

Step (2) Update gate

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right); \tag{17}$$

Step (3) Candidate hidden state

$$\widetilde{h_t} = \tanh\left(W^{(h)}x_t + r_t \circ U^{(h)}h_{t-1}\right); \tag{18}$$

Step (4) Hidden state/output

$$h_t = z_t \circ \widetilde{h_t} + (1 - z_t) \circ h_{t-1}\,; \tag{19}$$

where r_t denotes the value of the reset gate; z_t denotes the value of the update gate; w and u denote the weights; h_{t-1} denotes the output of the previous cell; x_t denotes the input data of the current cell; $\widetilde{h_t}$ denotes the candidate value of the hidden state; and h_t denotes the output. The GRU represents h_t by selecting the necessary parts of $\widetilde{h_t}$ and h_{t-1}.

3.2.4. Gradient Boosting Machine

Ensemble learning is a technique for generating multiple classifiers and deriving more accurate predictions by combining them. Rather than using one robust model, it is a method of combining several weak models to help form more accurate predictions [26–30]. As one of the ML techniques, boosting is an algorithm that increases prediction or classification performance by sequentially combining several weak learners. The GBM regression method can be described as an ensemble of decision trees. The structure of the GBM is illustrated in Figure 7. Rather than building one tree, GBM predicts the outcome based on the regression model, which uses weak decision trees. This method of compiling decision trees to reflect residuals helps minimize errors in each of the next steps. This process should be repeated until the number of periods set by the hyperparameter is reached or the residual value is no longer reduced. The value from the second prediction tree was closer to the actual value than the first prediction tree, and the residual was lower. Therefore, prediction accuracy is improved as the utilization increases and the residual decreases.

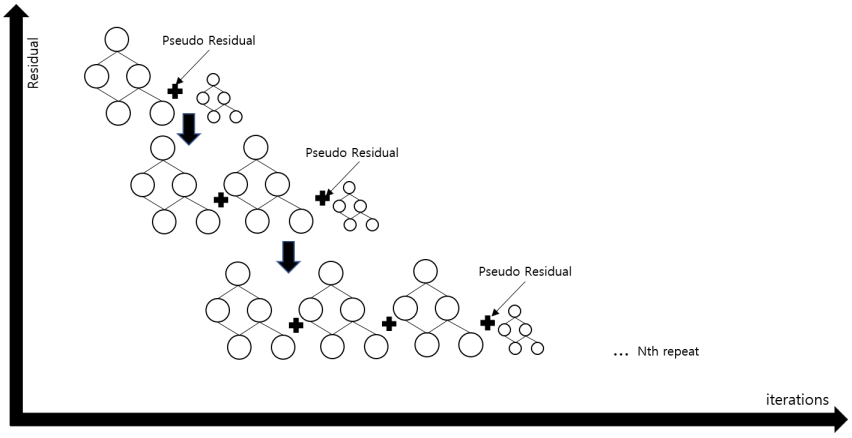


Figure 7. Structure of a GBM.

The equations of the GBM are as follows.

$$F_0(x) = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i,\ \alpha), \tag{20}$$

$$r_{im} = -\left[\frac{\partial L(y_i,\ F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{n-1}(x)}, \tag{21}$$

$$\gamma_n = arg\ \min_{\gamma} \sum_{i=1}^{N} L(y_i,\ F_{n-1}(x_i) + \gamma g_n(x_i)), \tag{22}$$

$$F_n(x) = F_{n-1}(x) + v\gamma_n g_n(x). \tag{23}$$

where $L(.)$ is the loss function; $F_0(x)$ is the initial model; r_{im} is the residual; g_n is the tree model of the n-th learning; γ_n is the optimal coefficient to update the model; v is the learning rate (default value is 0.1); and F_n is a model that has been learned n times.

4. Experimental Process and Results

4.1. Experimental Process

The experimental procedure was as follows. First, four lithium-ion batteries were fully charged with a constant voltage of 4.2 V; in particular, the batteries used in this study were lithium-ion-polymer full-cell batteries. The cathode was Li (NiCoMn)O_2, and the anode was graphite. The capacity of the fully charged battery was 100% SOC. After charging, the batteries were stabilized for 1 h and then connected in series to provide a voltage of 12 V via a DC converter. Finally, the external temperature of the batteries was adjusted using a thermostat, and the discharge experiment was conducted using the vehicle-driving simulator.

The discharge experiment was conducted until the motor of the vehicle-driving simulator stopped, and the data acquired through the experiment were defined as one cycle of the battery data. The acquired data comprised voltage, current, temperature, and time parameters. Six cycles of the battery data were used for the experiment, according to the temperature set during the operation of the vehicle driving simulator. The acquired data were then used as inputs for MNN, LSTM, and GRU, and the SOC was estimated using the generated models. The SOC-estimation model was created using Tensorflow and Keras based on Python.

