Journal of Computer Science and Artificial Intelligence
ISSN: 3078-8242 (Print), ISSN: 3078-2309 (Online) | Vol. 2, No. 1, 2025

Battery State of Charge Estimation Based on Improved Neural Network

Hong Zhang *

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, 454000, China

* **Corresponding author**: Hong Zhang (Email: zhanghonghpu@126.com)

Abstract: Accurate prediction of battery state of charge (SOC) is vital for extending battery lifespan and ensuring operational safety in electric vehicles. However, traditional approaches exhibit constrained predictive accuracy and robustness under complex environments and variable operating conditions. Therefore, this study proposes the AO-BiGRU-CAM model, integrating Aquila Optimization (AO), a bidirectional Gated Recurrent Units (BiGRU), and a channel attention mechanism (CAM). It refines parameter search while enhancing focus on salient features to improve training efficiency and predictive accuracy. Under diverse temperatures and drive cycles, AO-BiGRU-CAM reduces errors by approximately 5%–10% in MSE, RMSLE, and MAPE compared with other models, demonstrating robust adaptability.

Keywords: State of charge, Aquila Optimization, Deep learning.

1. Introduction

As global environmental challenges intensify, vehicle electrification has emerged as a vital strategy to reduce greenhouse gas emissions and alleviate energy crises. Electric vehicles (EVs) are steadily transforming the transportation sector due to their zero emissions and high energy efficiency [1]. However, the widespread adoption of EVs heavily depends on advanced battery technologies, particularly lithium-ion batteries (Li-ion Batteries) [2]. Li-ion batteries, with their high energy density, low self-discharge rate, long lifespan, and eco-friendly characteristics, have become the primary power source for Evs [3].

During EVs operation, the state of charge (SOC) serves as the key parameter for remaining battery capacity, and its accurate estimation is crucial for longevity, efficiency, and safety [4]. Nevertheless, precise SOC prediction encounters numerous obstacles [5]. The electrochemical properties of Li-ion batteries are complex, featuring significant nonlinearity and time variance, which hinders direct sensor-based SOC measurement [6]. Consequently, voltage, current, and temperature measurements must be used for indirect SOC estimation. In addition, battery aging, temperature fluctuations, and cycling effects add further complexity to SOC estimation.

Currently, the SOC estimation methods can be classified into open-circuit voltage, coulomb counting, model-based, and data-driven approaches [7, 8].

The open-circuit voltage method estimates SOC by mapping the resting voltage to the battery's SOC. This method is straightforward and accurate, yet it requires resting periods before measurement and thus cannot enable real-time estimation, limiting its practical utility [9].

Coulomb counting relies on the principle of charge conservation and integrates current measurements to estimate SOC. Although theoretically feasible, uncertainties in internal resistance and initial SOC introduce errors, and the accuracy of current sensors directly influences estimation precision. Over extended charge–discharge cycles, error accumulation worsens and undermines the practical accuracy of coulomb counting [10].

Model-based approaches encompass equivalent circuit model (ECM) and electrochemical model (EM). The ECM simulates the electrical behavior for real-time SOC estimation but lacks detailed electrochemical insights and often requires frequent parameter adjustments under varying conditions [11]. In contrast, EM simulate internal electrochemical reactions, mass transport, and heat conduction, offering detailed behavioral insights but demanding extensive computations and hard-to-obtain parameters, thus serving mainly design and performance evaluations [12].

Data-driven approaches rely on machine learning algorithms and extensive historical data to predict SOC by learning parameter variations during battery charging and discharging. This method possesses strong adaptability, enabling it to accommodate diverse operating conditions and battery types while supporting online updates [13-15]. Traditional machine learning techniques, such as Multiple Linear Regression (MLR), Support Vector Machines (SVM), Random Forest (RF), and Gaussian Process Regression (GPR), perform effectively with low-dimensional data and linear relationships. However, they exhibit constrained efficacy when confronted with high-dimensional and complex nonlinear scenarios [16-22].

In recent years, researchers have extensively investigated various deep learning architectures, including Recurrent Neural Network (RNN) [23], Long Short-Term Memory (LSTM) [24], Gated Recurrent Units (GRU) [25], and TCN [26], to enhance SOC estimation accuracy. These methods have partially improved SOC estimation precision. However, numerous challenges and limitations persist for different models under practical conditions. For instance, Meng Jiao et al. [27] proposed a GRU-RNN–based SOC model optimized by momentum gradient, yet heavy reliance on parameter tuning heightened the training complexity. Jichao Hong et al. [28] developed an LSTM-based SOC model incorporating various influencing factors for long-term forecasting. Although theoretically robust, its generalization in real-world settings remains unverified due to insufficient large-scale experimental data. Bei Yan et al. [29] employed a CNN-BiLSTM model with knowledge constraints to bolster robustness, yet limited comparative experiments obstruct a

comprehensive demonstration of its superiority. Kuo Yang et al. [30] introduced a TCN-GRU model that captures time-series features effectively, improving SOC accuracy but incurring large parameter sizes and complex training.

Despite the strong nonlinear modeling capacity of GRU-RNN, and LSTM methods, many obstacles remain in real-world implementations. Deep learning models generally demand extensive hyperparameter tuning to achieve satisfactory SOC estimation performance. Many deep learning approaches lack large-scale, diverse datasets, thereby limiting their generalization in complex operational contexts. Existing models must further improve robustness and adaptability when exposed to varied operating conditions and battery types.

To address these issues, this study proposes an AO-BiGRU-CAM model that integrates Archimedes Optimization (AO) [31], bidirectional GRU, and a channel attention mechanism (CAM) [32]. The AO algorithm autonomously optimizes model parameters, greatly reducing dependency on manual tuning. The bidirectional GRU captures two-way dynamics of battery states, while CAM heightens discrimination among different feature importances, thereby boosting SOC estimation accuracy and robustness. Introducing the AO-BiGRU-CAM model not only advances SOC prediction accuracy but also offers novel technical pathways and theoretical support for battery management system (BMS) optimization.

