IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 28, 2019, accepted April 14, 2019, date of publication April 24, 2019, date of current version May 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913078

Accurate State-of-Charge Estimation Approach for Lithium-Ion Batteries by Gated Recurrent Unit With Ensemble Optimizer

BIN XIAO, YONGGUI LIU, AND BING XIAO
School of Automation Science and Engineering, South China University of Technology, Guangzhou 510641, China

Corresponding author: Bing Xiao (aubxiao@scut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573153, in part by the Natural Science Foundation of Guangdong Province under Grant 2015A010106005 and Grant 2016A030313510, and in part by the Scientific Research Foundation of Central Universities under Grant 2018ZD51.

ABSTRACT State-of-charge (SoC) estimation is indispensable for battery management systems (BMSs). Accurate SoC estimation can improve the efficiency of battery utilization, especially for electric vehicles (EVs). Several kinds of battery SoC estimation approaches have been developed, but a simple and efficient method for battery SoC estimation that can adapt to a variety of lithium-ion batteries is worth exploring. To this end, a recurrent neural network (RNN) model based on a gated recurrent unit (GRU) is presented for battery SoC estimation. The GRU-RNN model can rapidly learn its own parameters by means of an ensemble optimization method based on the Nadam and AdaMax optimizers. The Nadam optimizer is used in the model pre-training phase to find the minimum optimized value as soon as possible, and then the AdaMax optimizer is used in the model fine-tuning phase to further determine the model parameters. To validate the effectiveness and robustness of the proposed method, the GRU-RNN model was trained and tested with three kinds of dynamic loading profiles and compared with existing SoC estimation methods. The experimental results show that the proposed method dramatically reduces the model training time and increases estimation accuracy.

INDEX TERMS Lithium-ion batteries, state of charge, gated recurrent unit, ensemble optimizer.

I. INTRODUCTION

The boom in electric vehicles (EVs) has motivated the study of battery management systems (BMSs). An efficient BMS is able to guarantee battery safety, and extend the battery's service life. As a key battery parameter, the state of charge (SoC) directly represents the remaining electrical energy of battery. Accurate SoC estimation is essential for a BMS to predict the remaining useful life (RUL) of a battery, and overcharging and over-discharging of the battery can also be avoided based on accurate SoC estimation. Thus, the role and importance of the SoC in BMS have led to extensive research and the emergence of various studies on battery SoC estimation [1].

A. REVIEW OF ESTIMATION APPROACHES

The traditional methods of battery SoC estimation mainly include the Coulomb counting method, the open-circuit voltage (OCV) method, and the electrochemical impedance spectroscopy (EIS) method [2]–[4]. These methods do not require the establishment of a battery model. So they are referred to as non-model-based or open-loop approaches. With such methods, estimation errors on the battery SoC caused by external disturbances such as variations in battery temperature or ambient noise, cannot be adaptively decreased.

The equivalent circuit model (ECM) utilizes resistors and capacitors (RCs) to simulate the electrical characteristics of a battery. Such model-based approaches include the first-order RC models, the second-order RC models, and the higher-order models [5]. As the model order increases, the computational complexity also significantly increases, which hinders the practical application of the higher-order models. The ECM approach only considers the electronic behaviors of a battery, it does not account for the internal characteristics of the battery, meaning that it cannot yield a complete

The associate editor coordinating the review of this manuscript and approving it for publication was Qiquan Qiao.

description of the battery. As an alternative to the ECM approach, an electrochemical model inherently describes the electrochemical reactions inside a battery in the form of partial differential equations [6], however, its extreme complexity prevents implementation on target microcontrollers.

Observer-based approaches provide another class of effective methods for battery SoC estimation. The well-known sliding mode observer technique, if combined with an adaptive switching model, can be used to improve the accuracy of battery SoC estimation [7]. Aiming at high accuracy, a method of SoC estimation based on a nonlinear fractional model has been proposed, and good performance has been achieved with a Luenberger-type observer [8]. Based on the linear ECM approach, a SoC estimation method using an $H\infty$ switched observer has been presented that guarantees robustness against inaccuracies in the initial state and disturbances [9]. Thus, observer-based methods offer enhanced robustness and accuracy in the face of model uncertainty and disturbances. However, parameter tuning and the Lyapunov stability proof are obstacles that need to be overcome in observer design.

Because of their high accuracy and fast convergence, filtering techniques, such as extended Kalman filter (EKF), unscented Kalman filter (UKF), Particle filter (PF), are widely used for battery SoC estimation [10]–[12]. However, a filter-based approach requires accurate knowledge of the battery model parameters and the covariance matrix of the measurement noise. It is difficult to obtain accurate model parameters in practical applications, because of the time-varying characteristics of lithium-ion batteries, and also because the errors do not correspond to white noise.

Data-driven and machine learning methods are also applied for SoC estimation. Using a moving window method with limited initial training samples, a battery SoC estimation approach based on the least-squares support vector machine (LS-SVM) has been proposed [13]. Methods based on neural networks (NNs) can be used to accurately estimate the SoC, but they require considerable time for model training [14]–[15]. In particular, for a deep feedforward NN (DF-NN) [14], the training time can reach 40 h when implemented on a graphics processing unit (GPU). A novel machine learning method using an RNN model with long short-term memory (LSTM) can perform accurate SoC estimation, but model training takes up 9 h to complete when using the Adam optimizer [21] implemented on a GPU [16]. For these methods, the demand for large number of data samples and the time-consuming model training are the main difficulties.

The approaches mentioned above can be roughly classified into three categories: traditional estimation methods [2]–[7], control-theory-based methods [8]–[12], and data-driven methods [13]–[16]. Methods in the first class are the simplest and easiest to implement in embedded devices, but they suffer from the effects of ambient temperature variations, inaccuracies in the initial value of the battery SoC, and long-duration of terminal voltage measurements. Methods in the second class can achieve highly precise and adaptive SoC estimation. However, whether EKF or PF is chosen for SoC estimation, inevitable difficulties arise in regard to parameter identification, battery modeling, and the complex proof of stability. This third class of methods represents a new research direction for battery SoC estimation, which requires many data samples and time-consuming model training. Although the estimation performance is gradually improving with the recent explosive developments in machine learning, more progress can still be made. However, the particular characteristics of the battery SoC estimation task present certain challenges.

The SoC of a battery cannot be measured directly, it can only be obtained through estimation methods, and the complex electrochemical reactions occurring in the battery and its strongly nonlinear characteristics increase the difficulty of SoC estimation. Moreover, the battery performance is also affected by variations in the ambient temperature, which result in changes to the battery's internal resistance, in turn making it difficult to estimate the battery SoC. These internal characteristics and external influencing factors pose major challenges in SoC estimation. Furthermore, the rapid development of EVs has motivated new developments in power batteries. If the battery SoC is still to be estimated based on battery models, then battery modeling work will continue to be necessary for various new batteries, and a significant amount of time will be required to calibrate the model parameters. Therefore, developing a simple and efficient SoC estimation method that can adapt to a variety of different batteries is a very meaningful and challenging task.

B. CONTRIBUTIONS AND ORGANIZATION

To conquer such challenges, a GRU-RNN model optimized via an ensemble optimization method is proposed for the SoC estimation of lithium-ion batteries. The specific research contributions made in this paper are as follows.

1) A new prediction method is proposed, in which the problem of SoC estimation is converted into a sequence prediction problem based on GRU-RNN model. After being trained on dynamic driving cycle data, the GRU-RNN model can work efficiently without any dependence on a battery model, complex mathematical calculations, or Lyapunov stability proof. Experimental results show that this approach is quite robust and highly precise.