4.2. MNN, LSTM, GRU, and GBM Models for SOC Prediction

In the study, the MNN model for the SOC prediction was applied first. The input parameters were obtained from the discharge experiment and comprised voltage, current, temperature, and time parameters. Five voltage values, five current values, one time value,

and one temperature value were used as the inputs to the MNN. For the SOC prediction (Table 2), the MNN had a structure of 12-256-128-1 and comprised layers in the following order: one input layer, two hidden layers, and one output layer. The number of epochs was 15,000. The ReLU activation function and Adam optimizer were used. The learning was considered complete when the mean squared error (MSE) was less than 10^{-6}. The structure of the MNN for the SOC prediction is shown in Figure 8.

Table 2. Hyper parameters of the MNN for SOC estimation.

Input_Layer	Hidden_Layer 1	Hidden_Layer 2	Output_Size	Activation Function	Epochs
12	256	128	1	Relu/Adam	15,000

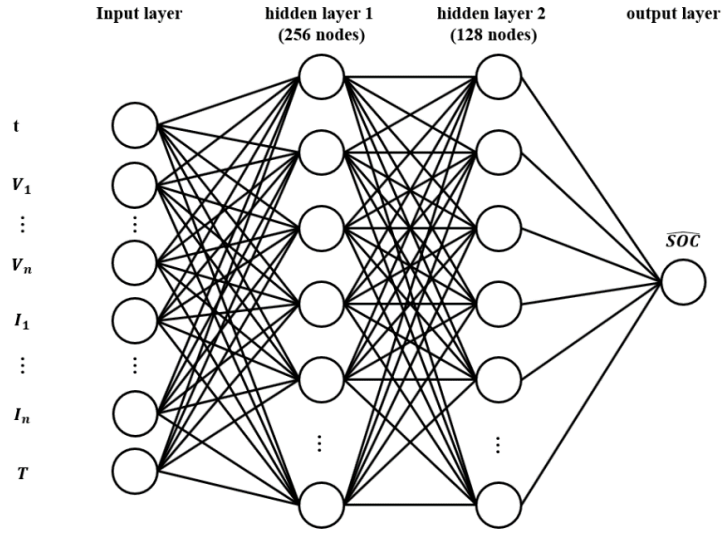


Figure 8. Structure of the MNN for SOC estimation.

In this study, the LSTM model for SOC prediction was applied second. The input parameters were the same as those in the MNN model. For the SOC estimation, the LSTM had a structure of 12-256-128-64-1 and comprised layers in the following order: one input layer, three hidden layers, and one output layer. The number of epochs was 5000. The tanh activation function and Adam optimizer were used. Figure 9 shows the structure of the LSTM for SOC prediction.

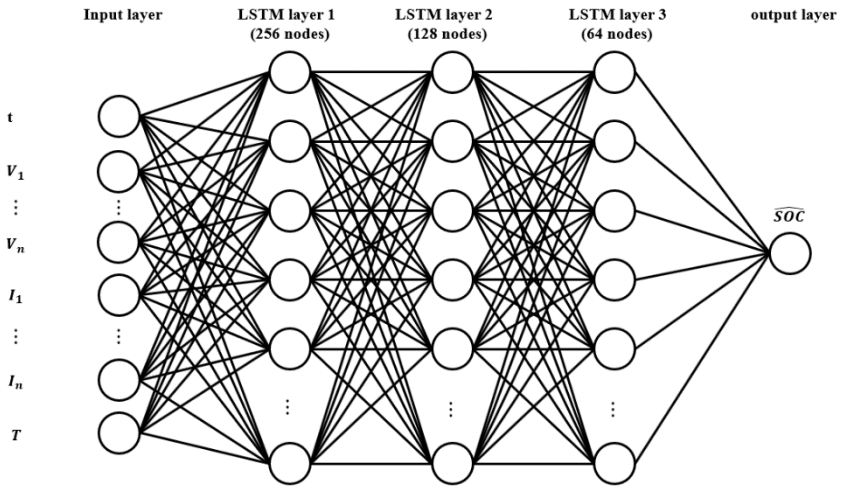


Figure 9. Structure of the LSTM for SOC estimation.

Next, the GRU model was used for the SOC prediction. The input parameters were the same as those in the MNN model. For the experiment, the GRU had a structure of 12-256-128-64-1 and comprised layers in the following order: one input layer, three hidden layers, and one output layer. The number of epochs was 200. The tanh activation function and Adam optimizer were used. Figure 10 shows the structure of the GRU for SOC prediction.