2. AO-BiGRU-CAM Model for SOC Estimation

This study proposes an SOC estimation model called AO-BiGRU-CAM. It integrates AO, a bidirectional gated recurrent unit (BiGRU), and a CAM. This approach enhances SOC prediction accuracy and robustness from multiple dimensions, offering new insights for BMS optimization. Figure 1 presents the model structure.

2.1. BiGRU

GRU were proposed by Cho et al. in 2014 to address short- and long-term dependencies in machine translation and time series prediction. GRU improve upon LSTM networks by addressing the same issues of limited memory and vanishing gradients found in conventional neural networks. However, GRU replace the memory cell and forget gate with an update gate, considerably accelerating training and mitigating LSTM's long-term dependency issues. They also reduce parameter counts and alleviate the computational burden caused by the forget and output gates in LSTM.

GRU employ two gates (update and reset) to regulate information flow, thereby achieving LSTM-like memory capabilities while retaining a more streamlined architecture. On this basis, BiGRU integrate forward and backward streams to simultaneously capture forward and backward dependencies within sequences. For instance, in time-series forecasting or natural language processing, subsequent inputs (future time steps or later sentence segments) can benefit the current representation. BiGRUs can thus offer a more holistic representation.

BiGRU comprise two GRU layers: one processes forward timesteps, and the other processes backward timesteps. This structure yields two distinct hidden-state sequences, each representing forward or backward information at each time point. At each time step, BiGRU form the final output by merging the hidden states from both directions. The structure is shown in Figure 2. For a BiGRU, given each time-step.

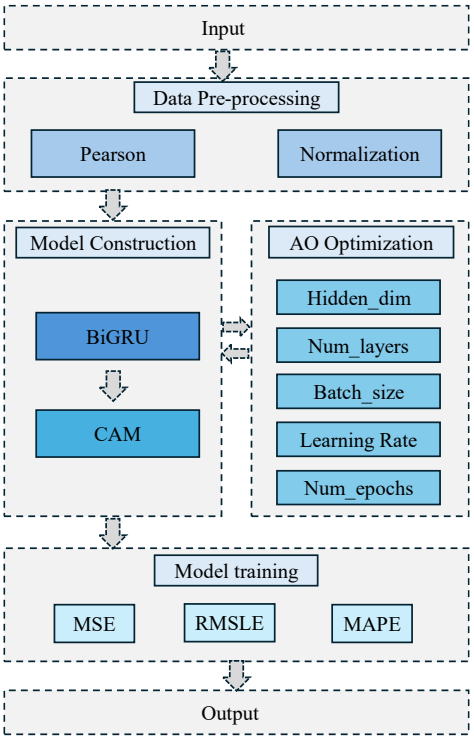

Figure 1. Model Structure

Input vector and the previous hidden state, the BiGRU updates its hidden state through the following gating mechanisms:

The update gate assesses the significance of the previous output and decides whether to retain it for the next computation. A larger update gate value amplifies the influence of the previous state on the current computation. The update gate is computed as follows:

$$Q_t = \sigma(W_{(x,Q)} \cdot [u_{t-1}, x_t] + b_Q) \quad (1)$$

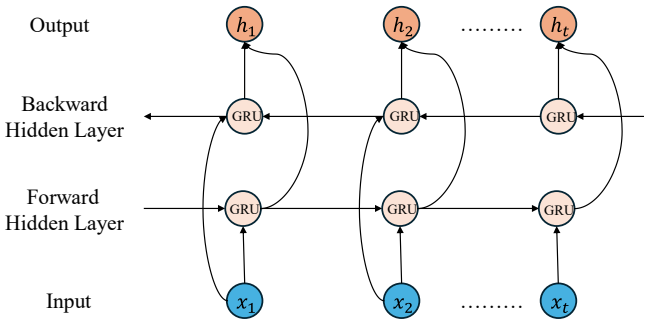

Figure 2. BiGRU

Here, W denotes the weight, u_{t-1} is the previous state, x_t is the current input, and b_Q is the bias vector.

The reset gate regulates how much current and previous states integrate, with smaller values implying greater forgetting and larger values indicating more information retention. The reset gate is computed as follows:

$$R_t = \sigma(W_{(x,R)} \cdot [u_{t-1}, x_t] + b_R) \quad (2)$$

In this expression, W is the weight, u_{t-1} is the previous state, x_t is the current input, and b_Q is the bias vector.

After calculating the parameters of the update and reset

gates using the above equations, the gate signal values are obtained through Sigmoid or tanh activation functions. Then, the reset gate computes the candidate state at the current time step, followed by updating the hidden state. Consequently, the neural network's output at the current time step is calculated using the following equation:

$$\tilde{H}_t = \tanh(\omega_H \cdot [R_t \cdot H_{t-1}, x_t] + b_H) \quad (3)$$

$$H_t = (1-Q_t) \cdot H_{t-1} + Q_t \cdot H_t \quad (4)$$

Here, $\tilde{H}_t$ represents the candidate hidden state, ω_H denotes the weight matrix of the candidate hidden state, tanh is the activation function, x_t indicates the current input, b_H represents the bias of the candidate hidden state, and Q_t denotes the hidden state at the current time step.