2) A new optimization method called ensemble optimization algorithm is proposed, which uses the Nadam [24] and AdaMax [21] algorithm as backpropagation optimization algorithms to achieve parameter self-learning for the GRU-RNN model. Not only can the model training be completed rapidly and stably with this ensemble optimizer, but the training time can also be reduced.

The remainder of this paper is organized as follows. Section 2 introduces the modeling approach. The ensemble optimizer is introduced in Section 3. The SoC estimation results and analysis are presented in Section 4. Section 5 serves as a conclusion.

II. MODELING AND METHODS

A. PROBLEM FORMULATION

Normally, the SoC of a battery is defined as the ratio of the battery's remaining capacity to its nominal capacity. The formula is as follows:

$$SoC(t) = SoC(t_0) - \frac{1}{C_n}\int_{t_0}^{t} \eta_c I(\tau) d\tau \tag{1}$$

where $SoC(t)$ is the value at time t, $SoC(t_0)$ is the initial value at time t_0, C_n is the battery's nominal capacity, η_c is the Columbic efficiency, and I is the battery current.

Based on the above formula, the SoC varies continuously with time. Thus, the SoC estimation of the battery can be treated as a time-series prediction problem. Then, the focus is placed on providing input series to the appropriate model to ensure accurate output series. Generally, current, voltage, and temperature are considered to be the most influential factors related to battery SoC estimation. Therefore, the data set used to train the model can be represented as shown below:

$$\Psi = \{X, Y\} \tag{2}$$

where $X = (x_1, x_2, .., x_n)$ and $Y = (y_1, y_2, .., y_n)$ are the input and output sequences, respectively. Here, $x_k = [I(k), V(k), T(k)]$ and $y_k = [SoC(k)]$, where $I(k), V(k), T(k)$ and $SoC(k)$ are the current, voltage, temperature and SoC of the battery as measured at time step k.

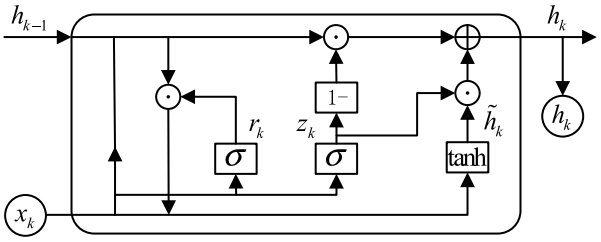


FIGURE 1. GRU structure.

B. GRU-RNN MODELING

The RNN architecture has been one of the most popular technologies for machine learning in recent years. It can make full use of time-series data samples to ensure the accuracy of its predictions, but it is susceptible to the problem of exploding or vanishing gradients [17]. Although LSTM-based RNN models have achieved state-of-the-art performance on multiple machine learning tasks, the gating mechanism leads to considerable complexity [18]. As an alternative, the GRU architecture [19] is a variant of the LSTM architecture with one fewer gate. Compared with an LSTM-based RNN model, a GRU-based RNN model has a simpler structure and fewer parameters, thus making model training easier. The GRU structure is shown in Fig. 1.

A GRU can be represented by the following block functions:

$$\begin{cases} z_k = \sigma(W_z^x x_k + W_z^h h_{k-1} + b_z) \\ r_k = \sigma(W_r^x x_k + W_r^h h_{k-1} + b_r) \\ \tilde{h}_k = \tanh(W_h^x x_k + W_h^h (h_{k-1} \odot r_k) + b_h) \\ h_k = z_k \odot \tilde{h}_k + (1 - z_k) \odot h_{k-1} \end{cases} \tag{3}$$

where z_k and r_k are the vectors for gate updating and resetting, respectively; h_k and $\tilde{h}_k$ are the state vector and the candidate state, respectively, at time step k; σ is the sigmoid function, as defined in Eq. (4); $\odot$ is the element-wise multiplication operator; x_k is the input vector; W_z^x, W_z^h, and b_z are the parameters of the gate updating operation, W_r^x, W_r^h, and b_r are the parameters of the gate resetting operation; W_h^x, W_h^h, and b_h are the parameters of the candidate state.

$$\sigma(x) = 1/(1 + e^{-x}) \tag{4}$$

The architecture of the proposed GRU-RNN model consists of an input layer, a hidden layer, a fully-connected dense layer and an output layer, which are denoted by l_1, l_2, l_3, and l_4, respectively, as shown in Fig. 2. The first layer is the input layer, followed by the GRU hidden layer; its unfolded structure is shown on the right side of Fig. 2. GRU_{nu} is the *nu-th* unit, where *nu* is the number of GRUs in the hidden layer.

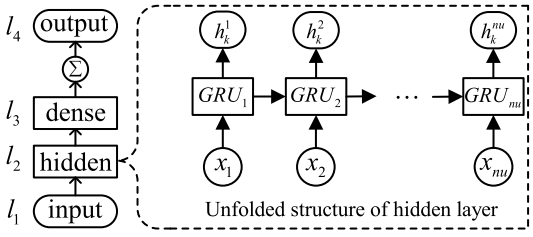


FIGURE 2. GRU-RNN model architecture.

To prevent overfitting, the dropout technique is applied in the hidden layer [20]. The third layer is a fully connected dense layer that applies a linear transformation to obtain the SoC estimation result by means of the sigmoid activation function, which is done as follows:

$$SoC_k^* = \sigma(W_k^s h_k + b_s) \tag{5}$$

where W_k^s and b_s are the weight vector and biases, respectively, of the fully-connected layer at time step k. The mean square error (MSE) is chosen as the loss function; it is defined as shown in (6):

$$L = (\sum_{k=1}^{l} (SoC_k - SoC_k^*)^2)/l \tag{6}$$

where SoC_k is the measured value, SoC_k^* is the estimated value, and l is the length of the time series.

III. AN ENSEMBLE OPTIMIZATION METHOD

In addition to the model design and feature selection, the optimization method is another crucial factor influencing the performance of the machine learning method. At present, the most commonly used optimization method is the Adam optimizer, which can achieve excellent performance by means of a weight updating rule based on adaptive learning rates derived from the first and second-order moment estimates [21]. However, it has two shortcomings that cannot be ignored. One is that the training process may not converge [22], and the other is that the globally optimal solution may be missed [23].

To combine the advantages of multiple optimization algorithms, a novel ensemble optimization method has been proposed based on the Adam and stochastic gradient descent (SGD) optimizers [24]. Although this optimization method achieves good performance, model training still takes a long time due to the slow optimization speed of the SGD algorithm. Consequently, a new ensemble optimization method based on the Nadam optimizer [25] and the AdaMax optimizer [21] is presented here.

A. NADAM OPTIMIZER

Incorporating the Nesterov momentum into the Adam optimizer yields a new algorithm, called the Nadam algorithm [25]. With the Nesterov momentum trick, the learning process during model training is accelerated by means of an exponential decay based on the moving average of the gradients. Compared with the Adam optimizer, the Nadam optimizer converges more rapidly and is more suitable for the pre-training phase. The details of the Nadam algorithm are as follows.