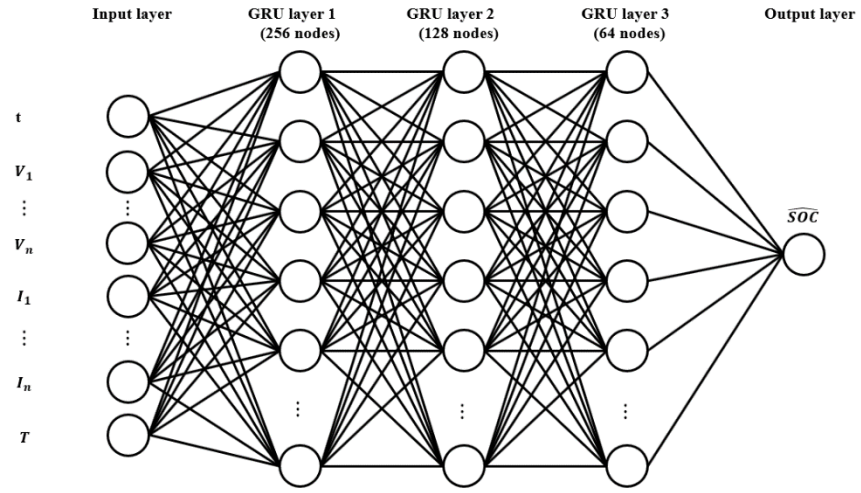


Figure 10. Structure of the GRU for SOC estimation.

In Figures 8–10, t denotes the time data, V denotes the voltage data, I denotes the current data, and T denotes the temperature data.

Finally, the estimation of the SOC using the GBM model was performed. The input parameters are same as those in the MNN model. For the experiment, the number of decision trees was 200, the trees' maximum depth was 5, the learning rate was 0.2, and another parameter was the default value. Note that 80% of the data were used as training data, and 20% were used as test data. The train_test_split() function of the sklearn library was used. The structure of the GBM for the SOC prediction is shown in Figure 11.

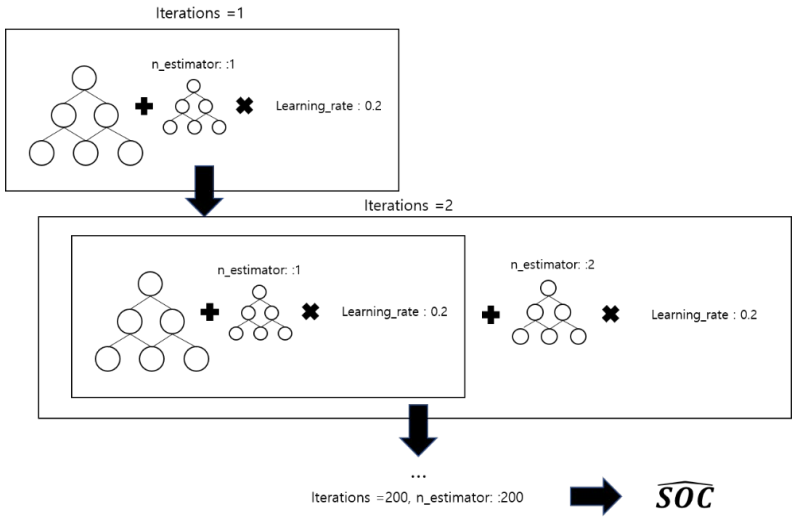


Figure 11. Structure of the GBM for SOC estimation.

The gradient boosting regression is a supervised learning algorithm that uses residuals to overcome the weaknesses of previous models and generate new models by linearly combining them. The gradient boost starts with a single leaf. Moreover, the target estimated value predicted by the single-leaf model is the average of all the target values. The difference

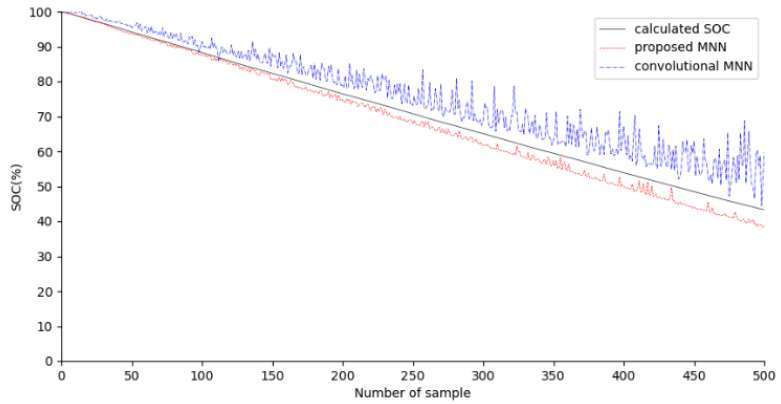
between the predicted value in the single leaf and the actual value is called a pseudo residual. Gradient boosting may cause overfitting; to prevent this, it is necessary to multiply by the learning rate. The learning rate is a hyperparameter that helps to ensure high accuracy in GBM learning processes.