2.2. CAM

The introduction of channel attention mechanisms originates from addressing feature redundancy and insufficient information utilization in deep neural networks. Early convolutional neural networks (CNN) often produced numerous highly correlated feature channels when processing images, harboring abundant redundant information. In order to exploit key information within these feature channels, researchers proposed relevant CAM. The core idea involves assigning distinct weights to each channel, thus dynamically adjusting their significance to emphasize critical features and suppress irrelevant data. The structure of the model is shown in Figure 3

To capture global context for each channel, global information is aggregated. Global Average Pooling and Global Max Pooling are applied to the input feature map across spatial dimensions, generating two distinct channel descriptors. The Global Average Pooling layer is formulated as shown in Equation 5.

$$Z_{avg} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X(c,i,j), \forall_c \in \{1,2,....,C\} \quad (5)$$

For an input feature $X \in \mathrm{R}^{C \times H \times W}$, C represents the channel count. Dimensions H and W indicate the feature map's height and width, while $Z_{avg} \in \mathrm{R}^C$ denotes the channel descriptor derived via Global Average Pooling.

The Global Max Pooling layer is shown:

$$Z_{\max} = \max_{i,j} X(c,i,j), \forall_c \in \{1,2,.....,C\} \quad (6)$$

Where $Z_{\max} \in \mathrm{R}^C$ denotes the channel descriptor obtained via Global Max Pooling.

Fully connected layers and nonlinear activations are used to learn inter-channel dependencies and generate channel attention weights. Specifically, the aggregated channel descriptors are encoded by a shared feedforward network, which typically includes two fully connected layers and a nonlinear activation. The equations for the fully connected layers and activation functions in both average and max pooling branches are shown in Equations 7 and 8.

$$s_{avg} = \sigma(W_2 \cdot RELU(W_1 \cdot z_{avg})) \quad (7)$$

$$s_{\max} = \sigma(W_2 \cdot RELU(W_1 \cdot z_{\max})) \quad (8)$$

In this formulation, $W_1 \in \mathrm{R}^{\frac{C}{r} \times C}$ and $W_2 \in \mathrm{R}^{C \times \frac{C}{r}}$ are the learnable weights in the fully connected layers, and r denotes the channel compression ratio. The functions $\mathrm{Re}LU$ and σ represent the activation functions, while $s_{avg}, s_{\max} \in \mathrm{R}^C$ indicate the attention weights for average and max pooling, respectively.

The attention weights obtained from Global Average Pooling and Global Max Pooling are summed to produce the final channel attention weights.

$$S = S_{avg} + S_{\max}, S \in \mathrm{R}^C \quad (9)$$

Here, $S \in \mathrm{R}^C$ signifies the final channel attention weights.

The generated channel attention weights are applied through channel-wise multiplication to modulate the original feature responses.

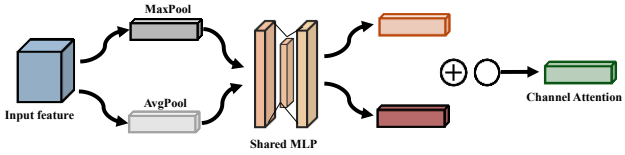

Figure 3. Google Scholar

2.3. Aquila Optimizer

The AO algorithm is inspired by the hunting behavior of hawks, emulating their efficiency and agility in prey search processes. The algorithm constructs a population of agents that iteratively update their positions within the search space to progressively approach or attain the global optimum. AO integrates multiple optimization strategies, including swarm intelligence, random search, and Levy flight steps, enhancing global search capabilities and preventing local optima entrapment. The specific mathematical modeling process of the AO algorithm is outlined below.

2.3.1. Initialization Phase

In the initial phase, the AO algorithm randomly generates a set of hawk agents (solution vectors) to form the initial population. Each hawk agent's position vector x_i is uniformly distributed within the defined search space. The number of hawk agents in the population is set to N .For each hawk agent $x_i (i = 1,2,...,N)$, each dimension $x_{i,j} (j = 1,2,...,d)$ of its position vector is randomly initialized within the specified bounds $[lb_j, ub_j]$, is shown:

$$x_{i,j} = lb_j + rand() \times (ub_j - lb_j) \quad (10)$$

Here, $rand()$ represents a uniformly random number within the interval $[0,1]$.

The fitness value f_i for each hawk agent's position vector x_i is calculated using the objective function $f(x)$, can be expressed as:

$$f_i = f(x_i) \quad (11)$$

Based on optimization objectives, fitness values are ranked to identify the best agent X^* in the current population, as shown:

$$x^* = \begin{cases} \arg\min_{x_i} f(x_i) \\ \arg\max_{x_i} f(x_i) \end{cases} \quad (12)$$

2.3.2. Levy Flight Step Calculation

Levy flight is a type of random step with a heavy-tailed distribution, utilized to enhance the algorithm's global search capability. The calculation formula for the Levy flight step L is follows:

$$L = \frac{u}{|v|^{1/\beta}} \times multiplier \quad (13)$$

Where $u \sim N(0, \sigma^2)$: u is a normally distributed random number with mean 0 and standard deviation σ; $v \sim N(0,1)$: v is a normally distributed random number with mean 0 and standard deviation 1; β is the Levy index, typically between 1 and 3, controlling step distribution characteristics; σ is the step scale parameter, calculated as:

$$\sigma = \left(\frac{\Gamma(1+\beta)\sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2})\beta 2^{\frac{\beta-1}{2}}}\right)^{\frac{1}{\beta}} \quad (14)$$

Here, $\Gamma(\cdot)$ denotes the Gamma function.

2.3.3. Position Update

The AO algorithm iteratively updates the hawk agents' position vectors to gradually approach the global optimal solution. Position updates are divided into exploration and exploitation phases, determined by the current iteration count. The exploration and exploitation phase are shown in Figure 4

(1) Exploration Phase

During the exploration phase, the algorithm focuses on global search to avoid entrapment in local optima. It primarily updates hawk positions using the following two strategies:

Strategy One: Move towards the global best direction, as shown in Equation 15.

$$x_i^{new} = x^* \times (1 - \frac{t}{T}) + rand() \times (\bar{x} - x^*) \quad (15)$$

Here, t denotes the current iteration count, T the maximum iterations, $rand()$ a random number in $[0,1]$, and $\bar{X}$ the current population's average position vector.