Algorithm 1 Nadam

Input: data samples
Output: weights and biases of the GRU-RNN model
1: Initialization $k \leftarrow 0$, $m_0 \leftarrow 0$, $n_0 \leftarrow 0$, $\tau \approx 10^{-8}$
2: $\beta_1, \beta_2 \in [0, 1)$, $\eta_1 = 0.001$
3: while ($L(w_k^s)$not converged) do
4: $k \leftarrow k + 1$
5: $g_k \leftarrow \nabla_w L(w_k^s)$ //obtain gradients at time step k
6: $\hat{g} \leftarrow g_k/(1 - \prod_{i=1}^{k} \beta_1^i)$ //correct g_k
7: $m_k \leftarrow \beta_1 m_{k-1} + (1 - \beta_1) g_k$ //the first moment estimate
8: $\hat{m}_k \leftarrow m_k/(1 - \prod_{i=1}^{k+1} \beta_1^i)$ //correct m_k
9: $n_k \leftarrow \beta_2 n_{k-1} + (1 - \beta_2) g_k^2$ //the second moment estimate
10: $\hat{n}_k \leftarrow n_k/(1 - \beta_2^k)$ //correct n_k
11: $w_k^s \leftarrow w_{k-1}^s - \frac{\eta_1}{\sqrt{\hat{n}_k + \tau}} (\beta_1^{k+1} \hat{m}_k + (1 - \beta_1^k) \hat{g}_k)$
12: end while

The weights are updated in accordance with the recursive law presented above in Algorithm 1. The parameters that need to be initialized are the time step k, the first-order moment estimation m_k, the second-order moment estimation n_k, the fuzz factor τ, the exponential decay rates β_1 and β_2, and the learning rate η_1. The momentum schedule is given by

$$\beta_1^k = \beta_1(1 - 0.5 \times 0.96^{k/250}) \tag{7}$$

where $\beta_1 = 0.99$, as the recommendation in [26]. Then, the main part of the algorithm is the iterative process. The first step is to calculate the gradient of the loss function $L(w_{k-1}^s)$, where w_{k-1}^s is the weight parameter, as shown in Eq. (6). The second step is to calculate the first-order moment estimatem_k and the second-order moment estimate n_k at time step k. After correction, the unbiased estimates $\hat{m}_k$ and $\hat{n}_k$ can be obtained. The last step is to update the weight parameters. This iterative process then continues until the optimal value is obtained.

Remark 1: The purpose of the pre-training phase is to endow the GRU_RNN model with the appropriate parameters to capture the inherent features of the training samples. The Nadam algorithm uses adaptive learning rates and approximates the gradient by means of the Nesterov momentum, thereby ensuring fast convergence of the pre-training process.

B. ADAMAX OPTIMIZER

An extension of the Adam optimizer, called AdaMax has been introduced in the literature [21]. In AdaMax, the infinity norm of the moment is used in place of the second-order moment estimate to update the weight parameters. Therefore, the magnitude of the parameter update has a simpler bound in AdaMax than in the Adam algorithm, and the weight updating rules are more stable. The details of the AdaMax optimizer are shown in Algorithm 2.

Algorithm 2 AdaMax

Input: data samples
Output: weights and biases of the GRU-RNN model
1: Initialization $k \leftarrow 0$, $m_0 \leftarrow 0$, $u_0 \leftarrow 0$, $\tau \approx 10^{-8}$
2: $\beta_1, \beta_2 \in [0, 1)$, $\eta_2 = 0.0005$
3: while ($L(w_k^s)$not converged) do
4: $k \leftarrow k + 1$
5: $g_k \leftarrow \nabla_w L(w_{k-1}^s)$ //obtain gradients at time step k
6: $m_k \leftarrow \beta_1 m_{k-1} + (1 - \beta_1) g_k$ //the first moment estimate
7: $u_k \leftarrow \max(\beta_2 u_{k-1}, |g_k|)$ //the weighted infinity norm
8: $w_k^s \leftarrow w_{k-1}^s - (\eta_2/(1 - \beta_1^k)) \cdot (m_k/u_k)$ //update parameters
9: end while

The weight updating process in Algorithm 2 is similar to that in Algorithm 1, except for the following differences: the term $(\eta_2/(1 - \beta_1^k))$ represents the learning rate with bias-correction for the first-order moment estimate, and u_k is the infinity norm of the moment.

Now, the pth-order moment estimate v_k is defined as follows:

$$v_k = \beta_2^p v_{k-1} + (1 - \beta_2^p)|g_k|^p \tag{8}$$

The iteration formula in (8) can be rewritten in the following form:

$$v_k = (1-\beta_2^p)\sum_{i=1}^{k}\beta_2^{p(k-i)}\cdot|g_i|^p \tag{9}$$

Let $p \to \infty$, and adopt the definition $u_k = \lim_{p\to\infty}(v_k)^{1/p}$; then,

$$\begin{aligned} u_k &= \lim_{p\to\infty}((1-\beta_2^p)\sum_{i=1}^{k}\beta_2^{p(k-i)}\cdot|g_i|^p)^{1/p} \\ &= \lim_{p\to\infty}(1-\beta_2^p)^{1/p}(\sum_{i=1}^{k}\beta_2^{p(k-i)}\cdot|g_i|^p)^{1/p} \\ &= \lim_{p\to\infty}(\sum_{i=1}^{k}(\beta_2^{(k-i)}\cdot|g_i|)^p)^{1/p} \\ &= \max(\beta_2^{k-1}|g_1|, \beta_2^{k-2}|g_2|, \ldots, \beta_2|g_{k-1}|, |g_k|) \end{aligned} \tag{10}$$

Because $\beta_2 \in [0, 1)$, $(1-\beta_2^p)^{1/p}$ goes to 1 when $p \to \infty$. Formula (10) can be derived from the definition of the infinity norm and then converted into the following recursive formula: $u_k = \max(\beta_2 \cdot u_{k-1}, |g_k|)$.

Remark 2: The purpose of the fine-tuning phase is to further adjust the parameters to achieve greater accuracy by means of the AdaMax algorithm, which converges to a more stable value.

C. ENSEMBLE OPTIMIZER

The proposed ensemble optimization method combines the Nadam and AdaMax optimizers. The Nadam optimizer, with its fast convergence speed, is used in the model pre-training phase to find the minimum optimized value as soon as possible. Then, the AdaMax optimizer is used in the model fine-tuning phase to further determine the model parameters. With the AdaMax optimizer, the phenomenon of gradient fluctuation during the model fine-tuning process can be reduced.

Algorithm 3 Ensemble Optimization Method

Input: data samples
Output: weights and biases of the GRU-RNN model
1: Initialize parameters
2: while epoch$< p_1$
3: if epoch$< p_2$ then
4: train model with Nadam //pre-training phase
5: else
6: train model with AdaMax //fine-tuning phase
7: end if
8: end while

The implementation of the ensemble optimization method includes the algorithm input and output, the initialization of the parameters, and the iterative process. During the iterative process, the ensemble optimization method switches the training mode from the pre-training phase to the fine-tuning phase based on the number of training epochs that have elapsed. The values of p_1 and p_2 are the numbers of epochs specified for the entire model training process and the pre-training phase, respectively.

Remark 3: Although the learning rate is adaptively adjusted in the Adam algorithm, during the later stage of model training, the learning rate changes very little, causing the gradient update to be too slow. By contrast, in the ensemble optimization method, the model training process is divided into two stages, one based on the Nadam optimizer and the other based on the AdaMax optimizer. In this way, not only can the model training be completed rapidly and stably, but the training time can also be reduced.

IV. ANALYSIS OF SOC ESTIMATION

This section presents the experimental analysis, including the data description, the model training, the model validation on dynamic driving cycles, and the performance comparison and results.

A. DATA DESCRIPTION

The data used for the experiment were collected from the Battery Research Group of the Center for Advanced Life Cycle Engineering (CALCE) [27]. The battery test bench adopted by the CALCE group consists of a 26650 LiFePO4/graphite cell, an Arbin BT2000 battery test system, and a computer serving as a user-machine interface. The method of charging used for battery testing is the constant current-constant voltage (CCCV) method; the standard charging and discharging current is 0.2C (A), and the range of cycle life lies between 1000 and 2000. Other parameters of the battery are shown in Table 1.