4.3. Experimental Results

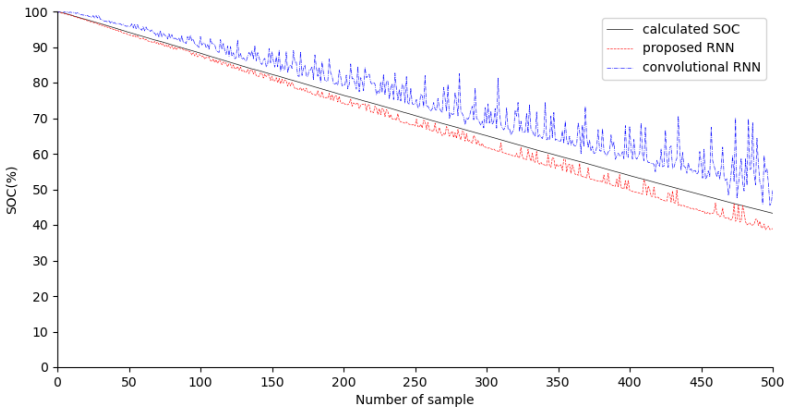
The SOC was estimated using the suitable MNN, LSTM, GRU, and GBM models according to the temperature measured during the discharge experiment. The SOC was estimated using the conventional method for comparison and to confirm the performance of the proposed method. For SOC estimation, the conventional method selects one model generated using all the measured parameters. In Figure 12A–D, we present the SOC-estimation results obtained for the MNN, LSTM, GRU, and GBM models. Each figure shows the results obtained using the proposed, conventional, and Coulomb counting methods. Tables 3–6 are the SOC errors obtained using each model. The estimated error was calculated using the mean absolute error (MAE), as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}|, \tag{24}$$

where n denotes the total number of parameters; x_i denotes the target value; and $\hat{x}$ denotes an estimate.

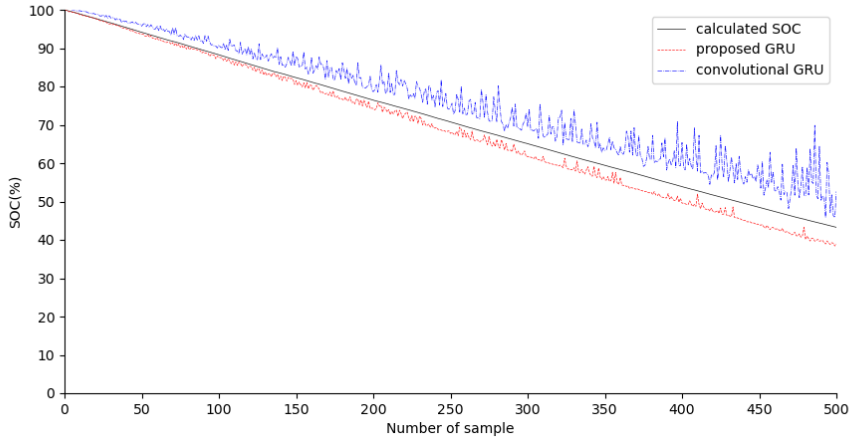


(**A**)

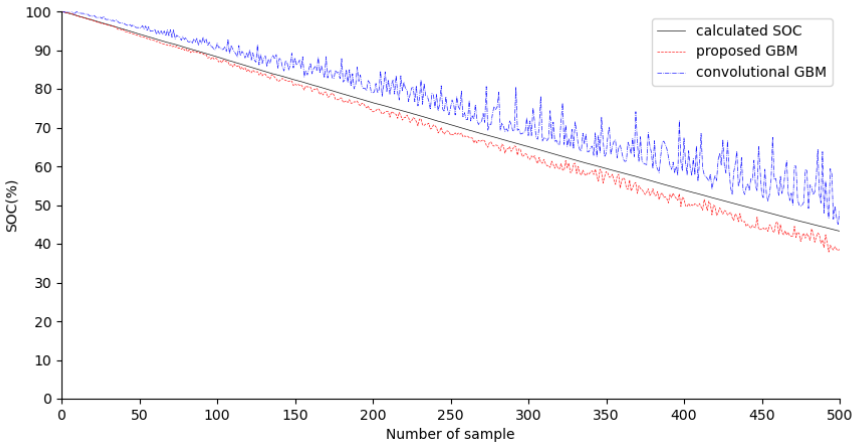


(**B**)

Figure 12. *Cont.*



(**C**)



(**D**)

Figure 12. (**A**) The SOC-estimation results for the MNN using the proposed and conventional methods; (**B**) the SOC-estimation results for the LSTM using the proposed and conventional methods; (**C**) the SOC-estimation results for the GRU using the proposed and conventional methods; (**D**) the SOC-estimation results for the GBM using the proposed and conventional methods.