Strategy Two: Levy flight combined with random agent influence, as shown:

$$x_i^{new} = x^* \times L + x_j + rand() \times (y - x) \quad (16)$$

Here, L is the Levy flight step length, x_j is the position vector of a randomly selected other hawk agent, and x and y are predefined parameter vectors.

(2) Exploitation Phase

During the exploitation phase, the algorithm emphasizes local search to accelerate convergence. It primarily updates hawk positions using the following two strategies:

Strategy One: Adjust based on average position and global best solution, as shown in Equation 17:

$$x_i^{new} = \alpha \times (x^* - \bar{x}) - rand() \times (rand() \times (ub - lb) + lb) \times \delta \quad (17)$$

Here, α and δ are control parameters, while ub and lb are the upper and lower bounds of decision variables.

Strategy Two: Comprehensive adjustment based on the global best solution, as shown:

$$x_i^{new} = QF \times x^* - (g_2 \times x_j \times rand()) - g_2 \times L + rand() \times g_1 \quad (18)$$

Here, $QF = t^{\frac{2 \times rand() - 1}{(1-T)^2}}$, $g_1 = 2 \times rand() - 1$, $g_2 = 2 \times (1 - \frac{t}{T})$.

2.3.4. Boundary Correction

After position updates, clipping is employed to ensure hawk agent position vectors x remain within predefined bounds.

$$x_i^{new} = clip(x_i^{new}, lb, ub) \quad (19)$$

Here, the clip operation restricts each dimension of x_i^{new} within the $[lb_j, ub_j]$ range.

2.3.5. Fitness Evaluation and Selection

The fitness value $f(x_i^{new})$ of updated hawk agents is computed. Based on optimization goals, the new fitness is compared with the original, retaining the superior agents, as shown in Equation 20.

$$x_i = \begin{cases} x_i^{new} \\ x_i \end{cases} \quad (20)$$

At each generation's end, the population is re-ranked to update the global best solution x^*. The algorithm terminates when the maximum iterations T are reached or other convergence criteria are met, outputting the global best solution x^* and its fitness value $f(x^*)$.

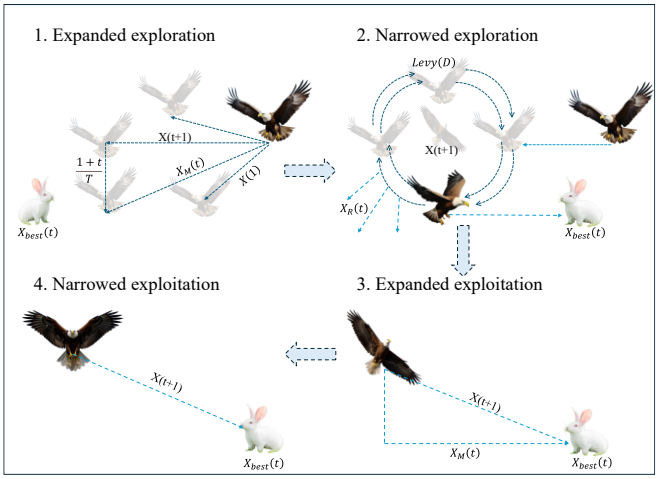

Figure 4. AO Position Update

3. Data Preprocessing and Experimental Setup

3.1. Dataset Description

The battery dataset utilized in this study was collected through testing conducted by Dr. Phillip Kollmeyer's team at the University of Wisconsin-Madison under simulated real-world driving conditions. To accurately emulate electric vehicle battery loading conditions, the battery cells were tested under five temperatures (-20°C, -10°C, 0°C, 10°C, and 25°C) across parameters Cycle1, Cycle2, Cycle3, Cycle4, Neural Network, HWFET (Highway Fuel Economy Test), LA92 (LA92 Dynamometer Driving Schedule), UDDS (Urban Dynamometer Driving Schedule), and US06 (Supplemental Federal Test Procedure Driving Schedule).

During training, data from Cycle1, Cycle2, Cycle3, Cycle4, Neural Network, and HWFET were used as the training set, while LA92, UDDS, and US06 data at various temperatures served as the test set to evaluate model performance. For better predictions and comparison of experimental results., the State of Charge (SOC) was calculated using the ampere-hour integration method. The formula for the ampere-hour integration method is presented below:

$$SOC(t) = SOC(t_0) - \frac{1}{Q_{Ah}} \int_{t_0}^{t} I(t)dt \tag{21}$$

Here Q_{ah} is the rated capacity of the battery, $I(t)$ is the battery current.

By selecting and partitioning the dataset, this study effectively simulates various complex real-world EV operational scenarios, ensuring the proposed AO-BiGRU-CAM model's efficiency and robustness in dynamic environments. The dataset includes records of key parameters such as voltage, current, temperature, and capacity variations under different temperatures, providing abundant foundational data for SOC prediction model development and optimization. The part of data is presented in Figure 5.