TABLE 1. Cell parameters.

Brand name	Battery weight	Nominal capacity	Nominal voltage	Charge/discharge cut-off voltage
A123	76 g	2.3 Ah	3.2 V	3.65 V, 2.0 V

Multiple factors affect SoC estimation because the loading conditions of EVs are complex and uncertain. The EV speed, road conditions, and energy feedback result in complicated power outputs. Hence, the training data set should cover the loading conditions of EVs as much as possible in terms of SoC range, current, voltage and temperature.

Therefore, to fully validate the model, three dynamic driving cycles were chosen to simulate the load characteristics of lithium-ion power batteries: the Federal Urban Driving Schedule (FUDS) is a driving cycle with a sophisticated dynamic current profile, the Dynamic Stress Test (DST) emulates a variable power discharge mode, and the US06 Highway Drive Schedule (US06) is an aggressive driving cycle for highway driving [28]–[30]. The durations of one cycle for FUDS, DST and US06 are 1372 s, 360 s, and 596 s, respectively.

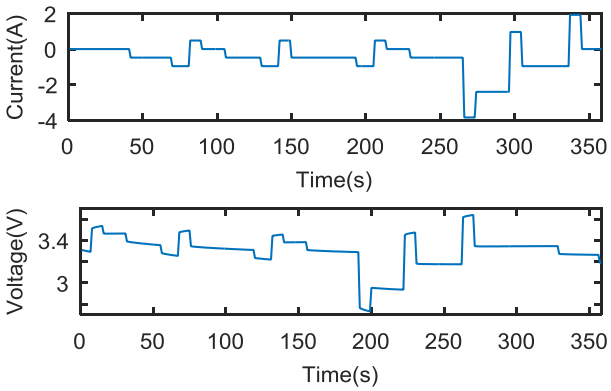


FIGURE 3. DST driving cycle: current and voltage profiles for one cycle.

The data set was acquired by simulating the FUDS, DST, and US06 driving cycles at 0-50°C (in intervals of 10°C), yielding a total of 18 data subsets and 126356 samples. The collected data include battery current, voltage and temperature data with a sampling interval of 1 s. In accordance with the cross-validation principle, 80% data set was selected as training set, and 20% data set was selected as test set. The current and voltage profiles for one DST cycle are shown in Fig. 3.

B. MODEL TRAINING

The experiment was implemented on a platforms running Windows 8.0 with an Intel i7 CPU, PyCharm 2018 [31], and Keras 2.2 [32]. PyCharm 2018 is a Python integrated development environment (IDE). Keras is a high-level NN API written in Python. Based on previous experience, the model hyperparameters were configured, as shown in Table 2. The order in which the data samples were fed into the GRU-RNN model was revised based on a random shuffling function. The model training process consisted of two main phases: the pre-training and the fine-tuning phase.

TABLE 2. Hyperparameters of the GRU-RNN model.

GRU unit	Batch size	Epoch	Optimizer	Learning rate	Dropout
260	230	100	Nadam, AdaMax	0.001, 0.0005	0.2

Data preprocessing was necessary to prevent the different dimensions of the raw data from affecting the model training process. Data normalization was applied to eliminate this effect, and improve the convergence speed during model training. Min-max normalization was applied to scale the feature data to the range between 0 and 1 using Eq. (11):

$$x' = (x - \min_x)/(\max_x - \min_x) \tag{11}$$

where x is a raw data value, x' is the scaled value, and $\max_x$ and $\min_x$ are the maximum and minimum data value, respectively.

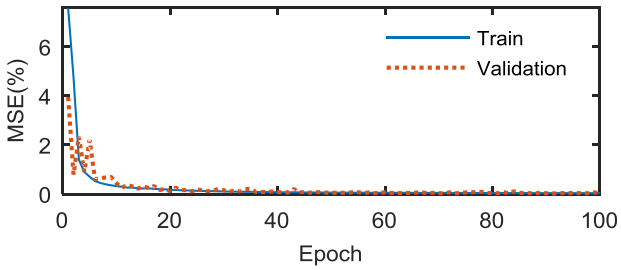


FIGURE 4. Convergence curves during model pre-training.

Fig. 4 presents the characteristic curves of the convergence process during pre-training. During the first 10 epochs, the training MSE quickly dropped from 7.58% to 0.32%. After approximately 20 epochs, the training MSE had converged almost to zero. In the final epoch of pre-training, the training and validation MSE values approached 3.68×10^{-4} and 3.77×10^{-4}, respectively; this required approximately 1.5 h.

The Nadam optimizer successfully found the minimum optimized value during pre-training. Throughout the 20th to 100th epochs, the value of the MSE showed almost no change, and the convergence curves were very stable. If we had continued training for more epochs, it is possible that the optimization process could have become trapped in a local minimum and required additional time; this can be easily verified through simulations. In fact, the Nadam optimizer is still, in essence, the Adam optimizer; it uses the Newtonian momentum to accelerate the convergence speed.

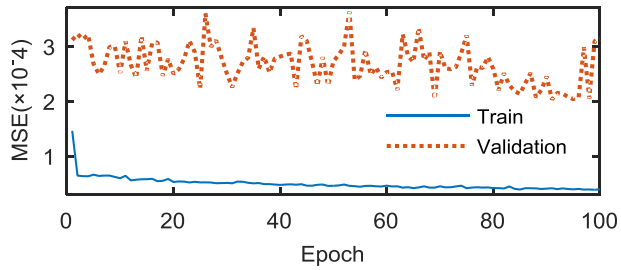


FIGURE 5. Convergence curves during model fine-tuning.

Fig. 5 shows the convergence curves during model fine-tuning. Even when the training error is small, fine-tuning can achieve further optimization. During the first 5 epochs, the MSE again showed a rapid decline. After 100 epochs of fine-tuning, the training and validation MSE values reached 3.94×10^{-5} and 3.13×10^{-4}, respectively; this required approximately 1.6 h.

As shown in Fig. 5, both the training and validation MSEs steadily converged to very small values. These results illustrate that the globally optimal value was ultimately found, and that on over-fitting or under-fitting occurred during model training. As a result, faster convergence and more efficient optimization were achieved for the model training process as a whole.

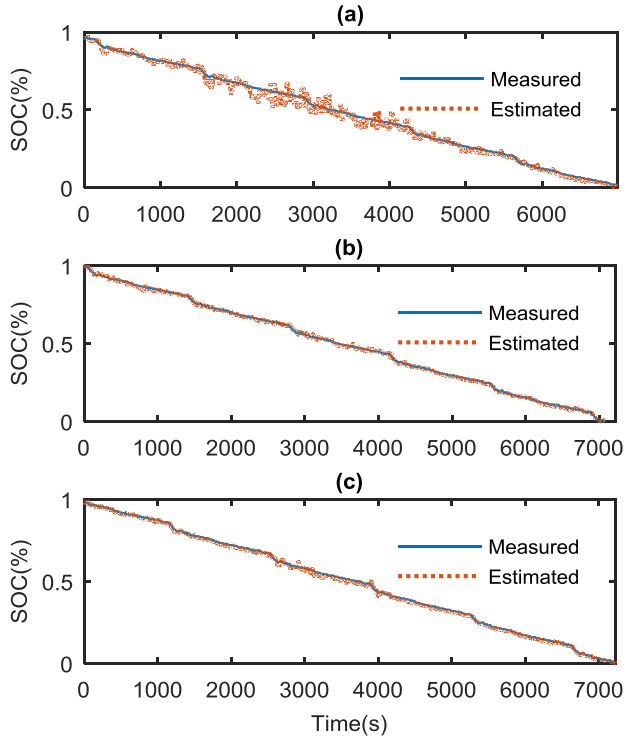


FIGURE 6. SoC estimation results for FUDS cycle. (a) 0°C, (b) 30°C, and (c) 50°C.