Table 3. Hyperparameters of the LSTM for SOC estimation.

Input_Layer	Hidden_Layer 1	Hidden_Layer 2	Hidden_Layer 3	Output_Layer	Activation Function	Epochs
12	256	128	64	1	Tanh/Adam	5000

Table 4. Hyperparameters of the GRU for SOC estimation.

Input_Layer	Hidden_Layer 1	Hidden_Layer 2	Hidden_Layer 3	Output_Layer	Activation Function	Epochs
12	256	128	64	1	tanh/Adam	200

Table 5. Hyperparameters of the GBM for SOC estimation.

Number of Trees	Tree Max Depth	Learning Rate	Criterion
200	5	0.2	Friedman_mse

Table 6. The SOC errors obtained using the MNN model generated by the proposed and conventional methods.

Temperature	Method	Battery 1	Battery 2	Battery 3	Battery 4	Average
25 °C Model	Proposed	2.00%	2.46%	1.97%	1.87%	2.07%
	Conventional	3.77%	1.54%	4.19%	2.21%	2.93%
30 °C Model	Proposed	1.65%	1.63%	2.40%	2.03%	1.93%
	Conventional	1.86%	1.53%	2.33%	2.15%	1.97%
35 °C Model	Proposed	2.09%	2.77%	3.89%	2.53%	2.82%
	Conventional	8.45%	8.16%	11.52%	7.81%	9.00%
40 °C Model	Proposed	0.98%	2.22%	2.00%	2.24%	1.86%
	Conventional	4.30%	4.12%	2.39%	5.04%	3.96%

Table 6 shows the SOC errors obtained using the MNN model generated by the proposed and conventional methods. The proposed method achieved minimum and maximum errors of 0.98% and 3.89%, respectively. The best average error, according to the temperature, was 3.89%, obtained from the model at 40 °C. The total average error of the proposed method was 2.17% and that of the conventional method was 4.46%.

Table 7 represents the SOC errors obtained using the LSTM model generated by the proposed and conventional methods. The proposed method achieved minimum and minimum errors of 0.93% and 3.31%, respectively. The best average error in terms of temperature was 1.82%, obtained from the model at 40 °C. The total average error of the proposed method was 2.19% and that of the conventional method was 4.60%.

Table 7. The SOC errors obtained using the LSTM model generated by the proposed and conventional methods.

Temperature	Method	Battery 1	Battery 2	Battery 3	Battery 4	Average
25 °C Model	Proposed	2.26%	2.83%	1.87%	2.06%	2.26%
	Conventional	3.97%	1.69%	5.00%	2.58%	3.31%
30 °C Model	Proposed	1.96%	1.72%	2.40%	1.72%	1.95%
	Conventional	1.88%	1.60%	2.85%	2.02%	2.09%
35 °C Model	Proposed	2.12%	2.96%	3.31%	2.54%	2.73%
	Conventional	8.63%	8.24%	12.41%	8.09%	9.34%
40 °C Model	Proposed	0.93%	2.13%	2.07%	2.17%	1.82%
	Conventional	3.97%	3.55%	2.07%	5.00%	3.65%

Table 8 presents the SOC errors obtained using the GRU model generated by the proposed and conventional methods. The proposed method achieved minimum and maximum errors of 1.43% 2.96%, respectively. The best average error in terms of temperature was 1.43%, obtained from the model at 30 °C. Table 5 summarizes the average battery errors obtained using the generated models. The average errors of the proposed and conventional methods were 2.13% and 4.40%, respectively.

Table 8. The SOC errors obtained using the GRU model generated by the proposed method.

Temperature	Method	Battery 1	Battery 2	Battery 3	Battery 4	Average
25 °C Model	Proposed	2.80%	3.79%	1.51%	1.79%	2.47%
	Conventional	3.33%	3.22%	3.96%	2.19%	3.18%

Table 8. *Cont.*

Temperature	Method	Battery 1	Battery 2	Battery 3	Battery 4	Average
30 °C	Proposed	0.60%	1.31%	1.44%	2.36%	1.43%
Model	Conventional	1.07%	1.10%	3.14%	2.05%	1.84%
35 °C	Proposed	2.21%	3.28%	4.46%	1.88%	2.96%
Model	Conventional	8.61%	8.44%	12.66%	7.76%	9.37%
40 °C	Proposed	1.07%	2.15%	1.88%	1.52%	1.66%
Model	Conventional	4.21%	4.65%	2.52%	4.63%	4.00%

Table 9 presents the SOC errors obtained using the GBM model generated by the proposed and conventional methods. The proposed method achieved minimum and maximum errors of 1.27% and 2.61%, respectively. The best average error in terms of temperature was 1.67%, obtained from the model at 30 °C. Table 6 summarizes the average battery errors obtained using the generated models. The average error of the proposed method was lower by more than 2.46% than that of the conventional method for the all models.