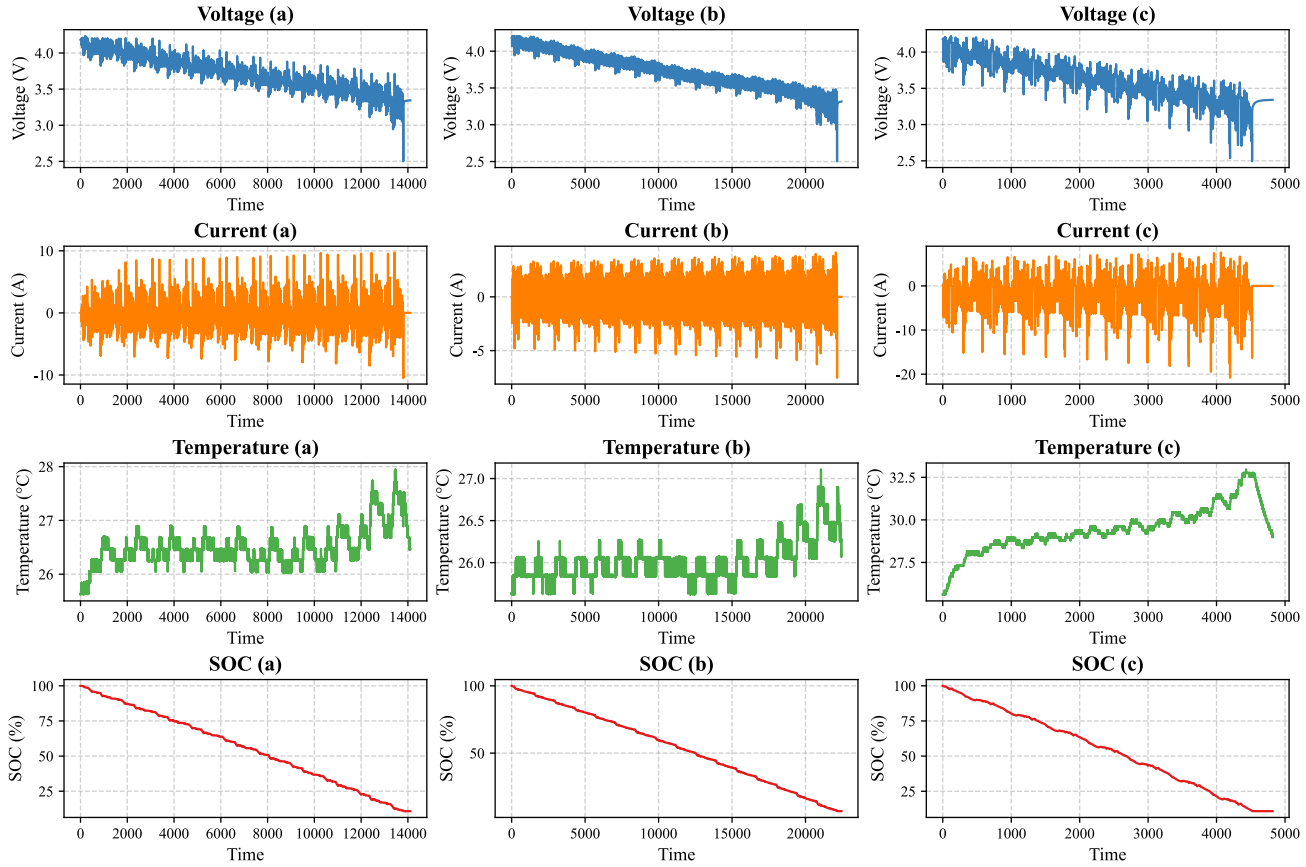

Figure 5. Current, voltage temperature and SOC profiles of the 25 °C: (a) LA92; (b) UDDS; (c) USO6.

3.2. Data Preprocessing

To ensure the model adequately captures SOC variations under multivariate inputs, this study initially employs Pearson correlation coefficients to quantitatively analyze the relationships between each feature and SOC in the raw data. Subsequently, features most influential for SOC prediction are selected for model input base on analysis.

Subsequently, to accelerate model convergence and minimize the impact of differing feature scales on prediction performance, Min-Max Normalization is applied to the selected features for normalization.

3.2.1. Pearson Correlation Analysis

In SOC prediction, typical features include voltage, current, temperature, and other environmental and operational parameters. However, the degree of association between these features and SOC may vary. Therefore, this study employs Pearson correlation coefficients to quantify the linear relationships between each feature and SOC, selecting voltage, current, and temperature as model input features. The calculation formula for the Pearson correlation coefficient ρ is follows:

$$\rho_{XY} = \frac{C_{ov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n} \frac{(X_i - E(X))}{\sigma_X} \frac{(Y_i - E(Y))}{\sigma_Y}}{n} \tag{22}$$

Here, $X:\{X_1, X_2 ..... X_n\}$ and $Y:\{Y_1, Y_2 ..... Y_n\}$ represent the overall data, $C_{ov}(X,Y)$ denotes the covariance between X and Y, σ_X is the standard deviation of X, and σ_Y is the standard deviation of Y.

3.2.2. Data Normalization

The physical dimensions and numerical ranges of different features often vary significantly; for instance, voltage typically ranges from a few volts to several tens of volts, whereas temperature spans from -20°C to 25°C. Directly inputting such data into the model can result in slow convergence or even entrapment in local optima during training. To address this, the study employs Min-Max Normalization to scale features, uniformly mapping them to the [0, 1] interval. The Min-Max Normalization formula as follows:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{23}$$

Here, X represents the original data, $X_{\min}$ and $X_{\max}$ are the data's minimum and maximum values, respectively, and X' is the normalized data.

3.3. Simulation Procedure

The AO algorithm is employed to optimize the BiGRU-CAM model's hyperparameters, including hidden_dim (number of hidden neurons), num_layers (stacked BiGRU layers), num_epochs (training epochs), batch_size, and learning_rate. The hidden_dim range is $[32,256]$, num_layers is $[1,5]$, num_epochs is $[50,500]$, batch_size is $[16,128]$, and learning_rate is within $[1e-4,1e-2]$. To prevent overfitting, an early stopping mechanism was employed during training. The process ceased when validation performance showed no notable improvement for 20 consecutive epochs. The hyperparameters optimized by AO are presented in Table 1. The detailed implementation steps are as follows:

(1) In the AO algorithm, each agent represents a hyperparameter configuration. To enhance population diversity and expedite early convergence, a chaotic map (e.g., the logistic map) initializes the population. The approach generates random values via the chaotic sequence, then maps them into feasible hyperparameter sets based on designated bounds.