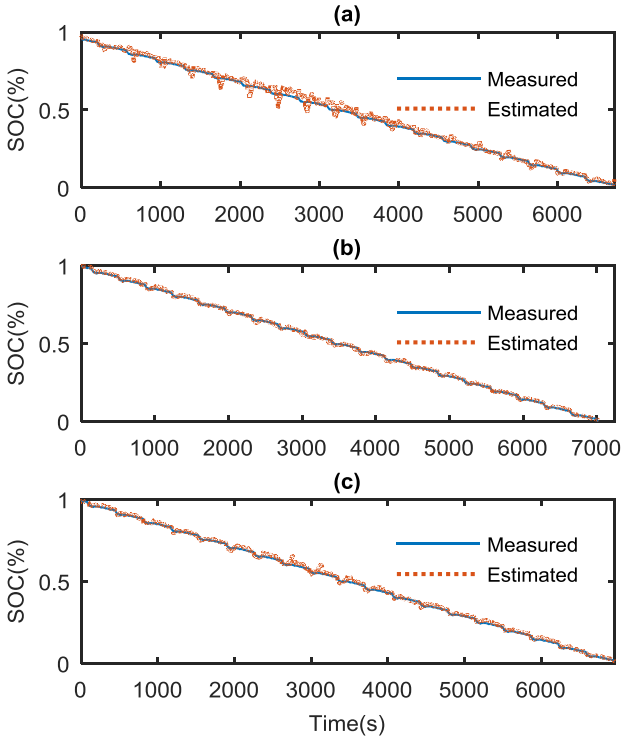


FIGURE 7. SoC estimation results for DST cycle. (a) 0°C, (b) 30°C, and (c) 50°C.

C. MODEL VALIDATION

To verify its robustness and accuracy, the trained GRU-RNN model for battery SoC estimation was separately tested on the FUDS, DST, and US06 cycle at 0°C, 30°C, and 50°C. All the data prepared for testing were distinct from the training sets. The battery SoC computed with the ampere-hour method [2] was taken as the actual SoC for use as a reference. The reference SoC and the SoC estimated with the proposed method were standardized to the range from 100% to 0%. The root mean square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), coefficient of determination (R^2), and SoC error, as defined in the following formulas, were employed as the performance metrics used to evaluate the SoC estimation model.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(SoC_k - SoC_k^*)^2} \tag{12}$$

$$MAE = \frac{1}{N}\sum_{k=1}^{N}|SoC_k - SoC_k^*| \tag{13}$$

$$MAPE = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{SoC_k - SoC_k^*}{SoC_k}\right| \tag{14}$$

$$R^2 = 1 - \frac{\sum_{k=1}^{N}(SoC_k - SoC_k^*)^2}{\sum_{k=1}^{N}(SoC_k - M_{soc})^2} \tag{15}$$

$$Error = \frac{SoC_k - SoC_k^*}{SoC_k} \tag{16}$$

where SoC_k is the actual value, SoC_k^* is the estimated value, M_{soc} is the mean SoC, and N is the number of test samples.

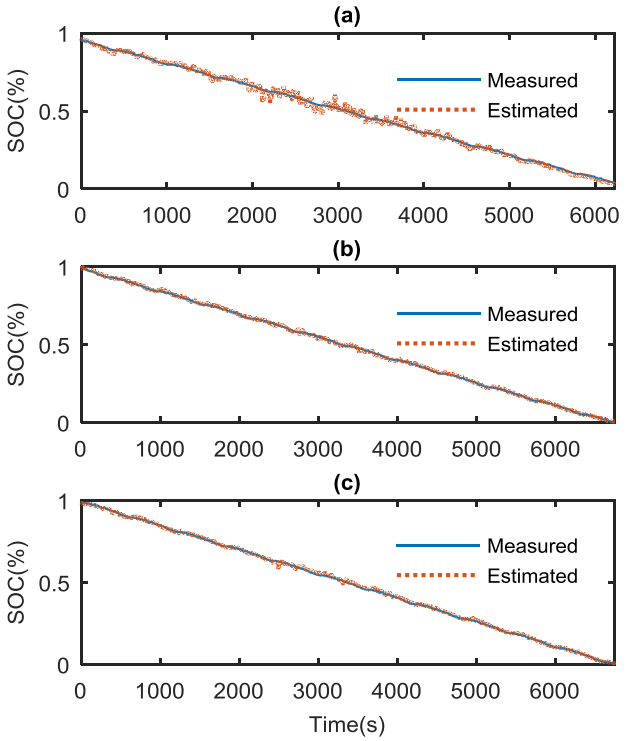


FIGURE 8. SoC estimation results for US06 cycle. (a) 0°C, (b) 30°C, and (c) 50°C.

Fig. 6-8 show the SoC estimation results. Obviously, the SoC values estimated by the GRU-RNN model on the FUDS, DST and US06 data sets at 30°C and 50°C are in nearly

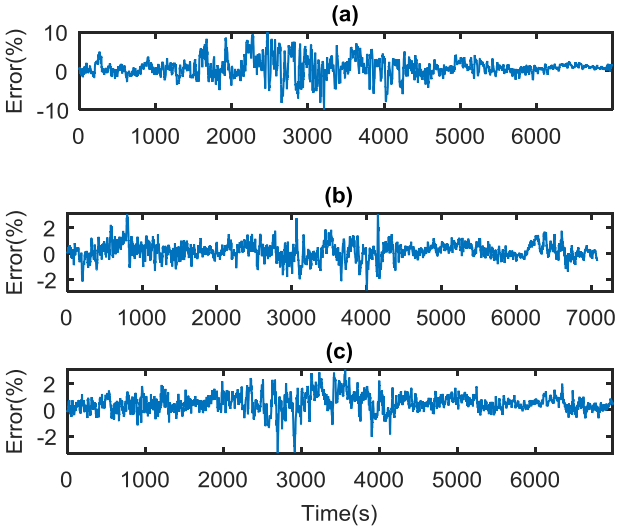


FIGURE 9. SoC estimation error for FUDS cycle. (a) 0°C, (b) 30°C, and (c) 50°C.

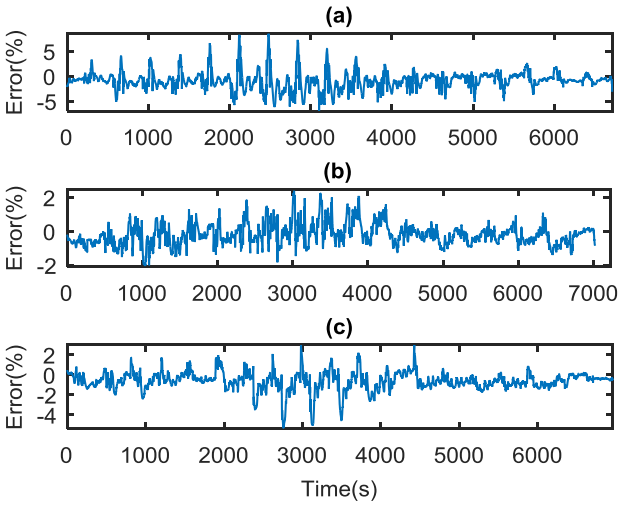


FIGURE 10. SoC estimation error for DST cycle (a) 0°C, (b) 30°C, and (c) 50°C.