Table 9. The SOC errors obtained using the GBM model generated by the proposed method.

Temperature	Method	Battery 1	Battery 2	Battery 3	Battery 4	Average
25 °C	Proposed	1.86%	2.45%	1.74%	1.66%	1.92%
Model	Conventional	3.38%	1.92%	3.35%	2.78%	2.86%
30 °C	Proposed	1.28%	1.57%	2.05%	1.76%	1.67%
Model	Conventional	1.54%	1.38%	2.11%	1.63%	1.66%
35 °C	Proposed	2.20%	2.58%	2.61%	1.99%	2.35%
Model	Conventional	8.46%	8.36%	10.58%	7.83%	8.80%
40 °C	Proposed	1.27%	1.98%	1.91%	1.92%	1.77%
Model	Conventional	4.66%	3.88%	3.33%	4.92%	4.20%

Figure 13 and Table 10 show the average MAEs, which were calculated based on the temperature and used to determine the average errors. It was confirmed that the proposed method outperformed the conventional method.

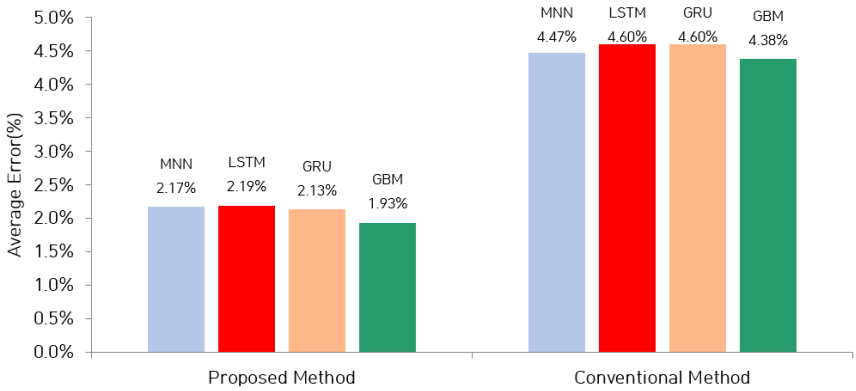


Figure 13. The MAE obtained using the proposed and conventional methods.

Table 10. Average battery errors produced using the generated models.

Method	Errors of Proposed Methods				Errors of Conventional Methods			
	MNN	LSTM	GRU	GBM	MNN	LSTM	GRU	GBM
Average Error	2.17%	2.19%	2.13%	1.93%	4.47%	4.60%	4.60%	4.38%

Table 11 presents the time taken to estimate the SOC using MNN, RNN, GRU and GBM. The experimental results demonstrate that the GBM outperformed the other models in terms of training duration, with a time of 00:00:01.48. However, the LSTM model had the longest training duration, of 2:09:10.01, with the MNN and GRU models taking 00:13:31.05 and 00:01:16.30, respectively.

Table 11. The SOC estimation times for MNN, RNN, GRU, and GBM.

Model	MNN	LSTM	GRU	GBM
Time	00:13:31.05	2:09:10.01	00:01:16.30	00:00:01.48

Online SOC Estimation by GRU

The SOC was estimated in real time using the data and expressed using graphs. Table 12 represents the real-time SOC-estimation error obtained using the Jetson Nano device and the vehicle-driving simulator; the real-time graphs of the SOC are shown in Figure 14. First, the input parameters were obtained using the vehicle-driving simulator, and the acquired data were then transferred to the Jetson Nano device as the input of the generated model for the SOC prediction.

Table 12. Real-time SOC-estimation errors using the GRU model.

Network Type	Battery 1	Battery 2	Battery 3	Battery 4
35 °C GRU	2.26%	2.83%	1.87%	2.06%