(2) During hyperparameter search, each agent's performance is evaluated. The BiGRU-CAM model is trained using that hyperparameter set, and the mean squared error (MSE) on the test set is taken as the fitness score. MSE exhibits higher sensitivity to outliers and can partially reflect prediction stability.

(3) The AO algorithm emulates hawk predation, where time-varying trajectories guide the search. By using chaotic initialization and Levy flight steps, it diversifies updates. At each generation, a new position (i.e., a new hyperparameter set) is computed for each agent. These position-update equations generally incorporate the distance to the global best (g_best), deviations from the population's mean fitness or randomly chosen agents, random Levy flight perturbations, and iteration-based decaying or reinforcing factors.

(4) After each iteration, agents are re-evaluated at their new positions for updated fitness values. The current fitness is compared with the agent's historical best (p_best). If improved, the agent's p_best is updated, and then the global best (g_best) is determined by comparing p_best across the population. This process ensures gradual convergence toward the global optimum during iterative searches.

(5) The AO algorithm terminates when the maximum iteration count is reached or when g_best shows negligible improvement for several consecutive iterations. The globally optimal hyperparameter configuration is then output. If no termination condition is met, the procedure returns to Step 3 for further iterations.

(6) After finalizing the hyperparameter set, the BiGRU-CAM model is retrained. The optimal hyperparameter configuration is shown in the table.

Table 1. Optimal hyperparameters

Configuration	Detail
Early stopping patience	20
hidden_dim	128
num_layers	3
Batch_size	64
Learning Rate	0.0012
Maximum epochs	500

3.4. Evaluation Metrics

This experiment employs mean squared error (MSE), root mean squared logarithmic error (RMSLE), and mean absolute percentage error (MAPE). Their definitions are as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(soc_{real} - soc_i)^2 \tag{24}$$

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(\log(soc_{real}+1) - \log(soc_i+1))^2} \tag{25}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{soc_{real} - soc_i}{soc_i}\right| \times 100\% \tag{26}$$

Here, N is the total number of samples, SOC_{real} represents the actual SOC derived via ampere-hour integration, and SOC_i is the model's estimated value.

3.5. Experimental Setup

All experiments were conducted within the same environment, as detailed in the following table2:

Table 2. Experimental Setup

Configuration	Detail
CPU	Intel® Xeon® Gold 5218 CPU @ 2.30GHz, 2.29 GHz
GPU	NVIDIA GeForce RTX 2080 Ti
RAM	64.0 GB
OS	Win11 x64
CUDA	12.1
PyTorch	2.2.2

4. Results and Discussion

In order to demonstrate the advantage of the prediction performance of the proposed AO-BiGRU-CAM neural network model, and also to show that the optimization algorithm has an enhancing effect on the optimization of model parameters. Four models, GRU, BIGRU, GRU-CAM, and BiGRU-CAM, are selected here to compare the model results. In order to ensure the reliability of the experiment, the network parameters of the proposed method with the same structure as the comparison model are kept consistent. MSE, RMSLE, and MAPE are chosen as the evaluation indexes, and different models are utilized to make predictions in the same way, and the error results are shown in Table 3.

Table 3. SOC estimation under different dynamic operating conditions and different models