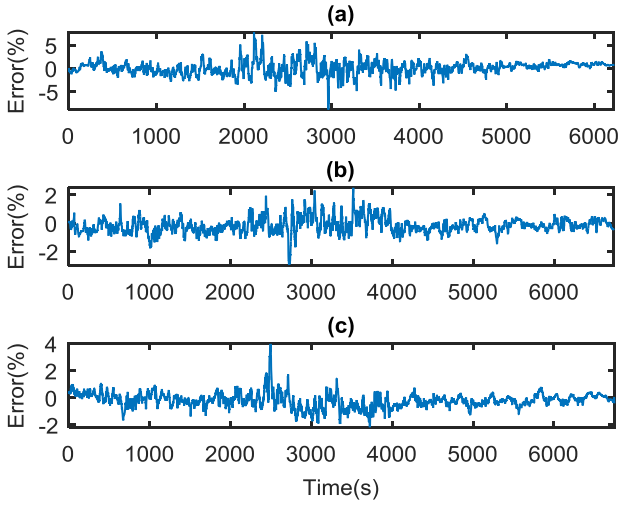


FIGURE 11. SoC estimation error for US06 cycle. (a) 0°C, (b) 30°C, and (c) 50°C.

TABLE 3. Statistical results of SoC estimation in FUDS, DST, and US06.

Test case	RMSE	MAE	MAPE	R^2 (%)	Error (%)
FUDS 0°C	2.45%	1.77%	6.19%	99.20	[-9.93 9.96]
FUDS 30°C	0.64%	0.49%	3.15%	99.95	[-2.81 3.05]
FUDS 50°C	0.83%	0.66%	2.46%	99.92	[-3.29 3.07]
DST 0°C	1.97%	1.44%	4.77%	99.48	[-6.91 8.52]
DST 30°C	0.64%	0.50%	2.35%	99.95	[-2.04 2.40]
DST 50°C	1.12%	0.83%	2.79%	99.84	[-5.30 2.99]
US06 0°C	1.40%	1.04%	3.20%	99.72	[-8.82 7.74]
US06 30°C	0.55%	0.43%	3.17%	99.96	[-2.92 2.50]
US06 50°C	0.57%	0.42%	1.51%	99.96	[-2.17 3.98]

aligned with the actual SoC, but those for the data sets at 0°C are not. Moreover, in Fig. 9-11, the SoC estimation errors are almost all close to zero, with the minimum and maximum values being −2.04% and 9.96%, respectively, while the SoC error range in Fig. 9(a) is [−9.93% 9.96%]. The values of the RMSE, MAE, MAPE, R^2 and Error performance metrics are shown in Table 3; they are close to the desired optimal values of 0, 0, 0, 1, 0, respectively.

As shown in Table 3, the best test results are obtained for the US06 cycle at 50°C. The R^2 metric in this case is the best among the three tests, indicating that the SoC estimation curve for this data set shows the best fit. Therefore, the metrics for this data set are naturally the best. By contrast, the worst test results are obtained for the FUDS cycle at 0°C. Normally, a decrease in temperature causes an increase in the internal resistance of the battery. As the temperature rises, the electrolyte becomes active, promoting lithium-ion diffusion and migration [33]. Therefore, the battery SoC is unavoidably affected by temperature.

The battery SoC varies at different charging and discharging current rates. A high discharge current rate causes a rapid decrease in the SoC, and this decrease shows a strong relationship with the voltage since an increase in the SoC causes the voltage to rise. The SoC estimation results for the DST cycle are more stable than those for the FUDS cycle because the DST current profile is more stable and exhibits less fluctuation than that the FUDS profile [28], [30]. Therefore, due to the corresponding temperature and load characteristics, the test results for the FUDS cycle at 0°C are the worst.

These test results confirm the dynamic tracking performance of the GRU-RNN model. Although this method suffers a slight loss in accuracy on the data set at 0°C, the estimation errors are still satisfactory. As shown in table 3, accurate estimation has been achieved on FUDS, DST, and US06 cycle at 0°C, 30°C, and 50°C, indicating that the trained model has good robustness and generalization.

TABLE 4. Model performance comparison in FUDS cycle.

Method	Temperature	RMSE	MAE	MAPE	Error (%)
LRA	0°C	2.53%	1.87%	5.79%	[-7.03 11.0]
	30°C	1.05%	0.81%	3.43%	[-2.13 5.02]
	50°C	0.97%	0.78%	3.38%	[-3.90 4.80]
LRE	0°C	2.18%	1.58%	5.87%	[-4.40 9.60]
	30°C	0.66%	0.50%	2.38%	[-2.28 2.85]
	50°C	0.83%	0.67%	2.70%	[-3.03 3.84]
GRA	0°C	2.31%	1.70%	6.28%	[-5.62 9.44]
	30°C	1.0%	0.70%	3.03%	[-4.42 2.33]
	50°C	1.01%	0.78%	3.0%	[-4.35 3.77]
GRE	0°C	2.45%	1.77%	6.19%	[-9.93 9.96]
	30°C	0.64%	0.49%	3.15%	[-2.81 3.05]
	50°C	0.83%	0.66%	2.46%	[-3.29 3.07]

TABLE 5. Model performance comparison in DST cycle.

Method	Temperature	RMSE	MAE	MAPE	Error (%)
LRA	0°C	3.39%	2.64%	6.29%	[-10.3 2.09]
	30°C	3.05%	2.25%	5.76%	[-9.12 2.11]
	50°C	4.73%	3.43%	7.39%	[-15.9 1.97]
LRE	0°C	1.84%	1.36%	4.55%	[-7.84 3.77]
	30°C	0.89%	0.74%	3.0%	[-2.73 2.18]
	50°C	1.23%	0.96%	3.32%	[-5.04 1.24]
GRA	0°C	4.79%	3.29%	11.12%	[-23.5 6.81]
	30°C	1.66%	1.40%	8.18%	[-5.16 7.38]
	50°C	2.26%	1.72%	5.0%	[-8.43 10.7]
GRE	0°C	1.97%	1.44%	4.77%	[-6.91 8.52]
	30°C	0.64%	0.50%	2.35%	[-2.04 2.40]
	50°C	1.12%	0.83%	2.79%	[-5.30 2.99]

D. PERFORMANCE COMPARISON AND RESULTS

To verify the performance of the GRU-RNN model and the ensemble optimization method, four methods were designed and tested in a comparative experiment, including an LSTM-based RNN model optimized by Adam (LSTM-RNN-Adam, LRA) [16], an LSTM-based RNN model optimized by the ensemble optimizer (LSTM-RNN-Ensemble, LRE), a GRU-based RNN model optimized by Adam (GRU-RNN-Adam, GRA), and a GRU-based RNN model optimized by the ensemble optimizer (GRU-RNN-Ensemble, GRE, the proposed method). The test results obtained under the same conditions with these four methods after 200 epochs are shown in Table 4-6. The times required for model training with the Intel i7 CPU were 3.62 h, 3.65 h, 2.95 h, and 2.97 h, respectively.