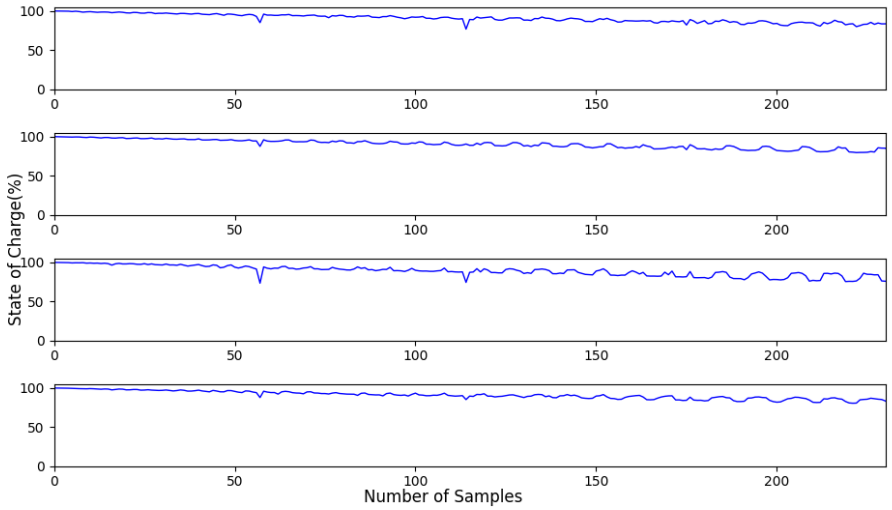


Figure 14. Real-time SOC estimation using the GRU model.

5. Conclusions

This study developed a method for estimating the SOC by selecting a suitable model according to the temperatures measured during an experiment. For the SOC estimation, a discharge experiment was conducted using a custom vehicle-driving simulator. The data acquired during the experiment were classified according to temperature and used as inputs for the MNN, LSTM, and GRU models. Finally, the SOC was estimated using the model generated from the data by the Jetson Nano device.

During the experiment, four temperatures were measured, and the SOC was estimated using the MNN, LSTM, and GRU models, according to temperature. Most of the proposed methods exhibited fewer errors than the conventional methods. The proposed MNN method achieved an average error of 2.17%, which was superior to the 4.46% obtained by using the conventional MNN method. The LSTM method achieved an average error of

2.19%, which was better than that obtained using the conventional MNN method (4.46%). The proposed GRU method demonstrated an average error of 2.15%, which was better than that obtained using the conventional GRU method (4.40%). The GBM method achieved an average error of 1.93%, which was better than that obtained by using the conventional GBM method (4.38%). The experimental results regarding computational time demonstrate that the GBM outperformed the other models in term of training time (00:00:01.48). These results suggest that the GBM algorithm can be an efficient tool for time-sensitive applications that require rapid model training and deployment.

The battery data were obtained using the Jetson Nano device for real-time SOC prediction. The SOC was expressed through a graphical user interface. The GRU model based on the data from the model at 35 °C was used for the real-time SOC estimation, and the average error for the SOC estimation was 2.25%.

In future research, the authors plan to incorporate a model-update function for SOC prediction using measured data, along with the ability to select a suitable model based on measured temperatures. Moreover, they plan to utilize Jetson Nano to enable real-time SOC predictions, with the addition of a GBM model, since the boosting model has time-related benefits, such as faster learning and improved scalability. In future research, the authors plan to apply several boosting models with the aforementioned advantages for SOC estimation. It is anticipated that the proposed method will have practical applications for solving SOC problems for real battery systems.

Author Contributions: Methodology, D.W. and I.L.; investigation, D.W., J.L., M.K. and I.L.; experiments, D.W., J.L. and M.K.; simulations, D.W.; data analysis, D.W., J.L. and I.L.; writing—original draft, D.W. and J.H.; writing—review and editing, D.W. and I.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the BK21 FOUR project, funded by the Ministry of Education, Korea (4199990113966).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zuo, H.; Zhang, B.; Huang, Z.; Wei, K.; Zhu, H.; Tan, J. Effect analysis on SOC values of the power lithium manganate battery during discharging process and its intelligent estimation. *Energy* **2021**, *238*, 121854. [CrossRef]
2. Panchal, S.; Dincer, I.; Agelin-Chaab, M.; Fraser, R.; Fowler, M. Experimental temperature distributions in a prismatic lithium-ion battery at varying conditions. *Int. Commun. Heat Mass Transf.* **2016**, *71*, 35–43. [CrossRef]
3. Tong, W.; Koh, W.Q.; Birgersson, E.; Mujumdar, A.S.; Yap, C. Correlating uncertainties of a lithium-ion battery—A Monte Carlo simulation. *Int. J. Energy Res.* **2015**, *39*, 778–788. [CrossRef]
4. An, K.; Barai, P.; Smith, K.; Mukherjee, P.P. Probing the thermal implications in mechanical degradation of lithium-ion battery electrodes. *J. Electrochem. Soc.* **2014**, *161*, 1058–1070. [CrossRef]
5. Chen, J.; Ouyang, Q.; Xu, C.; Su, H. Neural network-based state of charge observer design for lithium-ion batteries. *IEEE Trans. Control. Syst. Technol.* **2018**, *26*, 313–320. [CrossRef]
6. Ma, S.; Jiang, M.; Tao, P.; Song, C.; Wu, J.; Wang, J.; Deng, T.; Shang, W. Temperature effect and thermal impact in lithium-ion batteries: A review. *Prog. Nat. Sci.* **2018**, *28*, 653–666. [CrossRef]
7. Cheng, K.W.E.; Divakar, B.P.; Wu, H.; Ding, K.; Ho, H.F. Battery-management system (BMS) and SOC development for electrical vehicles. *IEEE Trans. Veh. Technol.* **2011**, *60*, 76–88. [CrossRef]
8. Hu, X.; Li, S.; Peng, H. A comparative study of equivalent circuit models for Li-ion batteries. *J. Power Sources* **2012**, *198*, 359–367. [CrossRef]
9. How, D.N.T.; Hannan, M.A.; Hossain Lipu, M.S.; Ker, P.J. State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review. *IEEE Access* **2019**, *7*, 136116–136136. [CrossRef]
10. Anton, J.C.A.; Nieto, P.J.G.; Viejo, C.B.; Vilan, J.A.V. Support vector machines used to estimate the battery state of charge. *IEEE Trans. Power Electron.* **2013**, *28*, 5919–5926. [CrossRef]
11. Bian, C.; He, H.; Yang, S. Stacked bidirectional long short-term memory networks for state-of-charge estimation of lithium-ion batteries. *Energy* **2020**, *191*, 116538. [CrossRef]
12. Park, J.; Lee, J.; Kim, S.; Lee, I. Real-time state of charge estimation for each cell of lithium battery pack using neural networks. *Appl. Sci.* **2020**, *10*, 8644. [CrossRef]