Temp (°C)	Cycle	Metric	AO-BiGRU-CAM	BiGRU	GRU	BiGRU-CAM	GRU-CAM
-20℃	LA92	MSE	2.56E-04	2.77E-04	2.96E-04	2.99E-04	2.66E-04
		RMSLE	9.93E-03	1.02E-02	1.08E-02	1.07E-02	1.00E-02
		MAPE	1.95E-02	2.01E-02	2.12E-02	2.14E-02	1.98E-02
	UDDS	MSE	2.55E-04	5.08E-04	4.12E-04	2.92E-04	3.27E-04
		RMSLE	9.27E-03	1.23E-02	1.15E-02	1.00E-02	1.03E-02
		MAPE	1.79E-02	2.18E-02	2.04E-02	1.88E-02	1.94E-02
	US06	MSE	4.88E-04	6.69E-04	6.72E-04	4.91E-04	5.29E-04
		RMSLE	1.47E-02	1.75E-02	1.76E-02	1.48E-02	1.52E-02
		MAPE	3.11E-02	3.61E-02	3.34E-02	3.08E-02	2.97E-02
-10℃	LA92	MSE	1.57E-04	2.94E-04	3.11E-04	1.92E-04	2.43E-04
		RMSLE	8.07E-03	1.13E-02	1.18E-02	8.98E-03	9.99E-03
		MAPE	1.71E-02	2.32E-02	2.49E-02	1.94E-02	2.21E-02
	UDDS	MSE	2.67E-04	3.62E-04	3.18E-04	2.51E-04	3.37E-04
		RMSLE	1.11E-02	1.21E-02	1.14E-02	1.03E-02	1.22E-02
		MAPE	2.07E-02	2.62E-02	2.31E-02	2.18E-02	2.46E-02
	US06	MSE	2.66E-04	2.87E-04	7.02E-04	3.07E-04	3.49E-04
		RMSLE	1.10E-02	1.12E-02	1.85E-02	1.17E-02	1.27E-02
		MAPE	2.48E-02	2.54E-02	4.34E-02	2.65E-02	2.94E-02
0℃	LA92	MSE	1.54E-04	1.60E-04	2.27E-04	1.59E-04	1.77E-04
		RMSLE	8.18E-03	8.15E-03	9.52E-03	8.28E-03	8.58E-03
		MAPE	2.04E-02	2.02E-02	2.44E-02	2.14E-02	2.17E-02
	UDDS	MSE	1.43E-04	2.30E-04	2.22E-04	1.67E-04	2.07E-04
		RMSLE	6.70E-03	1.00E-02	1.00E-02	8.57E-03	9.64E-03
		MAPE	1.75E-02	2.36E-02	2.40E-02	2.02E-01	2.31E-02
	US06	MSE	2.34E-04	2.90E-04	2.78E-04	2.40E-04	2.82E-04
		RMSLE	1.05E-02	1.24E-02	1.36E-02	1.12E-02	1.18E-02
		MAPE	2.58E-02	3.49E-02	3.03E-02	3.04E-02	3.24E-02
10℃	LA92	MSE	6.81E-05	1.44E-04	2.45E-04	1.23E-04	1.25E-04
		RMSLE	5.58E-03	8.16E-03	1.01E-02	7.37E-03	7.45E-03
		MAPE	1.36E-02	2.01E-03	2.63E-02	1.87E-02	2.00E-02
	UDDS	MSE	6.47E-04	8.10E-04	5.49E-04	3.04E-04	5.56E-04
		RMSLE	1.86E-02	2.23E-02	1.85E-02	1.24E-02	1.87E-02
		MAPE	1.44E-02	3.15E-02	2.64E-02	2.19E-02	1.71E-02
	US06	MSE	1.77E-04	3.15E-04	7.67E-04	2.85E-04	2.07E-04
		RMSLE	9.76E-03	1.19E-02	2.03E-02	1.11E-02	1.00E-02
		MAPE	2.67E-02	3.31E-03	3.51E-02	2.74E-02	2.51E-03
25℃	LA92	MSE	1.37E-05	6.48E-05	8.71E-05	2.77E-05	5.43E-05
		RMSLE	2.57E-03	5.66E-03	6.32E-03	3.72E-03	5.32E-03
		MAPE	7.73E-03	1.76E-02	1.89E-02	1.19E-02	1.79E-02
	UDDS	MSE	6.43E-05	7.10E-05	1.23E-04	3.63E-05	4.57E-05
		RMSLE	5.76E-03	5.64E-03	7.34E-03	4.67E-03	4.92E-03
		MAPE	2.21E-02	1.92E-02	2.43E-02	1.78E-02	1.74E-02
	US06	MSE	5.37E-05	2.96E-04	1.21E-04	1.28E-04	1.60E-04
		RMSLE	5.89E-03	1.34E-02	7.98E-03	8.48E-03	9.43E-03
		MAPE	2.26E-02	5.28E-02	2.46E-02	3.31E-02	3.55E-02

From Table 3, it is evident that the proposed AO-BiGRU-CAM model outperforms GRU, BiGRU, GRU-CAM, and BiGRU-CAM under most test conditions. These include varying temperatures and drive cycles. This model yields lower MSE, RMSLE, and MAPE values, achieving superior predictive accuracy overall.

Under extreme low temperatures (-20°C and -10°C), traditional GRU and BiGRU exhibit higher prediction errors. They lack deeper temporal feature extraction and attention-based mechanisms. In contrast, AO-BiGRU-CAM achieves smaller overall errors.At normal (0°C) and high (10°C, 25°C) temperatures, various models demonstrate certain accuracy improvements. AO-BiGRU-CAM maintains an advantage in most scenarios and metrics, showcasing strong adaptability and robustness. In both LA92, UDDS, and US06 drive cycles, AO-BiGRU-CAM achieves superior MSE, RMSLE, and MAPE metrics in most cases. This finding indicates that the model aptly captures crucial factors influencing predictive accuracy across varying complexities and loading conditions. In some scenarios, BiGRU-CAM performs comparably to AO-BiGRU-CAM. This outcome suggests that incorporating attention mechanisms enhances the model's focus on salient

features. Through parameter optimization, AO-BiGRU-CAM undergoes more targeted calibration of network parameters. Its performance becomes more stable and pronounced.

By comparing with other top-performing models, AO-BiGRU-CAM exhibits a 5%–10% reduction in error metrics. This finding demonstrates that introducing an optimization algorithm can enhance training efficiency under identical structural configurations, leading to more optimal parameters. This improvement not only elevates predictive accuracy but also fosters greater stability during training.

5. Conclusion

Addressing Li-ion battery SOC estimation, this study introduces the AO-BiGRU-CAM model, which employs AO to optimize BiGRU-CAM hyperparameters for various operating conditions. Experimental findings indicate superior predictive accuracy and stability across diverse temperatures and drive cycles, with enhanced focus on key features outperforming traditional alternatives. This study contributes novel methodologies to augment battery management system reliability and practicality, laying a foundation for exploring more advanced optimization strategies and architectures. Future research can incorporate large-scale validation under real operational environments, further unveiling the model's generalization capacity in practical vehicular applications.

References

[1] Miri I, Fotouhi A, Ewin N. Electric vehicle energy consumption modelling and estimation—A case study [J]. International Journal of Energy Research, 2021, 45(1): 501-520.

[2] Sanguesa, J. A., Torres-Sanz, V., Garrido, P., Martinez, F. J., & Marquez-Barja, J. M. (2021). A review on electric vehicles: Technologies and challenges. Smart Cities, 4(1), 372-404.

[3] Pramuanjaroenkij, A., & Kakaç, S. (2023). The fuel cell electric vehicles: The highlight review. International Journal of Hydrogen Energy, 48(25), 9401-9425.

[4] Pradhan, S. K., & Chakraborty, B. (2022). Battery management strategies: An essential review for battery state of health monitoring techniques. Journal of energy storage, 51, 104427.