Model design is a crucial factor influencing the performance of machine learning methods. In the following, comparative analyses of the LRA and GRA methods, the LRE and GRE methods are presented. The RMSE, MAE, and MAPE values for the GRA method tested on the FUDS cycle at 50°C are 1.01%, 0.78%, and 3.0%, respectively, which are −4.12%, 0, and 11.24% lower than the corresponding values for the LRA method. Similarly, the RMSE, MAE, and MAPE values for the GRE method tested on the FUDS cycle at 50°C are 0.83%, 0.66%, and 2.46%, respectively, which are 0, 1.49%, and 8.89% lower than the corresponding values for the LRE method. The LRA and GRA methods were implemented with LSTM-RNN and GRU-RNN model, respectively; although both models were optimized by the Adam optimizer, the GRA method performed better. Similarly, the LRE and GRE methods were also implemented with LSTM-RNN and GRU-RNN model, respectively, and althouth both were optimized by the ensemble optimizer, the GRE method performed better. Thus, for the same optimizer, the GRU-RNN model is superior to the LSTM-RNN model for battery SoC estimation.

TABLE 6. Model performance comparison in US06 cycle.

Method	Temperature	RMSE	MAE	MAPE	Error (%)
LRA	0°C	1.47%	1.14%	3.97%	[-5.10 6.43]
	30°C	0.86%	0.63%	3.99%	[-2.82 4.99]
	50°C	1.01%	0.73%	2.06%	[-3.41 4.61]
LRE	0°C	1.68%	1.19%	3.09%	[-7.59 7.82]
	30°C	0.80%	0.61%	3.84%	[-3.31 1.38]
	50°C	1.03%	0.80%	2.43%	[-3.36 3.24]
GRA	0°C	1.67%	1.21%	3.44%	[-8.26 7.35]
	30°C	1.30%	1.05%	7.36%	[-3.98 2.26]
	50°C	1.50%	1.17%	4.23%	[-6.08 3.50]
GRE	0°C	1.40%	1.04%	3.20%	[-8.82 7.74]
	30°C	0.55%	0.43%	3.17%	[-2.92 2.50]
	50°C	0.57%	0.42%	1.51%	[-2.17 3.98]

The optimization algorithm is another crucial factor influencing the performance of machine learning methods. In the following, comparative analyses of the LRA and LRE methods, the GRA and GRE methods are presented. The RMSE, MAE, and MAPE values for the LRE method tested on the FUDS cycle at 50°C are 0.83%, 0.67%, and 2.7%, respectively; compared to those for the LRA method, these values are reduced by 14.43%, 14.10%, and 20.12%, respectively. Similarly, the RMSE, MAE, and MAPE values for the GRE method tested on the FUDS cycle at 50°C are 0.83%, 0.66%, and 2.46%, respectively, and compared to those for the GRA method, these values are reduced by 17.82%, 15.38%, and 18.0%, respectively. The LRA and LRE methods were both implemented with the LSTM-RNN model, which was optimized by the Adam and ensemble optimizer, respectively; between the two, the LRE method performed better. Similarly, the GRA and GRE methods were both implemented with the GRU-RNN model, which was optimized by the Adam and ensemble optimizer, respectively, and the GRE method performed better. Obviously, both LSTM-RNN and GRU-RNN models optimized by the ensemble optimizer performed better than the same models optimized by the Adam optimizer.

Furthermore, the RMSE, MAE, and MAPE values obtained on the FUDS cycle at 0°C and 30°C show reductions similar to those observed on the FUDS cycle at 50°C. Let us further evaluate the performance of each method based on the SoC errors. Compared to the other three methods, the proposed GRE method achieves the most satisfactory SoC error range. The model performance comparison in DST and US06 cycle are shown in Table 5 and 6.

TABLE 7. Average of metrics for SoC estimation in FUDS, DST, and US06.

Method	LRA	LRE	GRA	GRE
RMSE	2.12%	1.24%	1.94%	1.13%
MAE	1.58%	0.94%	1.45%	0.84%
MAPE	4.67%	3.46%	5.74%	3.28%

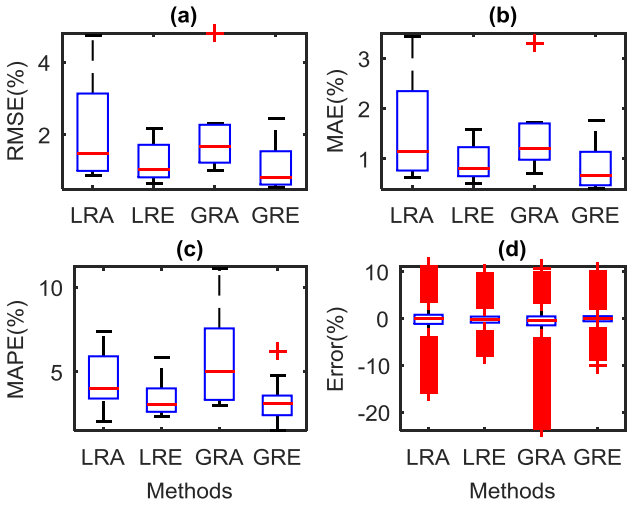


FIGURE 12. RMSE, MAE, MAPE, and Error for the LRA, LRE, GRA, and GRE methods.

Finally, to better understand the estimation results, we present statistical results of average value for performance metrics in Table 7 and error distribution charts for each of the four methods tested on all test data sets (FUDS, DST, and US06) in Fig. 12. Obviously, the results of the LRA method are the worst, while the results of the GRE method are the best. However, the results of the LRE and GRE methods are not much different, indicating that the performances that can be achieved with the LSTM and GRU can be similarly improved through appropriate optimization [34]. In addition, hundreds of outliers appear in the SoC error graph in Fig. 12(d). Because the test sets consisted of dynamic driving cycle data that were not filtered before being fed into the model and the total number of test sets was approximately 60,000, it was difficult to avoid a small number of outliers.

In a word, the comprehensive performance of the GRE method is obviously better than that of the LRA method. Because LSTM in the LRA method has three gates, while GRU in the GRE method has two gates. The input gate and forget gate of LSTM are fused into the update gate, and the reset gate is applied directly to the previous hidden state. The update gate determines how much the previously stored information should be kept. The reset gate is concerned with the way for the new input and the previously stored information should be integrated [18]–[19]. Thus, GRU has simpler structure, and can be trained easier or require less sample data. The SoC estimation approaches in the review are roughly classified into three categories: the traditional estimation methods, the control-theory-based methods, and the data-driven methods. The GRE belongs to the data-driven methods. Compared with the traditional estimation methods, the GRE method can achieve high estimation accuracy and adapt to different ambient temperatures; Compared with the control-theory-based methods, the GRE method only needs sample data for model training, does not require battery modeling, parameter identification, and the complex proof of Lyapunov stability.

V. CONCLUSIONS

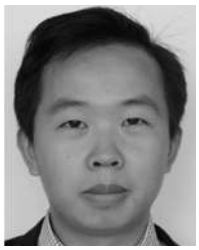
In this paper, we improved the LRA method in two respects. On the one hand, the GRU component was chosen instead of the LSTM component for the hidden layer units of the RNN model. Compared with an LSTM-RNN model, a GRU-RNN model has a simpler structure and fewer parameters, thus reducing the difficulty of model training. On the other hand, an ensemble optimization algorithm combining the Nadam and AdaMax optimizers was proposed to obtain the globally optimal values efficiently and rapidly, improving the optimization efficiency and shortening the training time for the battery model.