13. Lee, J.-H.; Lee, I.-S. Lithium battery SOH monitoring and an SOC estimation algorithm based on the SOH result. *Energies* **2021**, *14*, 4506. [CrossRef]
14. Lai, X.; Yi, W.; Cui, Y.; Qin, C.; Han, X.; Sun, T.; Zhou, L.; Zheng, Y. Capacity estimation of lithium-ion cells by combining model-based and data-driven methods based on a sequential extended Kalman filter. *Energy* **2020**, *216*, 119233. [CrossRef]
15. Xiao, F.; Li, C.; Fan, Y.; Yang, G.; Tang, X. State of charge estimation for lithium-ion battery based on gaussian process regression with deep recurrent kernel. *Int. J. Electr. Power Energy Syst.* **2021**, *124*, 106369. [CrossRef]
16. Chandran, V.; Patil, C.; Karthick, A.; Ganeshaperumal, D.; Rahim, R.; Ghosh, A. State of charge estimation of lithium-ion battery for electric vehicles using machine learning algorithms. *World Electr. Veh. J.* **2021**, *12*, 38. [CrossRef]
17. Wang, D.; Lee, J.; Kim, M.; Lee, I. Neural network-based state of charge estimation method for lithium-ion batteries Based on temperature. *IASC* **2023**, *36*, 20–25. [CrossRef]
18. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 3 September 2014; pp. 1724–1734.
19. Wang, Y.-X.; Chen, Z.; Zhang, W. Lithium-ion battery state-of-charge estimation for small target sample sets using the improved GRU-based transfer learning. *Energy* **2022**, *244*, 123178. [CrossRef]
20. Zhang, R.; Xia, B.; Li, B.; Cao, L.; Lai, Y.; Zheng, W.; Wang, H.; Wang, W. State of the art of lithium-ion battery SOC estimation for electrical vehicles. *Energies* **2018**, *11*, 1820. [CrossRef]
21. He, S.; Reif, K.; Unbehauen, R. Multilayer neural networks for solving a class of partial differential equations. *Neural Netw.* **2000**, *13*, 385–396. [CrossRef]
22. Alhagry, S.; Fahmy, A.A.; El-Khoribi, R.A. Emotion recognition based on EEG using LSTM recurrent neural network. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 10. [CrossRef]
23. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 30 January 2017.
24. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef] [PubMed]
25. Assunção, E.; Gaspar, P.D.; Mesquita, R.; Simões, M.P.; Alibabaei, K.; Veiros, A.; Proença, H. Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism. *Remote. Sens.* **2022**, *14*, 4217. [CrossRef]
26. Süzen, A.A.; Duman, B.; Şen, B. Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN. In Proceedings of the 2020 International Congress on Human -Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 28 June 2020.
27. Zhou, Z.H. *Ensemble Methods: Foundations and Algorithms*; Chapman and Hall/CRC: London, UK, 2012.
28. Natekin, A.; Knoll, A. Gradient boosting machines, a tutorial. *Front. Neurorobotics* **2013**, *7*, 21. [CrossRef]
29. Ivanna, B. Kristian Marinsen Prediction of geometry deviations in additive manufactured parts: Comparison of linear regression with machine learning algorithms. *J. Intell. Manuf.* **2021**, *54*, 1937–1967.
30. Bentéjac, C.; Csörgő, A.; Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. *Artif. Intell. Rev.* **2021**, *54*, 1937–1967. [CrossRef]