[5] Adaikkappan, M., & Sathiyamoorthy, N. (2022). Modeling, state of charge estimation, and charging of lithium-ion battery in electric vehicle: a review. International Journal of Energy Research, 46(3), 2141-2165.

[6] Liu, Y., He, Y., Bian, H., Guo, W., & Zhang, X. (2022). A review of lithium-ion battery state of charge estimation based on deep learning: Directions for improvement and future trends. Journal of Energy Storage, 52, 104664.

[7] Selvaraj, V., & Vairavasundaram, I. (2023). A comprehensive review of state of charge estimation in lithium-ion batteries used in electric vehicles. Journal of Energy Storage, 72, 108777.

[8] Sesidhar, D. V. S. R., Badachi, C., & Green II, R. C. (2023). A review on data-driven SOC estimation with Li-Ion batteries: Implementation methods & future aspirations. Journal of Energy Storage, 72, 108420.

[9] Pillai, P., Sundaresan, S., Kumar, P., Pattipati, K. R., & Balasingam, B. (2022). Open-circuit voltage models for battery management systems: A review. Energies, 15(18), 6803.

[10] Movassagh, K., Raihan, A., Balasingam, B., & Pattipati, K. (2021). A critical look at coulomb counting approach for state of charge estimation in batteries. Energies, 14(14), 4074.

[11] Tran, M. K., Mathew, M., Janhunen, S., Panchal, S., Raahemifar, K., Fraser, R., & Fowler, M. (2021). A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters. Journal of Energy Storage, 43, 103252.

[12] Liu, K., Gao, Y., Zhu, C., Li, K., Fei, M., Peng, C., ... & Han, Q. L. (2022). Electrochemical modeling and parameterization towards control-oriented management of lithium-ion batteries. Control Engineering Practice, 124, 105176.

[13] Eleftheriadis, P., Giazitzis, S., Leva, S., & Ogliari, E. (2023). Data-driven methods for the state of charge estimation of lithium-ion batteries: An overview. Forecasting, 5(3), 576-599.

[14] Hannan, M. A., Lipu, M. H., Hussain, A., Ker, P. J., Mahlia, T. I., Mansor, M., ... & Dong, Z. Y. (2020). Toward enhanced state of charge estimation of lithium-ion batteries using optimized machine learning techniques. Scientific reports, 10(1), 4687.

[15] Vidal, C., Malysz, P., Kollmeyer, P., & Emadi, A. (2020). Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art. Ieee Access, 8, 52796-52814.

[16] Ng, M. F., Zhao, J., Yan, Q., Conduit, G. J., & Seh, Z. W. (2020). Predicting the state of charge and health of batteries using data-driven machine learning. Nature Machine Intelligence, 2(3), 161-170.

[17] Chandran, V., Patil, C. K., Karthick, A., Ganeshaperumal, D., Rahim, R., & Ghosh, A. (2021). State of charge estimation of lithium-ion battery for electric vehicles using machine learning algorithms. World Electric Vehicle Journal, 12(1), 38.

[18] Babaeiyazdi I, Rezaei-Zare A, Shokrzadeh S. State of charge prediction of EV Li-ion batteries using EIS: A machine learning approach [J]. Energy, 2021, 223: 120116.

[19] Castanho, D., Guerreiro, M., Silva, L., Eckert, J., Antonini Alves, T., Tadano, Y. D. S., ... & Corrêa, F. C. (2022). Method for SoC estimation in lithium-ion batteries based on multiple linear regression and particle swarm optimization. Energies, 15(19), 6881.

[20] Antón, J. Á., Nieto, P. G., de Cos Juez, F. J., Lasheras, F. S., Vega, M. G., & Gutiérrez, M. R. (2013). Battery state-of-charge estimator using the SVM technique. Applied Mathematical Modelling, 37(9), 6244-6253.

[21] Lipu, M. H., Hannan, M. A., Hussain, A., Ansari, S., Rahman, S. A., Saad, M. H., & Muttaqi, K. M. (2022). Real-time state of charge estimation of lithium-ion batteries using optimized random forest regression algorithm. IEEE Transactions on Intelligent Vehicles, 8(1), 639-648.

[22] Deng, Z., Hu, X., Lin, X., Che, Y., Xu, L., & Guo, W. (2020). Data-driven state of charge estimation for lithium-ion battery packs based on Gaussian process regression. Energy, 205, 118000.

[23] Bahdanau, D. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

[24] Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[25] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[26] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497).

[27] Jiao M, Wang D, Qiu J. A GRU-RNN based momentum optimized algorithm for SOC estimation [J]. Journal of Power Sources, 2020, 459: 228051.

[28] Hong, J., Wang, Z., Chen, W., Wang, L. Y., & Qu, C. (2020). Online joint-prediction of multi-forward-step battery SOC using LSTM neural networks and multiple linear regression for real-world electric vehicles. Journal of Energy Storage, 30, 101459.

[29] Yan, B., Zheng, W., Tang, D., LaiLi, Y., & Xing, Y. (2023). A knowledge-constrained CNN-BiLSTM model for lithium-ion batteries state-of-charge estimation. Microelectronics Reliability, 150, 115112.

[30] Yang, K., Wang, Y., Tang, Y., Zhang, S., & Zhang, Z. (2023). A temporal convolution and gated recurrent unit network with attention for state of charge estimation of lithium-ion batteries. Journal of Energy Storage, 72, 108774.

[31] Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., & Gandomi, A. H. (2021). Aquila optimizer: a novel meta-heuristic optimization algorithm. Computers & Industrial Engineering, 157, 107250.

[32] Liu, Y., Shao, Z., & Hoffmann, N. (2021). Global attention mechanism: Retain information to enhance channel-spatial interactions. arXiv preprint arXiv: 2112.05561.