Based on this study, the following conclusions can be drawn. First, the GRU component is more suitable than the LSTM component for use in an RNN model for battery SoC estimate. Second, for a GRU-RNN battery model, it is better to use the proposed ensemble optimization method combining the Nadam and AdaMax optimizers than to use the Adam optimizer. Finally, a new method of battery SoC estimation called the GRE method is presented, which requires only data samples for model training and can accurately and efficiently estimate the battery SoC. From Table 7, it is easy to find that the mean values of the RMSE, MAE and MAPE metrics for the GRE method tested on all test data sets (FUDS, DST and US06) are 1.13%, 0.84% and 3.28%, respectively, which are 46.7%, 46.84% and 29.76% lower than the corresponding values for the LRA method. Moreover, the time needed for model training with an Intel i7 CPU is 2.97 h for the GRE method, meaning that its time consumption is 17.96% lower than that of the LRA method. Thus, it is obvious the GRE method proposed in this paper demonstrates superior comprehensive performance.

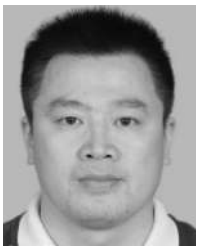
REFERENCES

[1] R. Xiong, J. Cao, Q. Yu, H. He, and F. Sun, "Critical review on the battery state of charge estimation methods for electric vehicles," *IEEE Access*, vol. 6, pp. 1832–1843, 2017.

[2] N. Yang, X. Zhang, and G. Li, "State of charge estimation for pulse discharge of a $LiFePO_4$ battery by a revised Ah counting," *Electrochim. Acta*, vol. 151, pp. 63–71, Jan. 2015.
[3] S. Lee, J. Kim, J. Lee, and B. H. Cho, "State-of-charge and capacity estimation of lithium-ion battery using a new open-circuit voltage versus state-of-charge," *J. Power Sources*, vol. 185, no. 2, pp. 1367–1373, Dec. 2008.
[4] J. Xu, C. C. Mi, B. Cao, and J. Cao, "A new method to estimate the state of charge of lithium-ion batteries based on the battery impedance model," *J. Power Sources*, vol. 233, pp. 277–284, Jul. 2013.
[5] X. Hu, S. Li, and H. Peng, "A comparative study of equivalent circuit models for Li-ion batteries," *J. Power Sources*, vol. 198, pp. 359–367, Jan. 2012.
[6] K. A. Smith, C. D. Rahn, and C.-Y. Wang, "Model-based electrochemical estimation and constraint management for pulse operation of lithium ion batteries," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 654–663, May 2010.
[7] B. Ning, B. Cao, B. Wang, and Z. Zou, "Adaptive sliding mode observers for lithium-ion battery state estimation based on parameters identified online," *Energy*, vol. 153, pp. 732–742, Jun. 2018.
[8] B. Wang, Z. Liu, S. E. Li, S. J. Moura, and H. Peng, "State-of-charge estimation for lithium-ion batteries based on a nonlinear fractional model," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 3–11, Jan. 2017.
[9] C.-Z. Liu *et al.*, "A state of charge estimation method based on H_∞ observer for switched systems of lithium-ion nickel–manganese–cobalt batteries," *IEEE Trans. Ind. Electron.*, vol. 64, no. 10, pp. 8128–8137, Oct. 2017.
[10] Y. Zheng, W. Gao, M. Ouyang, L. Lu, L. Zhou, and X. Han, "State-of-charge inconsistency estimation of lithium-ion battery pack using mean-difference model and extended Kalman filter," *J. Power Sources*, vol. 383, pp. 50–58, Apr. 2018.
[11] Q. Q. Yu, R. Xiong, C. Lin, W. X. Shen, and J. J. Deng, "Lithium-ion battery parameters and state-of-charge joint estimation based on H-infinity and unscented Kalman filters," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 8693–8701, Oct. 2017.
[12] S. Schwunk, N. Armbruster, S. Straub, J. Kehl, and M. Vetter, "Particle filter for state of charge and state of health estimation for lithium–iron phosphate batteries," *J. Power Sources*, vol. 239, pp. 705–710, Oct. 2013.
[13] J. Meng, G. Luo, and F. Gao, "Lithium polymer battery state-of-charge estimation based on adaptive unscented Kalman filter and support vector machine," *IEEE Trans. Power Electron.*, vol. 31, no. 3, pp. 2226–2238, Mar. 2016.
[14] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, "State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach," *J. Power Sources*, vol. 400, pp. 242–255, Oct. 2018.
[15] C. Jian, O. Quan, C. Xu, and H. Su, "Neural network-based state of charge observer design for lithium-ion batteries," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 1, pp. 313–320, Jan. 2018.
[16] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of Li-ion batteries," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6730–6739, Aug. 2018.
[17] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[19] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. 8th Workshop Syntax, Semantics Struct. Stat. Transl. (SSST@EMNLP)*, Doha, Qatar, Oct. 2014, pp. 103–111.
[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, no. 15, pp. 1929–1958, Jun. 2014.
[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.
[22] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018, pp. 1–23.
[23] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017, pp. 1–11.
[24] N. S. Keskar and R. Socher. (Dec. 2017). "Improving generalization performance by switching from adam to SGD." [Online]. Available: https://arxiv.org/abs/1712.07628
[25] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, 2016, pp. 1–4.
[26] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 2013, pp. 1139–1147.
[27] CALCE. *Lithium-Ion Battery Experimental Data*. Accessed: Jan. 5, 2017. [Online]. Available: http://www.calce.umd.edu/batteries/data.htm
[28] *USABC Electric Vehicle Battery Test Procedures Manual*, U.S. Adv. Battery Consortium, Southfield, MI, USA, 2006.
[29] *EPA US06 or Supplemental Federal Test Procedure (SFTP)*. [Online]. Available: https://www.dieselnet.com/standards/cycles/ftp_us06.php
[30] EG and G. Idaho, "Simplified version of the federal urban driving schedule for electric vehicle battery testing," Tech. Inf. Center, Oak Ridge, TN, USA, Tech. Rep., Aug. 1988, no. 10.
[31] (2018). *Pycharm*. [Online]. Available: https://www.jetbrains.com/pycharm
[32] *Keras 2.2*. [Online]. Available: https://keras.io
[33] D. H. Doughty, E. P. Roth, C. C. Crafts, G. Nagasubramanian, G. Henriksen, and K. Amine, "Effects of additives on thermal stability of Li ion cells," *J. Power Sources*, vol. 146, nos. 1–2, pp. 116–120, 2005.
[34] J. Y. Chung, C. Gulcehre, K. H. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Deep Learn. Represent. Learn. Workshop*, Montreal, QC, Canada, Dec. 2014, pp. 1–9.



BIN XIAO received the M.S. degree in computer application technology from the Hunan University of Science and Technology. He is currently pursuing the Ph.D. degree in control theory and control engineering with the School of Automation Science and Engineering, South China University of Technology. His current research interests include power lithium battery state estimation and intelligent detection technology.



YONGGUI LIU was born in Hunan, China, in 1978. He received the B.S. degree in electronic information engineering from the Hunan University of Technology, Zhuzhou, China, in 2001, and the M.S. and Ph.D. degrees from the South China University of Technology, Guangzhou, China, in 2008 and 2011, respectively. From 2012 to 2014, he was a Postdoctoral Fellow with the Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China. He is currently an Associate Professor with the School of Automation Science and Engineering, South China University of Technology. His research interests include autonomous vehicle control, cooperative control, networked control systems, wireless sensor/actuator networks, and estimation theory and application.

BING XIAO received the B.S., M.S., and Ph.D. degrees in automation control engineering from the South China University of Technology, Guangzhou, China, in 1987, 1990, and 2002, respectively. He is currently a Professor with the School of Automation Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include new energy vehicle electronic control systems, pattern recognition and car advanced assisted driving systems, and sensor technology and intelligent detection technology.